Modélisation d'événements composites répétitifs, propriétés et relations temporelles

Cyril Faucher

UNIVERSITÉ DE LA ROCHELLE

ÉCOLE DOCTORALE S2IM

Laboratoire L3i (Informatique, Image, Interaction)

THÈSE présentée par :

Cyril FAUCHER

soutenue le : **13 décembre 2012**
pour obtenir le grade de : **Docteur de l'Université de La Rochelle**
Discipline : **Informatique et Applications**

Modélisation d'événements composites répétitifs, propriétés et relations temporelles

---

JURY :

Mireille BLAY-FORNARINO	Professeur, Université Nice Sophia Antipolis, Rapporteur
Yamine AIT AMEUR	Professeur, Université Paul Sabatier - Toulouse III, Rapporteur
Gérard LIGOZAT	Professeur émérite, Université Paris-Sud Orsay, Examinateur, Président du jury
Frédéric MALLET	Maitre de Conférences HDR, Université Nice Sophia Antipolis, Examinateur
Jean-Yves LAFAYE	Professeur, Université de La Rochelle, Directeur de thèse
Frédéric BERTRAND	Maitre de Conférences, Université de La Rochelle, Encadrant de thèse

Thèse réalisée au Laboratoire L3i
Pôle Sciences & Technologies, Université de La Rochelle
avenue Michel Crépeau
17042 La Rochelle cedex 01

Tél : +33 5 46 45 82 62
Fax : +33 5 46 45 82 42
Web : http ://l3i.univ-larochelle.fr/

Sous la direction de Jean-Yves Lafaye jean-yves.lafaye@univ-lr.fr

Encadrement Frédéric Bertrand frederic.bertrand@univ-lr.fr

Financement 2009-2011 : Projet ANR RelaxMultiMedias 2
2011-2012 : 1/2 ATER IUT de La Rochelle

Cyril Faucher

cyril.faucher@univ-lr.fr

septembre 2009 - décembre 2012

Remerciements

Tout d'abord je remercie vivement les membres du jury qui m'ont fait l'honneur de juger ce travail de thèse. Je tiens à présenter mes sincères remerciements à Pr Gérard Ligozat d'avoir présidé mon jury de thèse et pour son apport scientifique lors de sa venue à La Rochelle lors d'un séminaire. Je remercie Pr Mireille Blay-Fornarino et Pr Yamine Ait Ameur pour avoir rapporté ce manuscrit de thèse et pour les conseils qu'ils ont pu apporter pour donner une touche finale à la rédaction. Je tiens également à remercier Dr Frédéric Mallet pour avoir accepté de participer au jury en tant qu'examinateur.

Des remerciements très appuyés et toute ma reconnaissance à mes encadrants, Pr Jean-Yves Lafaye en tant que Directeur de thèse et Dr Frédéric Bertrand. Il est difficile de résumer en quelques mots près de dix années de travail en commun. Les années se sont succédées depuis 2002, tout d'abord comme étudiant puis comme doctorant depuis 2009, sans perte de motivation ni d'envie pour toujours avancer ensemble. Une disponibilité et une réactivité à toute épreuve pour avancer et passer les épreuves. Merci pour votre encadrement, vos relectures et les relations humaines que nous avons développées et entretenues toutes ces années.

Merci aux membres du projet RelaxMultiMedias 2, Mathieu Bully, Denis Teyssou, Jean-Luc Minel, Delphine Battistelli, Dina Garnier-Oeliarisoa, Morgan Banville et tout particulièrement à Charles Teissèdre avec qui j'ai partagé de bons moments de science et de sympathie notamment lors d'EKAW'10.

Je souhaite également remercier les directions successives du L3i pour leur accueil et leur soutien pendant ces trois dernières années.

Un grand merci à tous ceux que j'ai pu rencontrer au L3i pendant ces trois années : doctorants, ATER, post-docs, ingénieurs, stagiaires, permanents et assistants à la recherche. Il est difficile de tous les citer ici sans oublier quelqu'un, mais j'ai une pensée pour tous et la vie au laboratoire et en dehors a été plus agréable grâce à leur bonne humeur notamment lors des incontournables déjeuners !

Merci aux membres du Département Informatique de l'IUT de La Rochelle pour leur accueil et à ceux qui m'ont formé et confié des enseignements depuis 2009, plus particulièrement à Marie-Christine Lafaye, Annick Lassus, Georges Louis, Jamal Malki, Marie-Hélène Verrons, Philippe Coulaud et Philippe Crottereau.

Je profite de ces remerciements pour avoir une pensée pour ceux avec qui j'ai pu travailler et échanger avant de débuter cette thèse. Hormis la découverte de la Bretagne coté Mer et coté Terre, mes deux expériences au CNRS puis à INRIA ont été déterminantes pour ce travail de thèse, les compétences acquises durant ces deux périodes ont largement contribué à me faire devenir informaticien, à aimer la recherche et donc à la réussite de cette thèse.

Tout d'abord une pensée pour les membres de Géomer que j'ai côtoyé pendant près de deux ans : Mathias Rouan, Cyril Tissot, Jacqueline et Emmanuel Giraudet et Françoise Gourmelon qui malgré des circonstances très douloureuses à accepter de reprendre la direction de mon stage.

Une pensée également pour les membres de l'équipe Triskell de l'Irisa qui sont désormais aux quatre coins de la planète. À ceux qui m'ont fait confiance pendant près de trois années : Jean-Marc Jézéquel, Didier Vojtisek, Olivier Barais et les développeurs de Kermeta. Une pensée spéciale pour François Tanguy pour son amitié et son esprit de compétition positif.

Une pensée plus qu'amicale à Franck et Damien pour notre petite aventure qui je l'espère deviendra grande !

À Nathalie qui a permis que ce dernier été passe plus vite !

À Antoine pour son ultime regard avant l'impression.

À Bruno, Christophe, Clément, Dounia, Élodie, Émilie, Fida, François, Gaël, Guillaume, Julie, Matthieu, Maroua, Mélanie, Mickaël, Nicolas, Omar, les Romain, Rouaa, Salomée, Sophea et Thomas.

À mes amis que j'ai trop délaissés ces derniers mois à qui je n'ai pas cessé de penser pour arriver au but.

À mes parents et ma famille qui m'ont toujours soutenu et apporté leur aide matérielle et morale.

C'est avec émotion que je conclue ces remerciements pour celle qui illumine mes soirées et week-ends, merci ma Sylvie pour ton soutien sans faille durant ces si longs derniers mois. La légèreté des premiers jours peut être maintenant de retour.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Première partie

Contexte et état de l'art

« An event is "something that happens" by definition. For the news industry, it is "something that happens and is subject to news coverage." All the events in a day make up an "agenda", which can be a marketable product sold to clients or simply an internal daybook used by editors to organise their work. »

(EventsML G2, International Press and Telecommunication Counsil)

Chapitre 1

Introduction

Les travaux présentés dans cette thèse concernent la gestion d'événements répétitifs et composites que nous appliquons au domaine des médias (agence de presse). Notre contribution concerne la modélisation, la vérification et la validation des propriétés temporelles définissant les périodes d'occurrences des événements. Notre approche s'intéresse à leur exploitation pour les restituer aux utilisateurs soit par un système d'interrogation soit par des interfaces de visualisation. Nous utilisons l'Ingénierie Dirigée par les Modèles (IDM) pour mettre en œuvre nos approches [Faucher 12a] et abordons nos problématiques dans le contexte de l'ingénierie des Systèmes d'Informations (SI). Nous sommes amenés à nous intéresser à la géométrie et à la topologie du temps en nous basant sur les travaux d'Allen et en utilisant des résultats issus du Traitement Automatique de la Langue (TAL).

1.1 Contexte - RelaxMultiMedias 2

Présentation générale

Ce travail a été réalisé dans le cadre du projet ANR RelaxMultiMedias 2 [1] (RMM2, 2009-2011) conduit en partenariat avec les agences de presse Relaxnews [2] et AFP [3] et le laboratoire MoDyCo [4] spécialisé en Traitement Automatique des Langues (TAL).

RMM2 avait pour objectif de spécifier et de réaliser une chaîne complète de production et de gestion d'événements dans le domaine de l'information de loisir. Cette chaîne a été mise en œuvre par une plate-forme logicielle [5] qui intègre des sources d'informations hétérogènes et assure la validation, l'interrogation et la visualisation des informations traitées.

Le L3i a piloté deux lots de ce projet ANR portant sur la modélisation et la visualisation d'événements issus de dépêches d'information. Le projet RMM2 fait suite à un précédent projet ANR RelaxMultiMedias (2005) pour lequel nous avions utilisé l'IDM pour transformer un modèle métier objet annoté en une ontologie [Faucher 08] afin d'interroger, avec les outils du web sémantique, les concepts principaux d'une dépêche (personnes, lieux, objets...).

Trois thèses, dont celle-ci, sont réalisées dans le cadre du projet RMM2. Dans ce manuscrit nous référençons à plusieurs reprises la thèse de Charles Teissèdre dont l'objectif est de contribuer à la fois à la reconnaissance d'expressions temporelles en intension (exprimant une périodicité temporelle) [Carnap 47] dans des textes en langage naturel et à la visualisation des événements associés sous forme de chronologies.

Les événements dans les dépêches

Les agences de presse partenaires créent des événements touristiques et culturels à partir d'informations apparaissant dans des dépêches ou bien à partir de contenus issus du web (pages, flux RSS [6], Données Ouvertes, etc).

La création d'événements repose initialement en quasi-totalité sur un travail humain de collecte d'information. D'un point de vue métier, les événements sont des éléments de dépêches à intégrer dans des agendas thématiques afin d'en planifier la couverture médiatique. Les événements ainsi constitués sont ensuite proposés comme sujets d'intérêt aux clients : hebdomadaires, magazines, etc.

Un événement est défini comme « *Quelque chose qui se produit à un moment et en un lieu donné* », cette définition est issue de l'IPTC [7] qui est l'organisme de normalisation

1. http://www.rmm2.org
2. http://www.relaxnews.com/
3. http://www.afp.com/
4. http://www.modyco.fr/
5. http://www.afprelaxnews.com
6. http://www.rss-specifications.com/
7. http://www.iptc.org

international pour les agences de presse. Selon cette définition, un événement possède nécessairement une composante temporelle et spatiale. En ce qui concerne le spatial, nous l'aborderons sans contribution spécifique. En effet, nous nous reposons sur les nombreux travaux existants et notamment sur la série de normes ISO 19100 [ISO/TC211 10].

Les événements portent quatre types d'information descriptives qu'il est possible de trouver dans une dépêche, l'IPTC les appelle les quatre « W ». Ils sont relatifs au *Who* (qui, les personnes concernées), *What* (quoi, l'objet), *Where* (où, le lieu), *When* (quand, la date d'occurrence).

Prenons l'exemple d'un texte annonçant le festival des Francofolies de La Rochelle de 2011 (figure 1.1.1) qui est un extrait de l'article « La Rochelle : les Francofolies dévoilent leurs têtes d'affiche 2011 » du journal Sud Ouest du 31 janvier 2011 (source[8]). Les fragments de texte surlignés correspondent à des lieux (*Where*), les encadrés sont des entités nommées [entités nommées] de type *What*, les soulignés à des dates (*When*), les encadrés arrondis sont des (personnes) (*Who*).

La Rochelle : les [Francofolies] dévoilent leurs têtes d'affiche 2011.
« Sud Ouest » révèle [le programme] en exclusivité des soirées Saint-Jean-d'Acre du 12 au 16 juillet.
Une offre de billets à prix serrés seront en vente dès le mardi 1er février.
Programme en exclusivité :
Mardi 12 juillet : (Christophe Maé), (Zaz), (Ours)
Jeudi 14 juillet : (Lilly Wood and the Prick), (Cocoon), (The Do)

FIGURE 1.1.1: Texte en langage naturel annonçant les Francofolies de La Rochelle 2011 extrait du journal Sud Ouest du 31 janvier 2011

Les journalistes interprètent ce texte et vont en déduire des événements simples ou des hiérarchies d'événements à plusieurs niveaux de granularité (composites). En effet, les événements décrits sont souvent complexes, i.e. : ils possèdent des sous-événements dont il est intéressant de conserver les relations mutuelles.

Événements répétitifs

Il s'avère également que les événements gérés par les agences de presse sont souvent répétitifs, comme par exemple un événement sportif ou une exposition qui aura lieu *« tous les jours de 10h à 20h sauf le lundi durant le mois de février 2012 »*. Ce type d'expression est dit « exprimé en intension » *versus* « en extension » [Carnap 47], c'est-à-dire qu'une personne va utiliser des termes du langage naturel pour définir quand va avoir lieu un événement plutôt que de donner la série complète des dates se référant à

8. `http://relaxmultimedia2.univ-lr.fr/resources.html`

un calendrier (cf. annexes A et B). Suivant le contexte (temps et espace), une personne sera capable d'interpréter cette expression en intension et de déterminer la prochaine date d'occurrence (extension) de l'événement considéré.

Origine et qualité des données

Comme nous l'avons vu précédemment, les sources de données contenant l'information temporelle sont représentées par des textes de dépêches qui annoncent les événements. Ces dépêches sont rédigées en langage naturel. Dans le cadre du projet RMM2, le laboratoire MoDyCo et la société Mondeca ont travaillé à la détection assistée des événements et de leurs métadonnées dans ces textes. À partir des éléments détectés et de l'expertise métier (qui valide les éléments extraits), un ensemble d'annotations est alors produit. Ces annotations sont stockées dans un format de données semi-structurées (XML). Elles sont suffisamment riches sémantiquement pour que nous puissions réaliser une modélisation objet pertinente de ces expressions temporelles afin de mettre en œuvre des corpus que nous interrogeons pour restituer l'information à des journalistes ou à des utilisateurs finaux.

Le travail d'extraction de l'information, de validation et d'insertion dans la base sont des points clés notamment du fait que les informations sont issues de textes en langage naturel et, par conséquent, avec une forte probabilité que des ambiguïtés soient introduites. Il est nécessaire de s'assurer que les dates sont correctes syntaxiquement et sémantiquement avant d'être interrogées ou visualisées par un utilisateur.

Gestion des occurrences dans les SI des agences de presse

Pour les agences de presse, il est stratégique de stocker les informations liées aux événements dans des systèmes d'information pour, d'une part, fournir des contenus éditoriaux aux médias (clients), mais également pour prévoir les événements à venir à moyen ou long terme. La prédiction d'événements se traduit par un potentiel de demandes de contenu qui engendrent une activité créatrice de revenus pour l'entreprise. Ainsi l'objectif est d'extraire des dépêches, des dates d'événements qui seront validées par les journalistes. Cette validation par un humain nécessite que les langages que nous proposons soient proches du langage naturel. De nombreux travaux dans la communauté TAL consistent à extraire et à interpréter des dates dans des textes en langage naturel [Weiser 08, Teissèdre 10, Battistelli 11].

Un de nos objectifs est de participer en aval à la vérification et la validation de la structure et de la sémantique des expressions temporelles extraites en les reformulant sous forme de textes structurés selon une grammaire (formelle) interprétable par une machine. Enfin, la structuration et les relations qui sont identifiées entre les événements sont des éléments de premier ordre pour les journalistes.

Exploitation et cas d'utilisation

Les événements traités sont destinés à être interrogés et visualisés par un journaliste.

Le journaliste doit pouvoir retrouver les événements et prévoir leurs occurrences, c'est-à-dire, accéder aux propriétés temporelles des événements, périodiques inclus. Il peut, par exemple, vouloir connaître tous les événements qui auront lieu « *les lundis de juillet de 10h à 19h* ». Les occurrences de ces événements ne sont pas encore dans le système d'information, en revanche il est possible d'interroger les propriétés temporelles en intension qui les décrivent.

La visualisation des événements consiste à les rendre disponibles sur une chronologie et une carte, et de pouvoir les filtrer par catégories.

Les événements que le journaliste considère comme pertinents servent de base pour constituer l'agenda d'un envoyé spécial chargé de les couvrir. Cet agenda peut être peuplé à partir de données issues des standards tels que EventsML G2[9] et iCalendar [Desruisseaux 09]. Ainsi, il est important de fournir des passerelles entre nos modèles et de tels standards. Pour réaliser ces importations et exportations, l'usage de l'IDM et des transformations de modèles est une solution adéquate et plus généralement pour prendre en compte l'interopérabilité des données [Authosserre-Cavarero 12].

Contribution de cette thèse au projet RMM2

Un des aspects originaux de notre travail est de proposer un (méta-)modèle d'événement dont la partie temporelle permet l'expression, la vérification et la validation de propriétés temporelles en intension. Ce modèle est pivot, il est muni de transformations de modèles pour mettre en œuvre l'interopérabilité avec les standards (EventsML G2, iCalendar). Il possède également des grammaires qui en sont une contre-partie textuelle. Les principaux objectifs de RMM2, présentés dans le processus général de la figure 1.1.2, se résument pour notre contribution à :

- la conception de logiciels d'aide au processus de création d'événements (jusque-là intégralement manuel). Il s'agit de fournir au responsable de la création d'événement une proposition initiale résultant de l'analyse automatique des dépêches et de sources complémentaires. L'activité humaine peut alors se concentrer sur la validation de la proposition et à son éventuel enrichissement. L'utilisation de syntaxe concrète et d'un vérificateur sémantique permettent de contrôler la correction finale des événements saisis.
- le développement de passerelles d'importation et d'exportation d'événements représentées dans les formats métiers existants (NewsML G2 et EventsML G2, RSS, iCalendar).
- la diffusion d'événements à travers différents types de plates-formes (mobiles, PC) supportant des interfaces graphiques variées.

9. `http://www.iptc.org/site/News_Exchange_Formats/`

– l'interrogation des expressions temporelles en intension afin de retrouver des événements dans une base de données pour aider les journalistes à constituer leur agenda d'événements à couvrir.

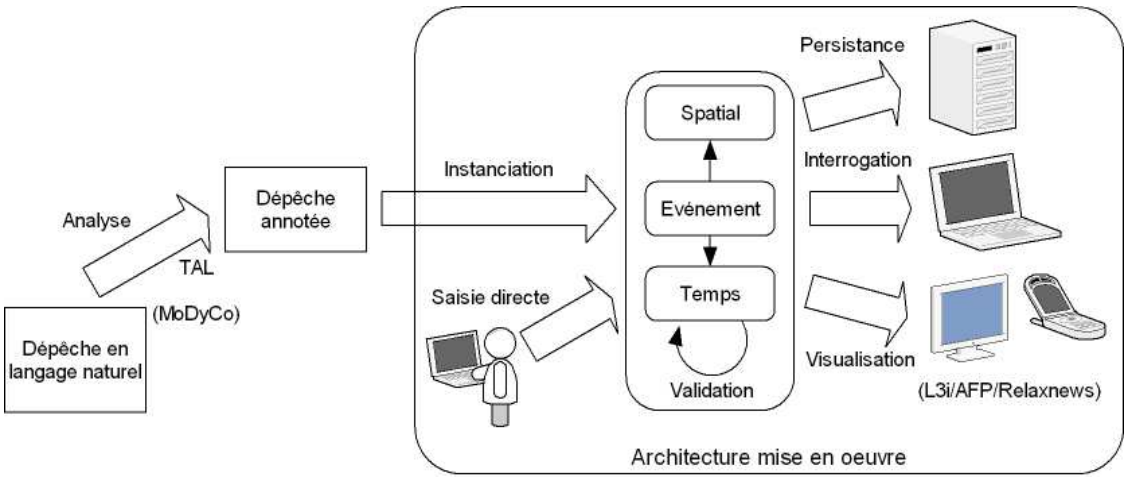


FIGURE 1.1.2: Architecture générale

Ces objectifs nécessitent de gérer des informations complexes, que ce soit pour la représentation interne des événements ou pour produire des interfaces de consultation. Il convient également de garantir la qualité des informations saisies ; vérifier leur structure et s'assurer que leur sémantique est valide en exploitant, par exemple, la connaissance métier contenue dans des modèles de référence. Ces besoins nous ont naturellement conduits à utiliser l'IDM qui permet :

– de modéliser et de rendre interopérable la grande variété de formats utilisés ;
– d'automatiser, par l'intermédiaire de transformations de modèles, la traduction entre ces différents formats et la génération d'applications de visualisation
– de bénéficier de l'ensemble des outils de développement de la plate-forme Eclipse EMF [10] (Eclipse Modeling Framework), KERMETA [Muller 05], ainsi que des outils développés dans les projets TOPCASED [11] et OPENEMBEDD [12].

1.2 Problématiques

Nos objectifs principaux consistent à :

1. définir un modèle pour représenter la notion d'événement pour des activités (loisirs, divertissement, tourisme, sport...). Le modèle comporte une partie temporelle

10. http://www.eclipse.org/modeling/EMF/
11. http://www.topcased.org/
12. http://openembedd.org/

permettant d'exprimer en intension quand les événements ont lieu et d'une partie structurelle pour gérer des hiérarchies d'événements (composite) ;

2. définir un moyen d'interaction avec l'utilisateur final dans un langage proche de la langue naturelle, i.e. : une reformulation sous forme textuelle.
3. définir un processus de vérification et de validation assurant que l'expression temporelle décrivant les occurrences d'un événement soit syntaxiquement et sémantiquement correcte ;
4. développer des outils d'importation, d'exportation, de visualisation et d'interrogation permettant d'exploiter les événements créés.

Nous allons ci-dessous détailler ces différents objectifs.

1.2.1 Modéliser des événements et des expressions temporelles

Afin de manipuler et exploiter des événements avec des outils informatiques, il est nécessaire de les modéliser, plus particulièrement leur structure et leurs relations mutuelles. Il sera, dans un premier temps, intéressant d'étudier les modèles d'événements existants traitant de leur structuration pour réutiliser ou tirer parti des expertises en ce domaine.

Un événement possède également des propriétés temporelles décrivant ses occurrences. Celles-ci s'expriment soit en extension, soit en intension. De nombreux travaux et mises en œuvre traitent des expressions en extension. En revanche, l'expression en intension fait l'objet de beaucoup moins d'études. Nous porterons notre attention sur les expressions en intension que ce soit en terme de modélisation, de manipulation ou d'exploitation afin de réduire leur volume et de préserver leur sémantique. En effet, s'appuyer sur la forme intensionnelle permet d'accéder à l'ensemble des occurrences possibles pour un événement avec une sémantique maîtrisée, alors qu'en extension on ne peut travailler que sur les occurrences qui peuplent effectivement une base à un instant donné ; les phénomènes périodiques sont alors enfouis et difficilement calculables.

Nous nous intéressons particulièrement aux travaux sur les événements périodiques, les logiques temporelles, les calendriers et les relations temporelles entre événements. Les travaux de J.F. Allen et de G. Ligozat et les standards tels que l'ISO 19108 [ISO 02] et iCalendar constituent les fondements de notre travail.

Comme nous l'avons indiqué précédemment, nous nous plaçons à l'interface de l'ingénierie des systèmes d'information et du traitement automatique des langues. Les propriétés temporelles des événements nous sont fournies par les experts du TAL sous forme de textes annotés. À partir de ces annotations, nous souhaitons instancier le modèle d'événement et sa partie temporelle. Pour réaliser cela, une passerelle est à développer pour traduire les textes annotés en événements. L'IDM est, à notre connaissance, un des moyens les mieux adaptés à cette réalisation. Pour utiliser ce type d'ingénierie, il sera nécessaire d'identifier les correspondances entre un modèle d'annotation linguistique et notre modèle.

1.2.2 Vérifier et valider des expressions temporelles

Pour garantir la cohérence des événements créés, une activité de vérification et validation est nécessaire qui assure que l'interprétation du texte en langage naturel est correcte et complète. L'objectif est de vérifier syntaxiquement et sémantiquement les expressions. Notre but n'est pas de définir des contraintes de sûreté et de vivacité entre événements. La cohérence des informations temporelles peut être intrinsèque (e.g. : la fin d'un événement ne peut précéder son début) mais également calendaire (e.g. : « *le 8^e jour de la semaine* » n'existe pas). Nous offrons ainsi à l'utilisateur le moyen de visualiser un texte reformulant l'expression conformément à une grammaire issue de notre modèle afin d'éviter certaines ambiguïtés liées à l'usage du langage naturel.

1.2.3 Manipuler et exploiter des événements et des expressions temporelles

Nous souhaitons manipuler et exploiter les événements créés à travers des applications d'interrogation et de visualisation d'événements. Comme pour la modélisation, l'interrogation d'expressions temporelles en extension a été traitée par de nombreuses contributions et les systèmes de gestion de bases de données proposent des fonctions avancées de recherche temporelle. Cependant, pour des événements périodiques cela implique que toutes les occurrences soient créées ce qui signifie un nombre important de dates à traiter.

Par exemple, le corpus de dépêches liées aux loisirs de Relaxnews est de taille moyenne (20000 par an). Cependant le facteur qui augmente la quantité d'information est le nombre d'occurrences de chacun de ces événements. En effet, nous avons pu constater une moyenne de 163 occurrences par an et par événement. Si l'historique de ces événements est conservé, le volume d'information augmentera rapidement au cours du temps. Par conséquent des structures et modes d'interrogation adéquats doivent être choisis pour gérer efficacement ces données.

Comme nous l'avons précédemment indiqué, notre objectif est de réduire la quantité de données stockées et de préserver l'information représentant la périodicité temporelle. Ainsi nous proposons de spécifier un système d'interrogation d'expressions temporelles en intension qui consiste en une structure de données et un langage de requête dont l'objectif sera de retrouver les événements possédant les propriétés temporelles recherchées. Il s'agit de démontrer la faisabilité et l'efficacité de ce système d'interrogation par rapport aux méthodes classiques d'interrogation de dates en extension.

Exploiter les modèles consiste également à proposer des passerelles (bidirectionnelles) avec les standards d'échange d'événements comme EventsML G2 et iCalendar.

Afin de visualiser les événements, nous proposons d'utiliser une bibliothèque de composants graphiques, Simile Exhibit [13], qui offre la possibilité de créer aisément des chro-

13. `www.simile-widgets.org/exhibit/`

nologies (*timelines*). L'utilisation de cette bibliothèque nécessite la configuration de paramètres pour chaque composant graphique. Ce type de bibliothèque étant soumis à des améliorations et des extensions régulières, adapter une démarche IDM facilite la prise en compte de ces mises à jour : générer les applications et créer des familles d'application où les points de variabilité concernent la disposition des composants et les sources de données. Ces dernières sont gérées avec une approche générique mettant en correspondance ces sources avec les composants graphiques les affichant.

1.2.4 Problématiques détaillées

Cette section illustre certains aspects des problématiques énoncées ci-dessus. Il s'agit de préciser les contributions attendues.

Evénements composites et répétitifs

Nous présentons cette problématique en nous appuyant sur un cas d'étude portant sur un festival annuel de musique française : les « *Francofolies* » de La Rochelle (France).

Depuis 1985, le festival des Francofolies a lieu chaque année autour du 14 juillet et se reproduira tant qu'il y aura des sponsors, du public et des artistes, etc. Par conséquent ce festival peut se répéter une infinité de fois. De manière générique, ce festival est identifié et nommé « *Francofolies* », mais concrètement ce sont les éditions qui ont lieu : « *Francofolies1985*, ... , *Francofolies2012* », elles forment une série d'événements se répétant chaque année. Ainsi, « *Francofolies* » est un concept regroupant d'autres événements spécifiques.

Chaque édition des « *Francofolies* » se compose de différents spectacles donnés chaque jour pendant le festival comme ceux de la « *Grande Scène* » (cf. figure 1.2.1), cette décomposition est purement temporelle au sens où leur déroulement est défini par la périodicité liée à une édition (i.e. : « *tous les jours* »).

Chaque jour, les Francofolies rassemblent également plusieurs spectacles spécifiques, tel que le « *SFR Jeunes Talents* » (JT, cf. figure 1.2.2) qui accueille des débutants sur scène, ou « *La fête à Xyz* » (FA) qui offre à un artiste confirmé la possibilité d'inviter les confrères de son choix au cours de son spectacle. Ainsi plusieurs points de vue peuvent être adoptés :

- d'une part, pour une année donnée, e.g. 2011, l'événement « *Francofolies2011* » est vu comme un ensemble d'événements spécifiques, e.g. : la série de JT ou de FA, chacune ayant son programme quotidien spécifique ;
- d'autre part, « *Francofolies2011* » se présente sous forme d'un programme quotidien, chaque élément du programme quotidien contenant des événements spécifiques tels que les JT ou FA.

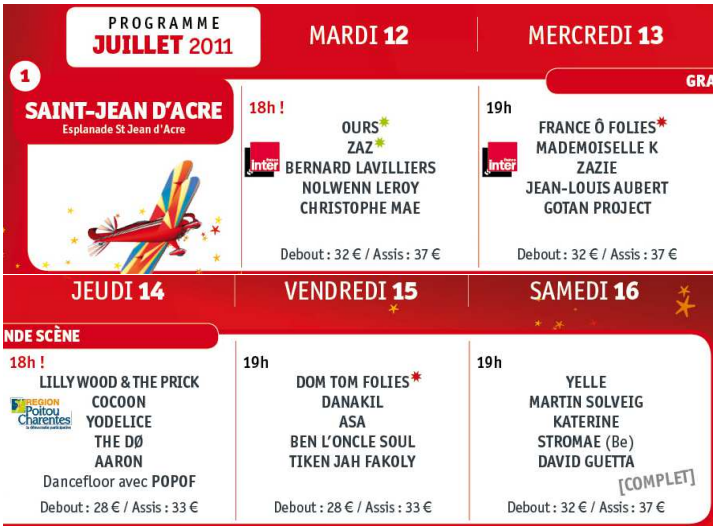


FIGURE 1.2.1: Programme journalier de la Grande Scène pour le festival des Francofolies 2011

Les propriétés temporelles liées à un événement composite donnent des informations sur l'étendue temporelle durant laquelle les occurrences des sous-événements peuvent se produire, e.g. : tous les sous-événements des Francofolies (concerts) se déroulent pendant une édition des Francofolies. Les propriétés temporelles d'un événement simple (i.e. : sans sous-événement modélisé) apportent des informations sur la périodicité de ses occurrences, e.g. : « *le musée est ouvert tous les jours* ».

On notera qu'un site de publication d'événements comme EVENTFUL [14] propose de créer des sous-événements attachés à un événement de taille importante. Ceci démontre un besoin réel d'organiser les événements. EVENTFUL et d'autres sites comme UPCOMING [15] de Yahoo et QUIVIENT [16] proposent de créer des événements répétitifs. Le métamodèle d'événements composites doit prendre en considération les aspects temporels et structurels. Ils seront au cœur de sa conception qui sera présentée dans le chapitre 3.

Expressions temporelles en intension issues de textes en langage naturel : période d'accessibilité

Notre travail porte principalement sur des données issues de textes en langage naturel. Dans ces textes sont définis des événements décrits par des expressions temporelles portant sur les dates et horaires auxquels ils surviennent, comme par exemple, une période d'ouverture d'un musée. Dans la suite nous parlons de « période ou horaire d'accessibilité » comme cela est défini dans [Teissèdre 10, Faucher 10c]. En effet le terme

14. `http://eventful.com`
15. `http://upcoming.yahoo.com`
16. `http://www.quivientmanger.fr/`

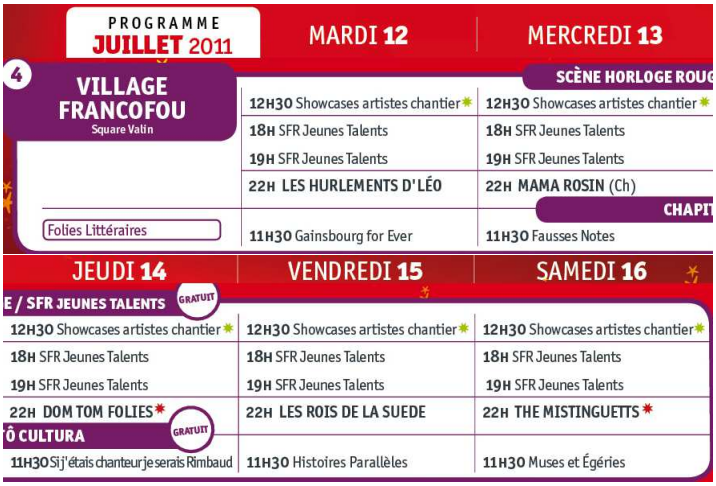


FIGURE 1.2.2: Programme journalier des spectacles annexes du festival des Francofolies 2011

de période « d'ouverture » implique le statut « ouvert » pour l'élément considéré. Par conséquent, nous préférons utiliser le terme « d'accessibilité » qui permet d'abstraire ce statut. Les statuts peuvent être : « *ouvert* », « *fermé* », « *réservé aux professionnels* », etc.

L'accessibilité est également définie par une des quatre relations de GOSSELIN [Becher 06], qui agrège quatre relations d'ALLEN : *Finished_by*, *Contains*, *Begun_by* et *equals*, pour former la relation d'accessibilité.

Les périodes d'accessibilité sont utilisées pour des lieux ouverts au public. Elles sont représentées sous forme d'instants pour définir des débuts d'activités : « *départ à 14h, 15h ...* » (cf. détails en annexe B.1.1) ou des périodes décrivant les horaires d'ouverture d'un lieu : « *de 10h00 à 18h30* » (cf. figure 1.2.3 et détails en annexe B.1.2).

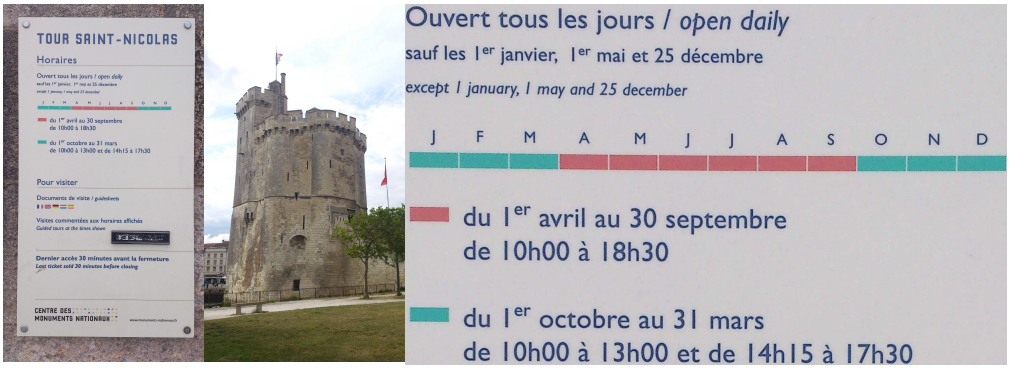


FIGURE 1.2.3: Période d'accessibilité de la Tour Saint Nicolas de La Rochelle

Certaines expriment la règlementation d'activités : « *horaires limitant la vente sur*

les marchés », « *la pêche à pied sur les côtes* » (cf. détails en annexe A) ou encore la circulation automobile : « *interdiction de stationner du coucher du soleil au lever du soleil* » (cf. figure B.1.6).

Le langage naturel est parfois utilisé pour compléter des données déjà structurées par exemple en XML. Des notes comme : « *fermé le mardi après-midi et le jeudi matin* » sont souvent ajoutées pour préciser une exception alors que la partie nominale de l'expression (horaires classiques) est sous forme semi-structurée et facilement interprétable (cf. annexe C).

L'usage du langage naturel ou bien des erreurs humaines lors de la saisie dans des formulaires peuvent engendrer des ambiguïtés ou des imprécisions sur l'accessibilité à une ressource (cf. figures B.1.5, B.2.1).

L'accessibilité est également parfois dépendante d'événements externes (e.g. : « *marées, phases de la Lune, coucher/levé du soleil, travaux, etc* ») comme le montrent les figures A.1.2, A.2.1, B.1.4 et B.1.6.

Ces différents exemples soulignent la nécessité de proposer un cadre de modélisation et de vérification et validation pour les expressions temporelles en intension dédiées aux activités humaines en général et aux loisirs en particulier.

Dépendances temporelles entre événements

Comme nous l'avons décrit précédemment, l'occurrence d'un événement est parfois dépendante d'autres événements, cette dépendance peut être périodique : « *interdiction de stationner du coucher du soleil au lever du soleil* », dans cet exemple le coucher et le lever du soleil sont des phénomènes périodiques. Cette dépendance peut également s'exprimer dans un cas sans récurrence, ou encore un site qui est fermé exceptionnellement « *pour travaux ou mauvaises conditions météo* » (figure B.1.4) qui sont deux phénomènes non périodiques et très aléatoires, aucune prévision ne peut être faite.

Nous avons également été confrontés à la modélisation de dépendances temporelles entre des événements dans le cadre d'une collaboration avec le laboratoire Géomer (UMR 6554) [Faucher 10e] et lors de notre participation à une action du GDR-MAGIS [17], qui travaille à catégoriser et comparer les différentes approches du temps dans les pratiques des modélisateurs.

En effet, la règlementation d'activités humaines comme la pêche à pied s'exprime sous forme de textes en langage naturel dans lesquels, par exemple, un événement « *Pêche autorisée* » est lié avec l'événement « *Marée basse* ». Ainsi la règlementation stipule que la pêche à la Telline est autorisée *« de 3h avant à 3h après la basse mer* » (règle complète en figure A.1.2).

Ce problème peut se résumer à définir des relations topologiques (avant, égal, après) entre instants périodiques (le début de l'autorisation est en relation avec le début de la

17. `http://magis.ecole-navale.fr/`, action « Analyse des dynamiques spatiales et simulation, pour un débroussaillage du temps en géomatique » qui rassemble depuis décembre 2009 des chercheurs issus des communautés des géographes, des géomaticiens, des historiens et des informaticiens.

basse mer et de même pour les fins) en quantifiant cet écart (3 heures).

Selon un point de vue topologique, nous souhaitons modéliser et stocker des informations sur des instants ou des intervalles périodiques (i.e. : intervalles non convexes) pour définir des relations entre des éléments calendaires comme : *« mardi suit lundi »*. Ces relations sont ici dites « qualitatives ». Les travaux d'Allen [Allen 83] définissent treize relations qualitatives uniques entre intervalles. Dans ce cas, les intervalles sont convexes, alors que nos intervalles se répètent au cours du temps et forment des intervalles non convexes.

Ainsi pour exprimer des relations de dépendance entre de tels événements, il est nécessaire d'étudier à la fois les relations quantitatives entre instants périodiques et les relations qualitatives entre intervalles non convexes.

1.3 Propositions

Nos propositions concernent l'ensemble du processus annoncé précédemment, i.e. : de la modélisation à l'exploitation des événements ainsi que la vérification et la validation de leurs propriétés temporelles :

- Définir un métamodèle d'événements répétitifs composites (nommé Ptom : Periodic Temporal Occurrence Metamodel) comprenant une forte composante temporelle qui résulte de l'extension de la norme ISO 19108. Des concepts propres sont à ajouter comme des instants périodiques dont l'occurrence est définie de façon absolue ou relative.
- Des contraintes sont à définir afin de vérifier structurellement et sémantiquement les expressions temporelles en s'appuyant sur un modèle de calendrier.
- Nous souhaitons également offrir un générateur automatique de textes proche du langage naturel permettant à un utilisateur de vérifier l'interprétation de l'information saisie, pour ce faire plusieurs grammaires sont à développer afin d'offrir à l'utilisateur un moyen convivial pour lire et écrire des modèles d'événements composites accompagnés d'expressions temporelles.
- Concernant la topologie du temps, nous proposons des relations qualitatives entre intervalles non convexes, elles sont nécessaires pour exprimer des positionnements relatifs entre événements répétitifs comme pour définir que « *chaque semaine, le mardi suit le lundi* ». Le métamodèle Ptom et ces relations permettront de modéliser le calendrier grégorien, dès lors que l'on considère que les éléments constituants le calendrier sont eux-mêmes des événements. Un modèle de correspondance doit associer chaque événement ainsi défini à un élément de Ptom.
- Pour faire la jonction entre les analyses du TAL et le métamodèle Ptom, une transformation de modèles est nécessaire afin de passer de textes en langage naturel annotés vers des modèles de Ptom. Les sources de données sont multiples et hétérogènes : modèle de période d'accès, flux RSS, fichier XML, données ouvertes institutionnelles, textes naturels contraints. Ceci permettra de valider une partie

la chaîne de traitements allant de la modélisation à l'exploitation du corpus de dépêches du projet ANR RMM2.
– D'un point de vue exploitation du métamodèle d'événements répétitifs : un système d'interrogation est à définir et à développer afin de retrouver des événements dans une base de données et ceci en interrogeant des expressions temporelles en intension. Un générateur d'applications de visualisation est à développer afin de proposer différents moyens pour visualiser des événements avec une forte composante spatio-temporelle. Les données à afficher sont des flux d'événements provenant des agences Relaxnews et AFP.

1.4 Organisation de la thèse

La **partie I** du manuscrit positionne les travaux de cette thèse leur contexte et donne l'état de l'art. Le **chapitre 1** a introduit le contexte, les problématiques et le projet RelaxMultiMedias 2 qui a servi de support applicatif à cette thèse. Le **chapitre 2** donne l'état de l'art, nous faisons un rappel sur la modélisation du temps. Ensuite nous abordons les événements composites, suivis par la modélisation et l'expression de la périodicité des événements. Nous présentons des applicatifs gérant des événements et leurs propriétés temporelles. Pour finir, nous évoquons les contraintes temporelles qui peuvent être spécifiées entre événements.

La **partie II** est consacrée à la conception d'un langage spécifique pour le domaine des événements répétitifs et composites et des relations temporelles entre événements. Cela nous a amené à la production du métamodèle nommé Ptom et à l'ensemble de relations nommé Allen*. Le **chapitre 3** décrit dans un premier temps la modélisation structurelle des événements et leurs inter-relations. Puis dans un deuxième temps un langage d'expression de la périodicité en utilisant des concepts proches du langage naturel (expressions temporelles en intension). Cette discussion autour de la modélisation continue par la description des outils fournis avec le métamodèle Ptom pour faciliter l'utilisation du langage par des journalistes. Le **chapitre 4**, quant à lui vise à fournir une formalisation de relations qualitatives entre intervalles non convexes utilisées pour définir des relations temporelles entre des événements répétitifs. L'objectif étant de fournir à un utilisateur un ensemble de relations (Allen*) proches de la connaissance commune en s'inspirant de celles d'Allen mais cette fois-ci entre des intervalles convexes. Le **chapitre 5** est destiné à montrer l'utilisation de ces relations pour modéliser les calendriers et en particulier le grégorien. Cette modélisation est ensuite utilisée comme fondement pour la vérification sémantique d'expressions temporelles.

La **partie III** est consacrée aux aspects applicatifs exploitant les différents modèles et outils décrits précédemment. Le **chapitre 6** montre la création de modèles de Ptom à partir d'interface web ou bien de textes en langage naturel. Puis, nous discutons les problèmes d'interopérabilité avec des standards comme iCalendar et EventsML G2. Ensuite, nous décrivons l'exploitation de propriétés temporelles en intension pour interroger

et retrouver des événements dans une base de données. Enfin, ce chapitre décrit une démarche IDM pour générer des applications de visualisation d'événements à caractère spatio-temporel. Le **chapitre 7** est consacré aux cas d'application montrant l'utilisation des différentes approches et outils développés dans cette thèse. Tout d'abord nous proposons de mettre en œuvre nos approches sur un exemple tiré de travaux antérieurs comme ceux de Terenziani. Puis, nous abordons le cas des festivités de la ville Gand. Ensuite, nous montrons l'utilité de Ptom pour modéliser des propriétés temporelles dans le cadre d'une plate-forme de simulation multi-agent. Puis, nous décrivons la chaîne de production journalistique du projet RelaxMultiMedias 2 en insistant sur nos contributions. Pour finir, nous donnons des résultats d'expérimentation du système d'interrogation Ptom-S.

Pour conclure, le **chapitre 8** rappelle les contributions majeures de notre travail et indique les perspectives jugées nécessaires et prometteuses.

Chapitre 2

Temps et événements composites périodiques

2.1 Introduction

Le temps, les événements, et les calendriers sont sujets à diverses représentations et interprétations suivant le contexte applicatif. En informatique, les événements sont principalement liés à des automates. Ils sont généralement des éléments simples qui peuvent transmettre des informations et conduisent à des changements d'états (transitions) dans des conditions spécifiées. Notre contribution traite des constructions d'événements complexes, en effet l'objectif est la modélisation d'événements culturels (festival, saison théâtrale), sportifs et de loisirs (championnats, tournois...). Les événements ont généralement une durée et peuvent se décomposer en sous-événements, ayant chacun plusieurs occurrences situées dans l'espace et le temps. Nous adoptons un point de vue « système

d'information » plutôt qu'une approche « systèmes de transitions ».

Ce chapitre est organisé en deux parties, la section 2.2 traitant du temps et des événements composites et la section 2.3 dédiée à l'étude des phénomènes périodiques ou simplement répétitifs.

2.2 Temps et événements composites

Cette section présente les différents points qui nous intéressent dans la modélisation du temps et des phénomènes temporels. Les événements et le temps sont des concepts intimement liés. Le temps étant compté en référence à l'occurrence d'événements répétitifs, et les propriétés des événements étant souvent liées à la chronologie de leurs occurrences. Nous citons et discutons certains des principaux travaux antérieurs et leurs apports majeurs en la matière, en privilégiant naturellement les applications situées dans le périmètre de l'objectif de cette thèse, tel qu'énoncé au chapitre précédent.

La section 2.2.1 est dédiée à la modélisation et au traitement du temps, tandis que la section 2.2.2 est consacrée aux événements, à leur structure et à leurs propriétés.

2.2.1 Le temps

Le temps intervient dans de nombreuses disciplines, que ce soit lors de la conception ou de l'exploitation de systèmes d'information spatio-temporelle, dans des problèmes d'ordonnancement de processus, en vérification de contraintes, en dynamique des systèmes, en fouille de données, en intelligence artificielle, ou encore en traitement de la langue naturelle.

Dans la section 2.2.1.1, nous donnons des considérations générales communes aux champs applicatifs que nous venons de citer. Puis, nous traitons successivement en section 2.2.1.2 des principales approches existantes permettant de modéliser, de calculer et de raisonner sur le temps.

2.2.1.1 Modélisation du temps : Généralités

Lorsque nous parlons ici des propriétés du temps, nous nous intéressons en fait aux caractéristiques d'un système qui permet de représenter, d'analyser, de prévoir, en un mot de gérer les occurrences d'événements.

Dans ce contexte, les propriétés du temps peuvent être de trois ordres : géométrique, topologique et structurel. À ces préoccupations fondamentales, s'ajoutent des considérations opérationnelles que l'on peut rattacher à la problématique des horloges et des calendriers. Nous aborderons ces points successivement.

Géométrie

D'un point de vue formel, la géométrie du temps se fonde sur un ordre linéaire dans lequel les instants sont des éléments de dimension nulle et les intervalles des éléments de dimension un. Les intervalles convexes jouent un rôle particulier dans la mesure où ils constituent les éléments permettant de construire par union ensembliste des intervalles plus complexes, dont en particulier les intervalles non convexes qui seront d'un intérêt majeur par la suite.

Cette conception primaire est reconnue dans les modèles mathématiques et dans les normes (cf. section 3.4.2.1). Que ce soit conceptuellement (mathématiques) ou bien pratiquement (normes), la géométrie du temps repose donc sur la droite réelle à un isomorphisme près.

Une origine et une unité étant fixées, on peut alors parler de considérations métriques : écart ou de distance entre deux instants et longueur d'un intervalle convexe.

Topologie

Pour nombre de cas d'utilisation, les aspects géométriques quantitatifs sont moins importants que les aspects topologiques, traitant des relations qualitatives mutuelles entre instants ou intervalles. L'ordre des occurrences est significatif alors que la connaissance de leurs dates précises est accessoire. Les propriétés d'intérêt concernent ces aspects ordinaux et non l'échelle numérique sous-jacente.

Les travaux fondamentaux d'Allen sur l'algèbre des relations topologiques binaires entre intervalles convexes sont rappelés et discutés au chapitre 4. De ces travaux émerge une taxonomie intrinsèque des 13 relations de base qui ont une forte sémantique partagée par les utilisateurs et qui est fondée sur des arguments mathématiques clairs.

Des généralisations unifiant la représentation des relations topologiques entre instants et intervalles élargissent de manière élégante le champ d'application de ces théories.

Nous nous intéressons particulièrement aux extensions de ces résultats au cas des intervalles non convexes. La mise en œuvre de ces modèles et concepts est au centre des préoccupations de cette thèse. Le cadre théorique ainsi construit permet de raisonner en garantissant la cohérence des calculs et des conclusions tout en en contrôlant la complexité.

Cependant, la complexité intrinsèque (NP) des calculs pratiques, concernant les relations topologiques entre intervalles non convexes liés à la taille rapidement excessive des ensembles considérés, nuit à la mise en œuvre concrète de théories mathématiques puissantes telles que celle de Ligozat [Ligozat 91, Ligozat 10].

Des travaux proposent des structures (sous-algèbres calculables maximales, relations pré-convexes) qui permettent d'organiser l'ensemble des relations d'Allen étendues ou non, et dans un certain sens réduisent la complexité initiale. Ces résultats proviennent des communautés scientifiques liées aux problèmes de satisfaction de réseaux de contraintes (SAT) et restent en dehors des objectifs premiers que nous nous sommes fixés en rap-

port avec la communauté des Systèmes d'Information (SI). Sur ces questions qui nous animent, il existe peu de littérature.

Aspects structurels

Les différents modèles usuels de représentation du temps construisent tous le concept d'intervalle par agrégation de deux propriétés en caractérisant respectivement les extrémités : instant « début » et instant « fin ». La plupart des normes se limitent au cas où ces deux attributs sont obligatoires, tandis que des modèles théoriques permettent de traiter des intervalles infinis à gauche ou à droite.

Cette construction agrégative structure les concepts d'instant et d'intervalle mais nécessite naturellement d'imposer des contraintes telles que l'antériorité stricte du début par rapport à la fin pour un même intervalle.

S'intéresser aux intervalles non convexes relève également de considérations structurelles, un intervalle non convexe étant obtenu par réunion des éléments d'un ensemble d'intervalles convexes deux à deux disjoints.

Ces structures hiérarchiques amènent naturellement à s'intéresser aux calendriers.

Horloges et calendriers

Les modèles théoriques évoqués ci-dessus prennent leur sens dans la mesure où ils peuvent servir de base à des applications pratiques. Les difficultés sont nombreuses. Le modèle temporel initial est continu, tandis que l'expérience humaine est principalement discrète. La question fondamentale est donc de relier l'ordre linéaire sous-jacent à une série d'observations concrètes. Les horloges et surtout les calendriers, admettent pour cela une série d'approximations qui font, peu ou prou, coïncider le modèle et les faits.

L'expérience pratique se fonde sur la conscience partagée du caractère cyclique de nombre d'événements cosmiques (rythme circadien, phases lunaires, saisons. . .) qui structurent l'ordre linéaire du temps perçu. La difficulté majeure est la non synchronisation rigoureuse de ces événements cycliques. En conséquence, des problèmes de granularité se posent et il est difficile d'intégrer (i.e. : d'unifier de façon cohérente) les différentes instances de modèles temporels spécifiées à des niveaux d'abstraction différents.

Il est hélas indéniable que les applications pratiques ne peuvent se limiter à un seul niveau d'expression.

Cette question à des impacts sur plusieurs domaines :

- la conception des modèles de représentation du temps qui doivent fournir des solutions formelles (concepts, opérateurs) pour gérer ces difficultés
- la définition des calendriers et de leurs différents éléments constitutifs à des granularités différentes
- la spécification de la sémantique des langages exprimant des propriétés temporelles sur des événements, en particulier dans le souci de construire, comme nous envisageons de le faire, des interfaces en langage naturel à l'usage de utilisateurs finaux.

Une autre difficulté réside dans la traduction d'un système calendaire vers un autre. En théorie, la question est simple, mais les distorsions évoquées ci-dessus à propos des calendriers compliquent la tâche en pratique. Ce type de question n'est pas anodin, car un calendrier débute nécessairement à un instant donné ; le calcul rétroactif des dates d'événements ayant des occurrences antérieures à l'origine du calendrier soulève des difficultés du même ordre.

L'ensemble des concepts, opérations et contraintes élémentaires concernant le temps, les calendriers et les systèmes de référence sont spécifiés dans la norme ISO 19108 [ISO 02] qui s'insère dans la série des ISO 19100 [ISO/TC211 10] traitant des métadonnées attachées à l'information géographique. Le temps est ici considéré comme un type de métadonnée particulier. La partie formelle de cette norme utilise UML [OMG 12b] et OCL [OMG 12a].

Une présentation de l'ISO 19108 est donnée en section 3.4.2.1.

La plupart des environnements de gestion d'informations temporelles intègrent une partie des éléments de modélisation spécifiés par la norme et ajoutent des fonctionnalités diverses jugées utiles selon les cas.

2.2.1.2 Modèles et environnements logiciels dédiés au temps

Sur la base des modèles mathématiques représentant le temps et permettant d'analyser des ensembles d'occurrences d'événements, des modèles conceptuels (au sens informatique) ont été spécifiés. Nous nous intéressons principalement aux modèles récents qui relèvent du paradigme objet.

En effet, une modélisation objet favorise actuellement l'implémentation et l'interopérabilité avec les solutions logicielles (elles mêmes majoritairement objet) mais aussi la construction (relativement) facile de ponts, grâce à l'ingénierie dirigée par les modèles, vers d'autres espaces de modélisation tels que les modèles ontologiques (OWL [Patel-Schneider 04], RDF [Klyne 04]), relationnels (CWM [OMG 03]), ou enfin vers tout espace muni d'un DSL (Domain Specific Language, cf. section 3.2.3) formellement spécifié.

Nous traiterons tout d'abord des métamodèles liés à UML/OCL en précisant notre discours dans le cadre de la communauté des systèmes d'information géographique, en faisant référence aux normes et aux outils en usage.

Modélisation Objet du temps : métamodèle UML/OCL

La modélisation du temps a été très peu abordée dans la spécification d'UML2 (noté UML dans la suite). Le paquetage `SimpleTime` d'UML est une partie de la spécification des comportements (*Common Behaviors*) qui a pour vocation de gérer l'expression des contraintes sur les occurrences d'événements dans les machines à états, les diagrammes d'activités ou de séquences. Les notions de bases déjà évoquées, sont disponibles à savoir : `TimeEvent` pour un événement temporisé, `TimeExpression`, `Duration` qui expriment

respectivement une date et une durée sous la forme de textes aux formats normalisés (cf. ISO 8601:2004 [ISO 04]). Des clauses permettent de spécifier des invariants, des gardes et des pré-conditions incluant ces éléments.

Cependant, la sémantique n'est pas clairement définie et le document de l'OMG lui-même, invite à étendre UML pour y intégrer des concepts quant à la gestion du temps. La périodicité n'est pas abordée. Suivant cette recommandation, des extensions d'UML ont été proposées dans divers travaux dont SPT [OMG 05] puis MARTE [OMG 11d]. D'autres extensions ont des vocations plus spécifiques à un domaine particulier, à savoir les systèmes d'information géographiques. Le profil MARTE sera évoqué en section 2.3.4.2, dans le cadre plus général de systèmes d'expression et de vérification de contraintes temporelles. Nous donnons ci-après un aperçu des autres extensions citées.

Extensions temporelles d'UML

[Svinterikou 99] et [Cabot 03] proposent des extensions d'UML sous forme de stéréotypes dont les fonctionnalités permettent de caractériser le caractère ponctuel d'une observation ou sa période de validité.

[Cabot 03] met en œuvre des concepts assez riches qui permettent de définir des périodes sous forme d'intervalles non convexes pour représenter la période de validité d'un attribut ou même d'une association.

Des clauses écrites en OCL permettent de référencer sur des arguments temporels, un état particulier pour un ou plusieurs éléments.

Afin d'enrichir la syntaxe et la sémantique des `TimeExpression` d'UML, Konrad [Konrad 05] propose des patrons textuels pour éditer les propriétés temporelles des objets modélisés. On notera l'intérêt de fournir aux utilisateurs une version textuelle des modèles manipulés notamment lorsque ceux-ci sont proches du langage naturel.

Carlson et Fowler ont eux aussi défini des patrons de conception liés au temps [Carlson 99, Fowler 03]. Ces patrons temporels sont au nombre de trois à savoir : `Snapshot`, `Effectivity` et `Temporal Object`. Ils fournissent des implémentations pour ajouter des attributs temporels au modèle de données d'un système d'information et notamment dans une approche visant à contrôler la dynamique des objets au sein d'un système d'information. Lors d'un précédent projet, nous avions mis en œuvre ces patrons [Faucher 09b].

- Le patron `Snapshot` s'adresse aux objets temporels qui n'ont pas de durée d'existence tels que des événements non mémorisés mais pour lesquels il est utile de disposer d'une date de validité (dates de création, de mise à jour...).
- Le patron `Effectivity` permet de préciser le caractère valide d'une propriété pendant une période de temps avec des dates de début et de fin d'activité. Toutes les instances des périodes seront conservées pour former des historiques. À noter que les périodes peuvent se superposer temporellement.
- Le patron `Temporal Object` permet de localiser un objet dans le temps à travers un rôle de `continuity`, et d'assurer la gestion des versions successives de cet objet au cours de son cycle de vie. Ces versions possèdent un attribut `date` qui correspond à

la création de l'instance. La date de fin de validité pour une instance correspond à la date de création de l'instance de la version suivante. Ceci permet la reconstitution de l'évolution d'un phénomène au cours du temps.

Ces trois patrons de conception ont pour principal intérêt d'encapsuler les attributs temporels et les méthodes associées. Ces propriétés donnent accès aux fonctionnalités telles que l'archivage de chaque état jugé significatif de l'objet stéréotypé, l'accès à l'objet dans l'état courant, précédent ou suivant, l'identification des dates de transition et des événements déclencheurs.

Modélisation du temps dans les Systèmes d'Information Géographique

Dans la communauté des Systèmes d'Information Géographique (SIG), le temps est très présent et souvent associé à des considérations spatiales pour former des propriétés spatio-temporelles. Dans ce contexte, plusieurs recherches ont contribué à étendre les langages de modélisation pour y adjoindre la capacité à prendre en compte ces aspects. Si les principes proposés n'étaient pas initialement spécifiquement dédiés à enrichir des modèles UML, des versions récentes appliquent les principes initiaux avec UML comme cible.

Shaw inventorie les besoins pour la représentation d'événements à caractère spatio-temporel [Shaw 08]. Peuquet [Peuquet 95, Peuquet 02] propose un modèle de données spatio-temporelles fondé sur des événements pour l'analyse temporelle de données géographique.

Tryfona et Jensen [Tryfona 99, Tryfona 00] ont étendu le modèle entité-relation afin de créer un modèle entité-relation spatio-temporel (STER). Price, [Price 00, Price 02] propose « UML spatio-temporel » étendant directement UML.

Un environnement intégré de modélisation d'informations spatio-temporelles incluant un langage dédié connaît un succès certain dans la communauté des SIG. Il s'agit de MADS [Parent 06] qui propose un ensemble de constructions et de représentations pour spécifier de façon indépendante des aspects structurels, spatiaux et temporels attachés aux objets, attributs et associations d'un modèle objet ou entité-relation. Des types d'associations spécifiques stéréotypées rendent compte des relations d'Allen ou encore des relations métier spécialement définies par l'utilisateur. Les événements quant à eux, ne sont pas jusqu'ici intégrés au modèle comme un type particulier dans MADS.

La proposition de la notion d'occurrents est une réponse à cette lacune [Zaki 11]. Par contre, certaines associations reflètent les changements d'états des objets auxquels elles s'appliquent, et sont donc en cela assimilables à des événements.

Guidés par le souci de répondre à des besoins utilisateurs, des auteurs développent des fonctionnalités spécifiques.

Worboys et Hornsby [Worboys 04] combinent la modélisation orientée-objet avec celle basée sur des évènements pour modéliser des domaines d'application géospatiales dynamiques. Les paramètres d'entrée sont des entités spatiales, temporelles ou bien spatio-temporelles.

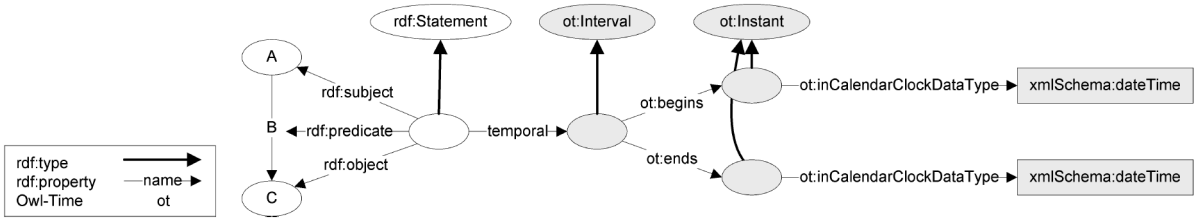
Tissot [Tissot 04, Tillier 10] et Le Tixerant [Le Tixerant 10] simulent des activités humaines à l'aide de Système Multi-Agent (SMA) en ayant comme paramètres des données spatiales et temporelles. Il propose notamment des Calendriers de Pratique Potentielle d'activités (CPP) qui s'appuient pour partie sur des textes règlementaires qui contiennent des expressions temporelles en intension (cf. figures A.1.2, A.1.3 et A.1.4).

Gutierrez [Gutierrez 07] introduit la notion de graphe temporel RDF pour lesquels des propriétés temporelles peuvent être ajoutées aux nœuds. La navigation dans de tels graphes peut être guidée par les propriétés temporelles et en permettre l'analyse.

Avec la même idée Perry [Perry 06, Perry 08] propose d'étendre RDF pour y ajouter une composante temporelle. Il traite des occurrences d'un phénomène mais n'aborde pas la spécification de la périodicité. Dans RDF-ST, Perry réutilise naturellement les concepts d'instant et de période d'OWL-Time (cf. section 2.3.2.1) pour indiquer la période de validité de relations entre concepts.

La figure 2.2.1 montre comment OWL-Time est intégré à un graphe RDF.

Ces travaux se sont poursuivis dans un but de mettre en relation trois composantes d'un événement que sont le lieu, le temps et la thématique [Sheth 08, Jain 08]. Ceci est à rapprocher des quatre « W » (cf. section 1.1). Ces travaux concernent également l'interrogation de ces ontologies notamment en SPARQL avec SPARQL-ST [Perry 07, Perry 11].



(source [Perry 08])

Figure 2.2.1: Extension de RDF avec OWL-Time pour y intégrer des propriétés temporelles

À côté des modèles conceptuels évoqués ci-dessus, il est nécessaire de disposer de modèles ou de recommandations pour implémenter les concepts temporels. L'ISO 8601 définit le codage des dates concrètes sous forme de chaînes de caractères et fait autorité en la matière.

Les SGBD relationnels intègrent maintenant des fonctionnalités temporelles natives puissantes pour gérer les estampilles des attributs et propriétés de la base. Le système offre naturellement l'accès à la date système et une large collection d'opérateurs de traitement des dates et des éléments calendaires. En outre le système scrute et enregistre dans la métabase les opérations CRUD (`create`, `read`, `update`, `delete`) sur les attributs de la base utilisateur. « SQL #3 » offre des opérations d'interrogation actives (e.g. : `version.between(date#1, date#2)`) sur la métabase qui permettent d'accéder à

ces informations, et par ce biais, de gérer les nuplets selon les événements système ayant concerné un déclencheur donné (`create`, `update`, etc) durant une période fixée.

Interface avec des langages et outils dédiés

Le paradigme Objet (et UML en particulier) n'est ni suffisant ni nécessairement adapté au raisonnement temporel. Des langages spécifiques puissants existent qui implémentent les logiques temporelles et les systèmes à horloges et automates temporisés.

En revanche la modélisation objet est efficace pour capturer les concepts métiers et servir de modèle pivot vers les différents utilisateurs. Il est donc naturel de chercher à modéliser un système dans le paradigme objet, puis à traduire les questions spécifiques touchant au raisonnement temporel dans un système mieux adapté. L'IDM permet ou pour le moins facilite l'automatisation de cette traduction.

Ainsi, [Ziemann 03] intègre la logique temporelle linéaire (LTL) *via* une extension d'OCL pour spécifier des clauses de logique temporelle pouvant automatiquement être traduites en LTL.

Il est ainsi possible d'utiliser le pouvoir expressif de la LTL pour spécifier en OCL des invariants ou des *pre/post* conditions. En générant la contrepartie LTL de ces clauses, on peut vérifier les systèmes et mener un raisonnement logique *via* les nombreux outils spécifiquement dédiés à la LTL.

La figure 2.2.2 donne un exemple dans lequel les mots-clés `always` et `until` sont utilisés lors d'une phase d'initialisation d'un système. Ainsi, le système reste (`always`) en mode utilisation jusqu'à (`until`) ce que les unités physiques soient prêtes.

```
context Program inv:
  self.mode = #initialization implies
    always self.mode = #initialization
    until (PhysicalUnit.allInstances->forAll(pu | pu.ready)
      or self.wlmdFailure)
```

FIGURE 2.2.2: Exemple de LTL avec l'extension TOCL

Cette approche est fondée sur l'ajout de contraintes afin de définir le comportement d'objet.

Dans notre travail, ces aspects seront pris en charge par la spécification des relations (ALLEN et extensions) entre événements. Dans un premier temps, il s'agit de spécifier les dates d'occurrence pour un événement donné, l'étude des relations mutuelles sera détaillée par la suite.

2.2.2 Evénements composites

Comme nous l'avons déjà évoqué, notre étude concerne des événements complexes potentiellement considérés à différents niveaux d'abstraction. Un festival est un événement.

Mais cet événement n'existe que par l'existence et l'occurrence de sous événements qui le constituent : des spectacles, des concerts, des activités de différents types, etc. Un concert lui-même existe seulement par le fait que différents artistes se produiront sur une ou plusieurs scènes, etc. Il convient de garder cette structure agrégative hiérarchique active dans les modèles.

On comprend que, dans cet esprit, l'événement se doit de faire partie des éléments de modélisation de premier rang, et ne peut seulement être vu comme un artefact paramétrant par exemple une machine à état.

Nous traiterons successivement des manières de promouvoir ainsi les événements en avant de la modélisation, puis ensuite des possibilités offertes par les modèles et langages existants pour structurer ces éléments promus.

2.2.2.1 Les événements : éléments de premier rang dans la modélisation

Ce point est souligné par Olivé [Olivé 06] et les auteurs proposent un profil UML muni d'un stéréotype nommé « event » défini sur l'élément `UMLClass` du métamodèle UML.

Les `tagValues` permettent d'attacher des propriétés temporelles aux classes ainsi stéréotypées. En utilisant le stéréotype sur les éléments d'une hiérarchie de classes (spécialisation / agrégation), il est théoriquement possible de définir des événements complexes.

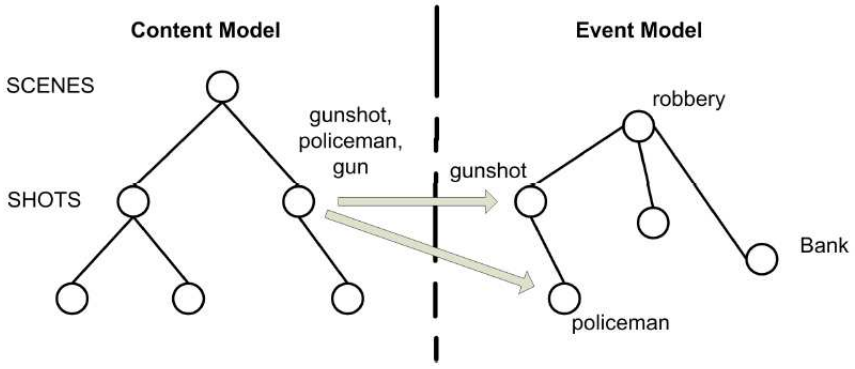
En pratique le problème n'est pas simple, et la sémantique des attributs temporels définis sur une classe et ses spécialisations/composants n'est pas triviale. Nous aborderons cette question en profondeur lors de la présentation de notre métamodèle d'événements (cf. chapitre 3).

Dans [Noyrit 12], les auteurs rappellent les avantages et inconvénients de l'utilisation des profils UML à la place d'un DSL et inversement. Dans notre cas, le langage envisagé aura une structure complexe et nous préférons en conséquence adopter l'approche DSL plutôt que de définir des stéréotypes. De plus, lors d'une transformation de modèles l'usage des stéréotypes s'avère moins approprié que celui des métamodèles.

De façon plus spécifique, un modèle d'événement dédié au contexte utilisateur (les médias) a été défini dans le cadre du projet Européen Glocal [Consortium 10]. Ce modèle prend en charge les relations topologiques entre événements et utilise les travaux d'Allen. La causalité entre événements est également considérée. Par contre, la répétitivité n'est pas abordée.

La modélisation est structurée en deux paquetages : l'un pour définir le domaine métier, l'autre pour spécifier les événements. Des liens sont ensuite tissés pour faire correspondre les événements avec les éléments du domaine métier. La figure 2.2.3 donne un exemple d'utilisation de l'approche de Glocal.

Cette approche nous intéresse tout particulièrement dans la mesure où notre propre démarche sépare les préoccupations de la même manière.



(source [Consortium 10])

FIGURE 2.2.3: Exemple de modèle Glocal permettant de joindre modèle de contenu et d'événement

2.2.2.2 Les événements, éléments structurés

La communauté de l'ingénierie des connaissances a développé de nombreuses ontologies qui traitent de la modélisation d'événements. On trouve notamment l'ontologie Event [1] qui propose une relation de sous-événement et fait référence à OWL-Time (cf. section 2.3.2.1) pour la partie temporelle et à l'ontologie de géo-positionnement WGS84 pour la partie spatiale [2].

D'autres travaux comme ceux fondés plus ou moins directement sur NewsML G2 ont abouti eux aussi à la construction d'une ontologie [Troncy 08].

Le projet Linking Open Data (LOD [3]) a également suscité la création d'ontologies adaptées à ce contexte. En ce sens, [Scherp 09] propose une ontologie dite « formalisée » nommée « Event Model F » qui cherche à répondre aux besoins du LOD pour la gestion des événements à caractère spatio-temporel.

Les différents types de relation entre événements ont été étudiés en profondeur dans cet article, ce qui a conduit notamment au modèle de la figure 2.2.4 qui propose une relation de parenté et partition (« *mereologic* ») entre événements avec l'utilisation d'une adaptation du patron de conception composant-composite.

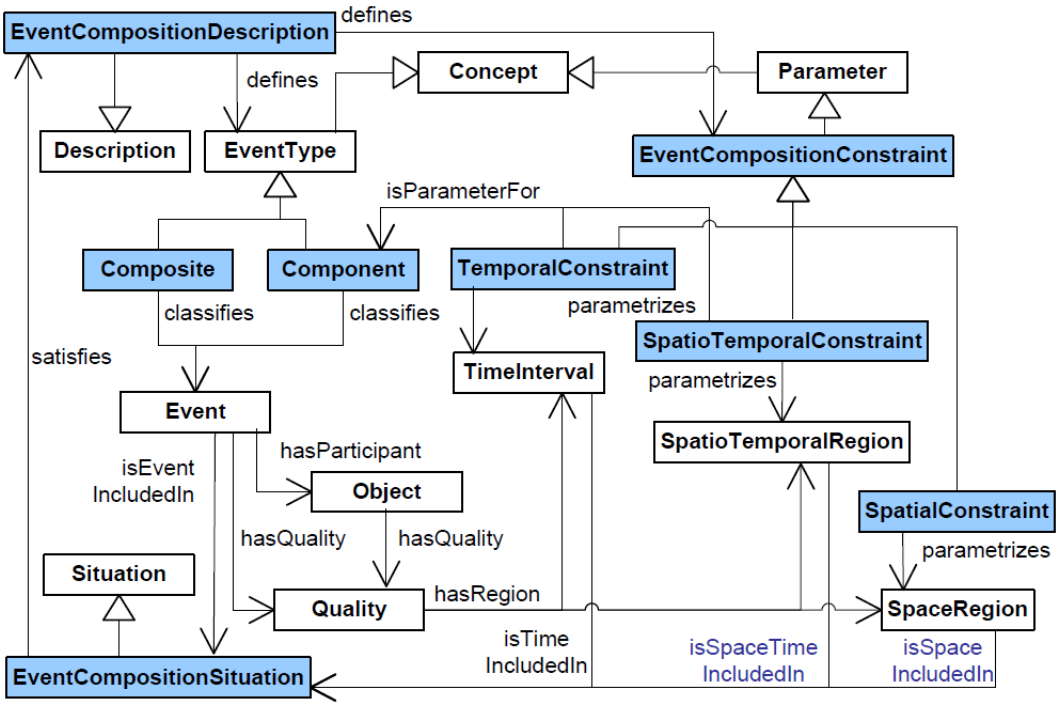
On y trouve également un comparatif des ontologies existantes suivant des critères tels que ceux figurant dans le tableau 2.1.

Les résultats de ce comparatif sont confirmés et complétés par d'autres critères dans [Consortium 10].

Cette dernière ontologie comporte également un aspect temporel qui prend en charge la définition de dates en extension et la capacité de définir des positions relatives entre événements en utilisant les relations d'ALLEN. En revanche, « Event Model F » ne pro-

1. `http://motools.sourceforge.net/event/event.html`
2. `http://www.w3.org/2003/01/geo/wgs84_pos`
3. `http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`

pose pas l'utilisation de propriétés temporelles en intension qui sont pour nous un point essentiel.



(source [Scherp 09])

FIGURE 2.2.4: La relation de parenté entre événements selon SCHERP

	Partici-pation	Time		Space		Relations			Documen-tation	Interpre-tation
		Rel.	Abs.	Rel.	Abs.	Mereo-logic	Causal	Corre-lation		
Eventory	Yes	Yes	Yes	Yes	Yes	Lim.	Lim.	No	Yes	No
Event Ontology	Yes	Yes	Yes	No	Yes	Lim.	Lim.	No	No	No
SsVM	Yes	Yes	Yes	Yes	Yes	Yes	Lim.	No	Yes	No
VERL	Yes	Yes	Yes	Yes	Yes	Lim.	Lim.	No	Yes	No
CIDOC CRM	Yes	Yes	Yes	Yes	Yes	Lim.	Lim.	No	Yes	No
EventML	Yes	No	Yes	No	Yes	Yes	No	No	Yes	Lim.
Event Calculus	No	Yes	Yes	No	No	Yes	Yes	Yes	No	No
Event Model E	Yes	Yes	Yes	Yes	Yes	Lim.	Lim.	No	Yes	Lim.
Event Model F	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Abbreviations: Rel.=relative, Abs.=absolute, Lim.=Limited

(source [Scherp 09])

TABLEAU 2.1: Comparatif des modèles d'événements sous forme d'ontologies selon SCHERP

En marge de l'ontologie « Event Model F » et dans le contexte du même projet (Event-

Media), [Shaw 09] propose des axiomes pour faire interopérer des ontologies composées d'événements : LODE (Linking Open Description of Events [4]). La mise en correspondance s'effectue sur les propriétés les plus couramment utilisées pour décrire des événements telles que les quatre « W » (cf. section 1.1). Ces travaux ont été notamment expérimentés pour extraire et mettre en relation des événements issus des chronologies de Wikipedia [5]. Dans ce même article, un tableau donne différentes définitions de la notion d'événements pour des ontologies.

2.3 Événements répétitifs

Après avoir décrit les travaux liés à la modélisation des événements et du temps, nous nous intéressons aux contributions antérieures traitant des événements considérés sous l'angle de la périodicité de leurs occurrences. En effet, caractériser une occurrence unique reste relativement simple, et nous centrons notre intérêt sur les événements répétitifs qui s'avèrent de fait souvent périodiques.

La section 2.3.1 présente des principes énoncés par différents auteurs pour classifier et représenter la répétition des occurrences d'événements.

Dans la section suivante (2.3.2), nous réalisons un inventaire des systèmes qui normalisent ou mettent en œuvre la gestion des événements répétitifs, en nous intéressant particulièrement aux communautés liées aux médias et à l'information journalistique.

Enfin, en section 2.3.3, nous traitons des calendriers qui constituent la référence pour situer les occurrences d'événements dans le temps.

2.3.1 Spécification de la répétitivité des occurrences

2.3.1.1 Différentes formes d'expression de la répétitivité

Pour structurer son analyse du sujet, Terenziani [Terenziani 00] donne une typologie des événements répétitifs.

Il distingue d'abord deux types principaux : les événements périodiques et les apériodiques. Les occurrences d'événements périodiques se répètent à intervalles réguliers, sur une période donnée. La périodicité étant donnée, il est alors possible de calculer précisément l'ensemble des dates d'occurrence (e.g. : « *tous les mardis de 10h à 12h* »), dès lors qu'une occurrence particulière est connue.

Les événements apériodiques se répètent mais la durée séparant deux occurrences successives est variable. Parmi les événements apériodiques, toujours dans [Terenziani 00], les notions de pseudo périodique et d'intermittent sont différenciées.

Pour pseudo périodique (e.g. : « *1 fois par semaine* »), on connaît la fréquence d'occurrence mais on ignore les dates précises d'occurrence. Il est à noter qu'une impréci-

4. `http://linkedevents.org/ontology/`
5. `http://en.wikipedia.org/wiki/Timeline`

sion résiduelle pourrait être levée en précisant si la fréquence affichée est une fréquence moyenne ou bien s'il s'agit d'une contrainte plus forte exigeant exactement une occurrence par unité calendaire. En toute rigueur, « *une fois par semaine calendaire* » n'est pas équivalent à « *une fois tous les sept jours* ».

Pour distinguer l'un de l'autre, il conviendra de considérer séparément le cas où la période de référence d'un événement pseudo périodique est spécifiée par une donnée quantitative de type durée (`duration`) ou par un élément calendaire précis.

La catégorie intermittent est encore plus imprécise, et indique simplement le caractère répétitif (e.g. : « régulièrement Tom va courir ») sans autre indication.

Ces nuances et ces typologies sont importantes en traitement du langage naturel, où l'on doit identifier et annoter des textes. Le rapport OGRE [Becher 06] (cf. section 2.3.3) préconise le même type de distinctions que celles énoncées par Terenziani en se fondant sur l'analyse de corpus de textes écrits en langage naturel.

Nous avons déjà évoqué les deux manières de spécifier la périodicité : donner la suite des dates d'occurrences (extension) ou bien spécifier une opération permettant de générer cette suite (intension) [Carnap 47]. En ce qui concerne l'extension, la question est d'une certaine manière réglée par l'adoption d'une norme (ISO 8601) pour représenter les dates concrètes dans un référentiel donné.

Pour ce qui est de l'intension plusieurs stratégies sont possibles. On peut fixer une date concrète de départ puis fournir un algorithme de calcul qui permet de déduire itérativement la date suivante à partir des dates précédentes.

On peut également donner un algorithme qui s'appuie sur les éléments calendaires (eux-mêmes répétitifs) et construit une date résultat. En appliquant l'algorithme à l'ensemble des occurrences des éléments calendaires considérés, on obtient la suite des dates d'occurrences cherchées.

Ce qui vient d'être dit pour des dates s'applique naturellement également pour des intervalles convexes.

Dans cette thèse, nous privilégions l'expression en intension pour deux raisons majeures. D'une part, contrairement au codage en extension, l'intension exprime la sémantique de la périodicité. D'autre part, la spécification en intension est plus économique en terme de gestion de la persistance et potentiellement en terme d'interrogation de bases de données (qui deviennent des pseudo bases de connaissance).

La spécification en intension n'est naturellement pas limitée aux phénomènes répétitifs. Les événements ayant une occurrence unique sont pris en charge de façon cohérente : simplement, la règle spécifiant la périodicité ou l'algorithme correspondant fournit retourne un singleton.

Ainsi lorsqu'un utilisateur spécifie une date en extension avec notre langage (cf. chapitre 3), elle sera convertie et stockée en intension par exemple le *« 15-03-2011* » devient le *« 15^e jour du 3^e mois de la 11^e année du 21^e siècle* »).

Dans le domaine de l'ingénierie des connaissances, Pan et Hobbs ont défini l'ontologie OWL-Time [W3C 06] pour exprimer des dates en extension et en intension. Notre approche est conforme à celle de OWL-Time qui propose deux formes, un type de donnée

XML et une structure *ad hoc* `DateTimeDescription`, pour stocker une date et privilégie la seconde sur les mêmes arguments que ceux que nous venons d'évoquer.

```
:meetingStart
  a :Instant ;
   :inDateTime :meetingStartDescription ;
   :inXSDDateTime 2006-01-01T10:30:00-5:00 .

:meetingStartDescription
  a :DateTimeDescription ; :unitType :unitMinute ;
   :minute 30 ; :hour 10 ; :day 1 ;
   :dayOfWeek :Sunday ; :dayOfYear 1 ; :week 1 ; :month 1 ;
   :timeZone tz-us:EST ; :year 2006 .
```

(source [W3C 06])

FIGURE 2.3.1: Différence entre `DateTimeDescription` et le type de données XML `dateTime`

La figure 2.3.1 illustre la différence entre l'usage de `DateTimeDescription` et le type de données XML `dateTime` (source [6]).

Avec `DateTimeDescription`, la date est explicitement décomposée suivant les unités temporelles identifiées, alors qu'avec `inXSDDateTime` elle se présente sous une forme textuelle qui masque la sémantique et nécessite un interpréteur pour reconnaître les unités calendaires référencées. En outre, aucun traitement simple ne peut extraire une information pertinente sur la périodicité d'un ensemble d'occurrences décrites sous forme textuelle selon `dateTime`.

Enfin, une approche connexe doit être évoquée, qui permet de définir *a priori* un système d'horloges sur lesquelles les événements d'intérêt sont synchronisés comme UPPAAL [7], LUSTRE [Caspi 87] et MARTE (cf. section 2.3.4.2). Il s'agit d'une description formelle abstraite qui décrit la logique temporelle d'un système et qui peut, dans des applications concrètes, s'instancier en identifiant plus ou moins directement les horloges avec des éléments calendaires.

2.3.1.2 Occurrences et instances

Il est nécessaire de s'arrêter sur la distinction entre événement et occurrence pour appréhender correctement la suite du manuscrit.

D'un coté nous avons un événement, auquel il est possible d'ajouter une règle de périodicité : « *les Francofolies ont lieu tous les ans en juillet* » et d'un autre coté il y a les éditions de cet événement que nous nommons « occurrences ».

En effet, nous considérons qu'un événement (« *Les Francofolies* ») qui possède une périodicité a des occurrences (« *Francofolies 2011* », « *Francofolies 2012* », etc), dont

6. `http://www.w3.org/TR/owl-time/#calclock`
7. `http://www.uppaal.org/`

les dates ancrées dans le calendrier sont conformes à la règle périodique spécifiée sur « *Les Francofolies* ». Chaque occurrence des « *Francofolies* » est considérée comme un événement à part entière, ainsi « *Francofolies 2012* » est un événement. Les « *Francofolies 2012* » ont lieu « *tous les jours du 11 au 15 juillet 2012* », par conséquent « *Francofolies 2012* » porte également une règle de périodicité. La profondeur de cette décomposition temporelle n'a pas de taille prédéfinie. Ceci montre bien les deux facettes que peut avoir un même événement, i.e. à la fois événement et occurrence.

Selon ce qui vient d'être énoncé, nous rappelons la définition d'un événement donnée par l'IPTC : « *toute chose qui se déroule en un lieu à une date* ».

Dans la suite de ce manuscrit nous parlerons d'événement et d'occurrence sachant qu'une occurrence est un événement particulier, issu d'un événement plus abstrait sur lequel sont définies des dates d'occurrence *via* une règle de périodicité.

À noter que plusieurs travaux dont ceux de Terenziani et le langage TimeML [Pustejovsky 10] utilisent le terme « instance » pour parler de l'occurrence d'un événement. Pour nous, une occurrence n'est absolument pas une instance d'événement au sens d'UML, la notion d'occurrence peut être considérée comme un sous-type d'événement.

2.3.2 Environnements gérant la répétition d'événements

Il existe de nombreux systèmes permettant de gérer les informations temporelles concernant des événements répétitifs. Nous présentons ci-dessous ceux qui nous semblent les plus représentatifs. La plupart intègrent à la fois la représentation de la périodicité et celle des événements concernés. Certains sont généralistes (OWL-Time, TIME-ML,) d'autres sont dédiés à un domaine métier particulier (NewsML, EventsML) ou encore spécialisés dans un type d'application : iCalendar pour la gestion d'agenda ou MARTE pour le raisonnement temporel, la vérification et la validation de systèmes temps réel.

2.3.2.1 Ontologie du temps : OWL-Time

Comme nous venons de le voir, l'ontologie OWL-Time [W3C 06] permet d'exprimer des dates en extension et en intension conformément à OWL [8].

Selon OWL-Time, les dates en intension s'expriment en utilisant le concept de `TemporalAggregate` qui est défini dans [Pan 05]. Un `TemporalAggregate` est un instant périodique tel qu'on peut le trouver dans MARTE (cf. section 2.3.4.2). La figure 2.3.2 en donne un exemple concernant le début de l'heure d'été aux Etats-Unis : « *le 1er dimanche de chaque mois d'avril* » (« *the US daylight saving starts on the first Sunday of every April* »).

OWL-Time permet de spécifier des durées entre instants et d'ajouter des relations qualitatives entre des instants et des périodes en réutilisant les résultats d'Allen. En revanche, cette ontologie ne permet pas de spécifier des intervalles périodiques ni des exceptions.

8. `http://www.w3.org/TR/owl-features/`

```
:tseq
  a :TemporalSeq ;
    :hasTemporalAggregateDescription :firstSunEveryApril .
:tseq-everyApril
  a :TemporalSeq ;
    :hasTemporalAggregateDescription :everyApril .
:everyApril
  a :TemporalAggregateDescription ;
    :hasTemporalUnit :unitMonth ;
    :hasithTemporalUnit 4 .
:firstSunEveryApril
  a :TemporalAggregateDescription ;
    :hasContextTemporalSeq :tseq-everyApril ;
    :hasContextTemporalUnit :unitMonth ;
    :hasithTemporalUnit 7 ;
    :hasTemporalUnit :unitDay ;
    :hasPosition 1 .
```

(source [W3C 06])

FIGURE 2.3.2: Instant périodique en OWL-Time

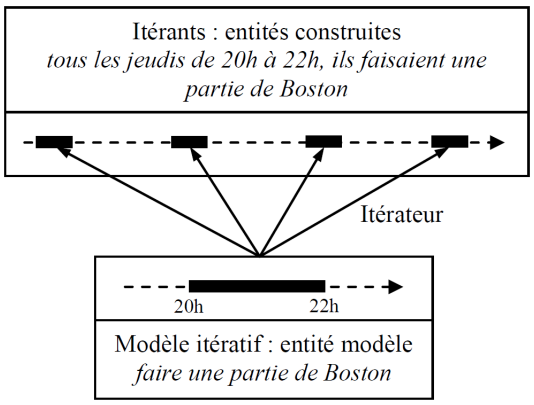
Il existe une forte convergence entre la spécification des ressources OWL-Time et les points de vue adoptés dans la norme ISO ainsi que dans nos propres propositions qui se présentent comme une intégration et une extension des concepts de bases reconnus et partagés par la communauté, dans le domaine du traitement de l'information temporelle et dans celui de la représentation des événements complexes.

2.3.2.2 Linguistique et TAL

Une part de notre travail traite des annotations de textes en langage naturel qui expriment des règles de périodicité. Nous donnons ci-après un ensemble de références sur le concept d'itération dans le domaine de la linguistique et du TAL.

Moteur d'itération (projet OGRE [Becher 06])

MATHET [Becher 06] et LEBRANCHU [Lebranchu 11] proposent un moteur d'itération (cf. figure 2.3.3) qui définit et exploite les notions de « Modèle itératif », « d'Itérateur » et « d'Itérants ». Le modèle itératif est l'événement référent qui doit être reproduit à l'aide de l'itérateur qui produit les itérants. L'itérateur utilise une règle de périodicité pour produire les itérants aux bonnes dates et horaires. Les concepts de « modèle itératif » et « d'itérant », qui ressortent de cette étude, correspondent pour nous respectivement à « événement » et « occurrence ».



(adapté de [Becher 06])

FIGURE 2.3.3: Modèle d'itération du projet OGRE

Cette contribution s'accompagne également d'une catégorisation du type des itérations qui se trouve dans des textes : « Itérateur Par Intervalles », « Calendaire Régulier », « Numéraire », « Fréquentiel », « Evénementiel », etc. Des itérateurs composés sont disponibles, ils permettent de spécifier deux niveaux de périodicité imbriqués.

La hiérarchie de types d'itérateurs fournie montre qu'il est nécessaire, comme nous le ferons, de définir un modèle de périodicité prenant en charge des expressions comme :

– des intervalles périodiques (*itérateur par intervalles*) ;
– avec des périodicités imbriquées, e.g. : « *15^e jour du 3^e mois* » (*itérateur composé*).

Modèle de période d'accessibilité (AP)

En section 1.2.4, nous avons défini la notion de période d'accessibilité (Access Period, AP). Celle-ci a été étudiée et développée autour de travaux en TAL comme [Battistelli 11, Teissèdre 10] qui font suite à [Weiser 08]. Ils consistent à analyser des fragments de textes contenant des expressions temporelles issues de sites web ou de dépêches d'information provenant d'agences de presse. Par conséquent ce sont des textes en langage naturel. Pour rappel, un modèle AP défini des horaires et/ou des dates d'occurrences d'événements en intension.

Les propriétés temporelles définies à l'aide d'un modèle AP concernent des événements qui se produisent une seule fois durant la période qu'elles décrivent. Ainsi, il n'est pas possible de décrire à l'aide d'un modèle AP une expression telle que : « ***2 fois*** *entre 10h et 12h* ». Dans ce cas, l'expression retenue sera « *de 10h à 12h* ». Il n'est pas non plus envisageable de définir des positions relatives entre événements.

Les analyses et traitements utilisés par TEISSÈDRE mettent en œuvre des techniques du TAL et permettent d'annoter des expressions temporelles en intension comme le montre la figure 2.3.4.

(source [Teissèdre 10, Mondeca 10])

FIGURE 2.3.4: Annotations pour l'expression « *du lundi au vendredi, de 12h à 14h et de 18h30 à 23h* » et « *le samedi, de 18h30 à 23h* »

Les résultats des analyses du système d'annotations sont uniquement des expressions en intension, ce qui rejoint notre postulat de départ. Cela englobe également les dates qui sont ancrées dans le calendrier (e.g. « *15/03/2011* »). Ce processus d'analyse et de production d'annotations temporelles a été implémenté dans l'application web TKA (Temporal Knowledge Acquisition [Mondeca 10]) à laquelle nous avons participé et sur laquelle nous reviendrons en section 7.5.

Notre approche est plus large que celle qui vient d'être décrite. En effet, au-delà des périodes d'accessibilité qui indiquent les propriétés temporelles individuelles d'un événement, nous souhaitons exprimer des relations structurelles et temporelles mutuelles entre deux événements.

2.3.2.3 ISO-TimeML : Annotation de textes en langage naturel

ISO-TimeML est issu de travaux de recherche qui ont constitué un standard ISO de la famille ISO/TC37 [Pustejovsky 03] (traite principalement de la segmentation de phrases, des annotations, etc). TimeML est un ensemble d'annotations qui permettent

d'identifier dans des textes en langage naturel les mots ou de groupe de mots décrivant l'aspect temporel d'une entité. Ces annotations sont conformes à un schéma XML, ainsi après annotation il est possible de les d'interpréter *via* un programme. TimeML propose des annotations pour :

– <EVENT> : identifie un verbe déclenchant un événement ;

– <MAKEINSTANCE> : relation entre un événement et ces « instances » (occurrences)

;

– <TIMEX3> : expression temporelle récurrente ou non ; la périodicité s'y exprime en terme de fréquence avec un intervalle de répétition, il n'est pas possible par exemple de spécifier une période d'accessibilité en intension en TimeML ;

– <TLINK> : relation temporelle qualitative entre événements (relations qualitatives d'Allen) ;

– <SLINK> : relation de subordination ;

– <ALINK> : relation aspectuelle entre événements, cette relation est à mi-chemin entre l'expression de la temporalité (<TLINK>) et la subordination (<SLINK>).

La figure 2.3.5 donne des exemples d'utilisation de ces annotations.

```
(6) twice a month
< TIMEX3 tid="t3" type="SET" value="P1M" freq="2X" >
twice a month
</TIMEX3>
(7) three days every month
< TIMEX3 tid="t4" type="SET" value="P1M" quant="EVERY" freq="3D" >
three days every month
</TIMEX3>
(8) daily
< TIMEX3 tid="t5" type="SET" value="P1D" quant="EVERY" >
daily
</TIMEX3>
(17) If Graham leaves today, he will not hear Sabine.
<SIGNAL sid="s1">
if
</SIGNAL>
Graham
< EVENT eid="e1" class="OCCURRENCE">
leaves
</EVENT>
< MAKEINSTANCE eiid="ei1" eventID="e1" pos="VERB" tense="PRESENT" aspect="NONE" polarity="POS"/>
<TIMEX3 tid="t1" type="DATE" value="XXXX-XX-XX" temporalFunction="true" >
today
</TIMEX3>
he will not
< EVENT eid="e2" class="OCCURRENCE">
hear
</EVENT>
< MAKEINSTANCE eiid="ei2" eventID="e2" pos="VERB" tense="FUTURE" aspect="NONE" polarity="NEG"
modality="WILL"/>
Sabine.
< SLINK eventInstanceID="ei1" subordinatedEventInstance="ei2" signaled="s1" relType="CONDITIONAL" />
< TLINK eventInstanceID="ei1" relatedToEventInstance="ei2" relType="BEFORE" />
(22) The search party stopped looking for the survivors.
The search party
< EVENT eid="e1" class="ASPECTUAL">
stopped
</EVENT>
< MAKEINSTANCE eiid="ei1" eventID="e1" pos="VERB" tense="PAST" aspect="NONE" polarity="POS"/>
< EVENT eid="e2" class="OCCURRENCE">
looking
</EVENT>
< MAKEINSTANCE eiid="ei2" eventID="e2" pos="VERB" tense="PRESPART" aspect="NONE" polarity="POS"/>
< ALINK eventInstanceID="ei1" relatedToEventInstance="ei2" relType="TERMINATES" />
for the survivors
```

(source http://www.timeml.org)

FIGURE 2.3.5: Exemples d'annotations TimeML

Chaque langue exprime de manière différente la temporalité, ainsi est-il nécessaire de définir des ensembles d'annotations TimeML pour chacune d'entre elles. Dans [Bittar 11], il est proposé une extension de TimeML pour le français qui est accompagnée d'une base de cas d'étude [9]. TimeML est dédié à l'annotation et n'a pas pour vocation de modéliser des événements ou des expressions temporelles exploitables par un SI. De plus,

9. https://gforge-qualif.inria.fr/projects/fr-timebank/

TimeML ne permet pas nativement d'exprimer des AP, ce qui limite considérablement nos possibilités de l'utiliser.

D'ailleurs, Teissèdre n'a pas réutilisé TimeML, mais créé ses propres annotations pour analyser des AP.

À noter que TimeML propose une modélisation intéressante et assez poussée des relations entre événements qui peuvent nous servir d'exemple.

2.3.2.4 NewsML G2 et EventsML G2 : Agences de presse et Medias

Dans le domaine de la presse, l'IPTC représente un consortium formé par les plus importantes agences de presse mondiales (AFP, Reuters, AP, etc). Son rôle est de développer des normes pour l'échange d'information telles que NewsML G2 et EventsML G2 [10]. NewsML se focalise sur les dépêches d'information alors qu'EventsML est spécialisée pour les événements.

EventsML est une partie de NewsML. NewsML définit des attributs permettant de décrire des dépêches d'information, alors qu'EventsML permet de décrire les événements cités dans la dépêche. EventsML permet notamment de définir des métadonnées concernant les quatre préoccupations du monde journalistique (i.e. : les quatre « W »). EventsML défini un schéma XML, qui permet le stockage dans des métadonnées concernant un événement et notamment de ses informations spatio-temporelles.

Nous discuterons successivement les aspects liés à la structure de événements et ceux dédiés à l'expression de la temporalité.

Structure des événements

Pour les aspects structurels, EventsML permet de spécifier des relations entre événements : parent (« *broader* »), enfant (« *narrower* ») et similaire (« *same as* »). Hormis ces relations structurantes, EventsML propose des relations sémantique (`item relation` [11]) entre les événements telles que : « *seeAlso* », « *dependsOn* », « *derivedFrom* », « *evolvedFrom* », etc.

EventsML permet de regrouper des événements (`KnowledgeItem`) selon différents critères, par exemple « *tous les événements qui se déroulent à La Rochelle* ».

Ces événements doivent avoir une propriété en commun, afin qu'il soit possible d'inférer les classes à partir de ce concept.

Dans [Pessemier 11], les auteurs recommandent l'utilisation d'EventsML G2 de préférence à iCalendar qui est clairement plus orienté vers la gestion d'agendas et de rendez-vous (cf. infra). Un comparatif est également réalisé avec FOAF [12] qui est jugé trop simpliste et limité. EventsML G2 est préféré en raison de sa richesse attributaire qui répond aux besoins des métiers des médias.

10. `http://www.iptc.org/site/News_Exchange_Formats/`
11. `http://cv.iptc.org/newscodes/itemrelation/`
12. `http://www.foaf-project.org/`

Spécification de la temporalité

La partie du schéma XML d'EventsML G2 consacrée à la spécification des aspects temporels est largement inspirée d'iCalendar (les correspondances avec ce standard sont explicitement fournies dans la spécification).

EventsML supporte la définition de propriétés temporelles à la fois en extension et en intension, mais dans la limite des capacités d'iCalendar et par conséquent il n'est pas possible d'établir des relations topologiques mutuelles entre intervalles non convexes.

EventsML répond à grand nombre de nos exigences tant selon le point de vue structurel et temporel. Cependant, l'expressivité d'EventsML reste insuffisante et l'absence de vue objet sur les concepts est limitative sur le plan applicatif (interopérabilité, raisonnement,...).

EventsML ne permet pas de connecter explicitement un événement à l'une de ses occurrences même s'il permet de spécifier une relation `isA` entre un concept et une de ses instances [13].

EventsML est dédié à l'échange et au stockage de données plutôt qu'à l'expression et au traitement de la sémantique.

2.3.2.5 Gestion d'agendas : iCalendar

iCalendar [Desruisseaux 09] est un standard IETF [14] (The Internet Engineering Task Force) utilisé dans de nombreuses applications à la fois professionnelles et personnelles. iCalendar est utilisé comme langage de spécification et format d'échange d'agendas que l'on retrouve notamment dans Outlook, Google Agenda, iCal, Sunbird, etc.

iCalendar permet de définir des événements géolocalisés que l'on peut considérer comme des « rendez-vous » auxquels des personnes peuvent participer. On peut définir des dates d'occurrence d'événements à la fois en extension et en intension.

Temporalité

La définition en extension précise une date et heure de début (`DTSTART`) et une date de fin (ou bien une durée).

Pour ce qui est des définitions en intension, iCalendar fournit une syntaxe permettant de spécifier des règles de périodicité (`RRULE`) basées sur les éléments nommés du calendrier grégorien. La figure 2.3.6 donne un exemple de règle de périodicité. La règle de récurrence `RRULE` (surlignée) définit que cet événement aura lieu tous les vendredi de 18h (`DTSTART`) à 19h (`DTEND`) et ce jusqu'au 29 septembre 2012 (`UNTIL`).

13. `http://cv.iptc.org/newscodes/conceptrelation/isA`
14. `http://www.ietf.org/`

Relations structurelles

La spécification de relations de parenté entre des événements est possible : *parent* (« *parent* »), *enfant* (« *child* ») et *frère ou sœur* (« *sibling* ») (cf. figure 2.3.6). Ceci permet de créer des hiérarchies d'événements qui peuvent être utiles pour gérer des questions de granularité.

```
BEGIN:VCALENDAR
BEGIN:VEVENT
CREATED:20120815T151627Z
LAST-MODIFIED:20120815T152141Z
DTSTAMP:20120815T151627Z
UID:6137c530-9e47-4d69-a3f7-ff7ed841b1b2
SUMMARY:Foot sur la plage
RRULE:FREQ=WEEKLY;UNTIL=20120928T200000Z;INTERVAL=1;BYDAY=FR
DTSTART;TZID=Europe/Paris:20120504T180000
DTEND;TZID=Europe/Paris:20120504T190000
LOCATION:Plage des Minimes
RELATED-TO;RELTYPE=PARENT:<20120815-151627-456789@foo.fr>
TRANSP:OPAQUE
X-MOZ-GENERATION:2
END:VEVENT
END:VCALENDAR
```

FIGURE 2.3.6: Exemple de règle de périodicité en iCalendar

Limitations

La syntaxe iCalendar est appropriée pour l'échange de données, mais non nécessairement adaptée à l'interaction avec un utilisateur final. iCalendar dans son ancienne version [Dawson 98] proposait des exceptions périodiques pour signifier par exemple qu'un lieu est *« ouvert tous les jours sauf le 1er mai de chaque année »*. Cette fonctionnalité est supprimée dans la version actuelle [Desruisseaux 09].

iCalendar ne permet pas de créer des règles de périodicité composées pour le même événement comme il serait nécessaire de le faire pour spécifier l'ouverture de la Promenade du Peyrou à Montpellier, i.e. : *« du 1er mars au 31 mai et du 1er juin au 31 août... »* (cf. annexe B.1.2).

Enfin, iCalendar ne prend pas en charge l'expression des positions relatives ni des relations topologiques entre événements.

Bibliothèques logicielles dédiées à iCalendar

iCal4j[15] est une bibliothèque écrite en JAVA qui permet de manipuler le langage iCalendar. iCal4j permet d'interpréter des textes en iCalendar pour manipuler sous forme d'objets des événements iCalendar. iCal4j supporte à la fois l'expression d'événements avec des dates de début et de fin ou bien des règles de périodicité. iCal4j supporte

15. http://wiki.modularity.net.au/ical4j/index.php?title=Main_Page

également les exceptions et plus particulièrement les exceptions périodiques même si celle-ci ont été retirées de la dernière version du standard [Desruisseaux 09]. Une méthode est fournie pour calculer des dates et périodes d'occurrence d'un événement à partir d'une règle de périodicité et d'une étendue temporelle (`getConsumedTime`). iCal4j est réutilisée par de nombreux outils.

PHPiCalendar[16] est un analyseur et un moyen de visualisation sous forme d'un calendrier de fichiers iCalendar dans un navigateur web. Cet outil est fondé sur la version 2009 de la spécification IETF [Desruisseaux 09]. Il inclut un moteur de recherche et propose de générer des flux RSS à partir de fichiers iCalendar.

De nombreuses extensions de CMS (Drupal, Joomla !, etc) ou bibliothèques (Zend, Yii, etc) proposent la gestion plus ou moins complète d'iCalendar.

2.3.3 Calendriers

Les calendriers jouent un rôle central dans la gestion des occurrences d'événements.

Nous évoquerons d'abord ce rôle et la manière de spécifier la sémantique des calendriers. Cette question est importante car il s'agit de réutiliser dans le cas général cette connaissance calendaire à des fins de raisonnement.

Nous évoquerons ensuite une autre question importante qui traverse toutes les réflexions sur la mesure et l'expression du temps, à savoir la gestion de la granularité.

2.3.3.1 Définition de la sémantique calendaire

En modélisation temporelle, le rôle des calendriers est double dans la mesure où d'une part, leurs composants périodiques (années, mois, jours...) servent d'ancrage à l'expression des périodicités des événements, et d'autre part, on peut utiliser un modèle d'événements et d'expressions temporelles pour exprimer la sémantique des calendriers.

Dans notre cas, on peut schématiquement décrire le processus comme suit. Le modèle d'événements composites est défini en premier lieu. Ce modèle est utilisé pour reconnaître les éléments calendaires comme des événements périodiques particuliers. Les relations d'Allen (étendues) permettent de spécifier la topologie qui représente une partie de la sémantique des calendriers. L'ajout de clauses OCL contraint les cardinalités et multiplicités des éléments calendaires. En dernier lieu, la sémantique calendaire ainsi définie peut servir à décrire de façon souple les périodicités d'occurrences d'événements quelconques. En tout état de cause, la sémantique des calendriers est également décrite dans les normes ISO construites à cet effet. L'intérêt du schéma précédent est d'intégrer dans une même approche de modélisation objet, les trois constituants que sont les événements, le langage d'expression temporelle (incluant la topologie d'Allen étendue) et les calendriers.

Ce processus ne nous est pas propre, et à la fois Leban, Ladkin et Terenziani ont utilisé les langages de spécification d'expressions temporelles pour décrire les calendriers.

16. `http://phpicalendar.net`

Terenziani [Terenziani 02] propose une étude exhaustive des langages formels constructifs dédiés à l'expression à la fois de la périodicité définie par un utilisateur et des contraintes temporelles sur des événements périodiques. En outre, [Leban 86] comme [Niezette 92], fournissent des grammaires et des opérateurs permettant, entre autres, d'exprimer la sémantique des calendriers. Grâce à cela, la sémantique du calendrier peut donner un apport conséquent pour faire correspondre le raisonnement qualitatif sur les relations entre des points ou des intervalles avec la connaissance géométrique et topologique concernant des événements ancrés dans le calendrier.

Dans [Leban 86], les auteurs proposent les concepts de « collections » pour définir des périodicités de base et deux opérateurs *dice* et *slice* qui s'appliquent sur ces collections. Ces opérateurs permettent de définir des périodicités personnalisées. De manière générale, le calendrier est considéré par Leban comme une séquence infinie d'intervalles positionnés sur une droite représentant le temps. En revanche le langage qu'il a développé ne prend pas directement en compte des périodes qui seraient définies par deux instants périodiques : un pour le début et un pour la fin. Prenons l'exemple d'une expression en utilisant le langage de Leban : « *La semaine du 15^e jour de chaque mois* » est codée par : « $the/Weeks.overlaps.15/Days : during : Months$ ».

[Niezette 92] reprend le concept de calendrier définit par l'utilisateur lui-même, plus précisément de périodicité basée sur des éléments calendaires, ceci en introduisant la notion « d'intervalles répétitifs linéaires ». Ils proposent notamment une syntaxe exprimant des périodicités comme par exemple : « $2012.Years + \{7, 8\}.Months \diamond 5.Days$ » qui représente les cinq premiers jours de chaque mois de juillet (mois n°7) et août (mois n°8) de l'année numérotée 2012.

De part ces différents travaux, nous pouvons en tirer que pour un humain, la spécification de la périodicité d'un événement est souvent associée au calendrier ce qui révèle l'importance de pouvoir accéder aux éléments du calendrier. Pour exprimer des périodes d'accessibilité tel que nous l'envisageons, les éléments calendaires sont également omniprésents.

Ces travaux montrent également l'intérêt porté à exprimer des périodicités avec des langages possédant une grammaire (formelle). En revanche ces langages ne sont pas orientés-objets, une des conséquences est l'impossibilité d'utiliser un langage comme OCL pour exprimer des contraintes non syntaxiques. En effet, il est nécessaire de vérifier par exemple que les expressions aient du sens, e.g. : $2012.Years+\{7, 13\}.Months\diamond 5.Days$ et $2012.Years + \{8, 7\}.Months \diamond 5.Days$ ne sont pas correctes.

La spécification du calendrier grégorien selon notre système, sera étudiée en détail au chapitre 5.

2.3.3.2 Gestion de la granularité

La granularité a été étudiée par Chittaro [Chittaro 02] dans un cadre général et une étude plus spécifique à la problématique des calendriers a été menée par Bettini [Bettini 04].

Les éléments calendaires sont des événements structurés. Les années sont constituées de mois, les mois de semaines, les semaines de jours, etc. La partie structurelle des modèles de calendrier rend compte de ces aspects.

Par contre, l'ambiguïté naît lorsque l'on se place sur un plan quantitatif, et qu'on utilise les éléments calendaires comme exprimant une notion de durée. En effet, l'expression « *quatre fois en janvier* » est strictement non ambiguë et précise.

Rien ne peut être déduit de façon certaine sur le nombre d'occurrences hebdomadaires. Cela est légitime car « *janvier* » représente un événement calendaire périodique et non une quelconque durée exprimée en nombre de semaines.

Un degré supplémentaire d'imprécision concerne les locutions telles que « *quatre fois par mois* ». Cette fois « *mois* » représente une unité calendaire qui permet d'exprimer des durées. La sémantique d'une telle expression est clairement statistique, il s'agit d'un taux moyen d'occurrence, qui doit être interprété comme tel.

Il n'y a pas d'équivalence sémantique entre la notion de mois de janvier et l'intervalle « *du 1er janvier au 31 janvier de chaque année* ». Cependant il est possible de définir des relations topologiques et ensemblistes mutuelles ce qui est une façon de gérer la granularité.

Une part de ces questions est réglée dans la spécification des calendriers eux mêmes qui distinguent précisément éléments et unités calendaires, dates et durées.

Néanmoins les difficultés subsistent lorsqu'on traite de raisonnement temporel et que l'on doit intégrer des assertions exprimées à différents niveaux de granularité.

2.3.4 Contraintes temporelles

La problématique que nous traitons peut être vu de deux façons équivalentes :

1. comme nous l'avons compris jusque là, nous souhaitons modéliser des informations sur les occurrences constatées ou prévues d'événements donnés ;
2. ou bien nous voulons imposer ou vérifier des contraintes sur les occurrences de ces mêmes événements.

La prépondérance de la vérification de comportement dans la discipline informatique a privilégié le point de vue « contrainte ».

Nous discutons brièvement ci-dessous de cette approche en citant des auteurs ayant construit une classification des contraintes temporelles, puis évoquons les outils de raisonnement qui peuvent être mis en œuvre pour en assurer la vérification.

Nous nous intéressons au profil UML MARTE, qui est représentatif en la matière, relève du paradigme objet, à côté de DSL issus des logiques temporelles, des automates de processus concurrents et des systèmes à horloge (UPPAAL, LUSTRE...).

2.3.4.1 Typologie des contraintes

Terenziani dans [Terenziani 00] propose de classifier des types de contraintes temporelles sur des occurrences d'événements. Il donne des exemples qui illustrent et dis-

tinguent trois catégories, respectivement étiquetées par : LOC, QUAL et LOC-QUAL. Notons que ces classes ont également été reconnues dans les expressions de la langue naturelle (cf. projet OGRE). Ainsi il distingue les notions de LOC, QUAL et LOC-QUAL :

- QUAL permet de spécifier une relation qualitative entre deux événements, e.g. : « *Student tests* *are* **always preceded by** *an electronic reservation* ».
- LOC vise à définir une fréquence ou encore un nombre de répétitions de l'événement lors d'une unité calendaire, e.g. : « **each Tuesday** *John went* **twice** *to the post office* ».
- LOC-QUAL met en œuvre les définitions précédentes en associant à la fois la fréquence et la dépendance temporelle servant à positionner de manière qualitative les événements : « **twice each Tuesday** *the class I_A has* **two units** *of Gym (strictly or not)* **before one unit** *of History* ».

On consultera également l'étude [Anselma 09] qui elle aussi inventorie les différentes manières d'exprimer des contraintes qualitatives ou quantitatives sur la périodicité d'occurrences d'événements.

Concernant les outils de raisonnement qualitatif sur le temps comme SPARQ [Wallgrün 06], GQR [Westphal 09] et QAT [Condotta 06]. Nous n'avons pas pu utiliser ces outils car ils supposent *a priori* les relations disjointes et possédant un symétrique. Il est possible d'établir un corpus de relations disjointes générant nos relations étendues, mais la notion de symétrie n'a pas de sens naturel pour celles-ci. En fait, nous nous attachons à travailler sur des types de relations et non sur des relations individuelles. Ceci empêche d'utiliser ces outils en l'état. En revanche, il est possible de s'inspirer des algorithmes qui y sont mis en œuvre et construire nos propres raisonneurs (par exemple en Prolog).

2.3.4.2 Vérification de contraintes temporelles : MARTE

Les communautés des systèmes temps-réels et embarqués et de l'IDM ont développé le profil UML MARTE [André 07, André 10] qui vise à ajouter à UML un modèle d'événement et de temps spécifique. Ce profil UML, étendant le package `SimpleTime` d'UML, s'attache à modéliser le temps logique.

Un utilisateur peut définir des relations de causalité et temporisées entre horloges. MARTE est couplé au langage de contraintes CCSL [Mallet 10] (Clock Constraint Specification Language).

MARTE et CCSL sont mis en œuvre à travers l'outil IDM TimeSquare [17] [DeAntoni 12b, Mallet 11] qui offre la possibilité de spécifier et d'évaluer les contraintes et propose des solutions suivant une spécification de contrainte d'horloge. Ainsi il est possible de définir des contraintes telles que « *une occurrence de A a lieu avant une occurrence de B* ». La figure 2.3.7 donne la syntaxe concrète en CCSL pour l'exemple précédent : relations entre A et B. Nous considérons ainsi que A est associé à l'horloge c1 et B à l'horloge $c2$. La ligne 12 de la figure définit la précédence de l'horloge $c1$ par rapport à l'horloge

17. `http://timesquare.inria.fr`

$c2$, par conséquent de A par rapport à B.

```
/** CCSL specification */
ClockConstraintSystem MySpec {
  imports {
    import "ccsl:kernel" as kernel ;
  }
  entryBlock main
  Block main {
    Clock c1
    Clock c2
    Relation r1[Precedes](LeftClock -> c1, RightClock -> c2 )
  }
}
```

FIGURE 2.3.7: Exemple de spécification de contraintes d'horloge avec TimeSquare

Un travail fondé sur MARTE, CCSL et TimeSquare vise à étendre OCL (i.e. ECL [Deantoni 12a]) pour spécifier des relations de causalité et temporisées dans des modèles. L'exemple donné dans [André 08] [18] : « *Pâques est le 1er dimanche après le 14^{e} jour du mois lunaire qui a lieu à partir du 21 mars* » (i.e. le jour de l'équinoxe de printemps), illustre la puissance du langage.

Les exemples que nous avons évoqués comme faisant partie de nos cas d'utilisation cibles « *3 heures avant la basse mer* » pourraient tout aussi bien s'exprimer en CCSL.

Un de nos objectifs dans cette thèse est de construire un métamodèle pivot qui puisse dans une forme proche du langage naturel capturer, enregistrer et restituer les informations fournies par un utilisateur ou issues de flux textuels structurés et annotés. Le raisonnement consiste à vérifier d'une part les aspects syntaxiques par rapport au métamodèle d'événements et d'expressions temporelles et d'autre part la vérification sémantique par rapport à la sémantique du calendrier. Un dernier aspect concerne la vérification de la cohérence des événements de la base dans leur ensemble.

Pour ce dernier type de raisonnement, il est préférable de déporter la charge de la vérification vers des environnements spécialisés comme MARTE.

Il convient donc de traduire (automatiquement) les propriétés des événements de notre modèle dans un format adapté à MARTE. Un tel traducteur opère *via* l'IDM sur les métamodèles eux-mêmes et gère les événements quels qu'ils soient. Il n'y a donc pas lieu de réécrire un traducteur suivant les cas traités mais bien de réutiliser le traducteur générique résultant de l'approche IDM.

Ce raisonnement vaut également pour d'autres raisonneurs.

18. Cette exemple est fourni pages 7 et 8 du rapport de recherche [André 08]

2.4 Conclusion

Dans ce chapitre nous avons réalisé un état de l'art concernant les modèles et langages dédiés à la représentation du temps et des événements composites. Nous avons évoqué les différents aspects géométrique, topologique et structurel de la spécification des éléments intervenant dans la modélisation du temps.

Pour les événements, il est indispensable de prendre en compte une structure hiérarchique, et de distinguer cette structure des liens qui unissent un événement et ses occurrences. Nous avons montré comment les communautés des linguistes, celle du traitement de la langue naturelle et enfin celle issue de l'informatique et de l'ingénierie des connaissances se rejoignaient pour structurer de façon cohérente les expressions temporelles d'une part et les contraintes temporelles d'autre part. Il apparaît clairement que les modélisations des événements des expressions temporelles et des calendriers sont intimement liées et que les synergies se situent aux interfaces de ces différents modèles. Nous avons indiqué que si notre modèle porte ses propres concepts et outils pour vérifier et valider les informations qui le peuplent, il est prévu de tisser des ponts vers des environnements spécialisés dans le raisonnement et les langages temporelles à base de contraintes (e.g. : MARTE).

Le tableau 2.2 compare les différents modèles étudiés précédemment, nous n'avons retenu que ceux qui présentent un concept d'événement et qui sont susceptibles de nous servir comme exemple à suivre. Les critères de comparaison distinguent les modèles possédant une relation de parenté, qui propose des relations sémantiques entre événements et une relation d'occurrence entre un événement et celui qui le définit. De plus, nous ajoutons des critères temporels qui s'appuient sur la capacité à exprimer l'occurrence d'un événement avec une règle de périodicité (en intension) et la possibilité de spécifier des positions relatives entre instants et intervalles convexes.

Ayant ainsi étudié la problématique qui nous intéresse sous un angle bibliographique, nous abordons dans les chapitres qui suivent la présentation des métamodèles que nous avons conçu et l'étude de leur mise en œuvre.

Modèle / Langage	Relation de parenté	Relation sémantique	Relation d'occurrence	Périodicité (expression en intension)	Relation en intension entre instants	Relation qualitative entre intervalles
Glocal	√	√	–	–	qualitatif	convexe
OGRE	–	–	√	multiple	qualitatif et quantitatif	convexe
TimeML	√	√	√	fréquentielle	qualitatif	convexe
iCalendar	√	–	√*	unique	–	–
EventsML G2	√	√	√*	unique	–	–
Event+OWL-Time	√	–	–	multiple	qualitatif	convexe
Event Model F	√	√	√	–	qualitatif	convexe
UML-MARTE	√	–	√	multiple	qualitatif et quantitatif	–

Relation en intension entre instants :
– qualitatif : « A *precedes* B »
– qualitatif et quantitatif : « A *precedes* B avec un délais de 3 heures »

* dans le cas d'une exception, la propriété RECURRENCE-ID peut être assimilée à une relation liant un événement à son occurrence

Tableau 2.2: Comparatif des modèles d'événements et de temps

Deuxième partie

Modélisation et relations temporelles pour des événements répétitifs et composites

« An interval-based temporal logic is introduced, together with a computationally effective reasoning algorithm based on constraint propagation. This system is notable in offering a dedicate balance between expressive power and the efficiency of its deductive engine. A notation of reference intervals is introduced which captures the temporal hierarchy implicit in many domains, and which can be used to precisely control the amount of deduction performed automatically by the system. »

(James F. ALLEN, "Maintaining Knowledge about Temporal Intervals", 1983)

Chapitre 3

Métamodélisation d'événements temporels composites

3.1 Introduction

Ce chapitre étudie la métamodélisation des événements en terme de structuration et d'expression de la temporalité en particulier de la récurrence.

Le métamodèle d'événement à définir doit servir de pivot entre plusieurs applications. En effet, l'objectif est de réunir en un seul modèle les concepts fondamentaux et partagés sur lesquels sont construites les propriétés structurelles et temporelles des événements considérés. L'accent est mis sur la représentation composite des événements complexes et sur la périodicité de leurs occurrences qui prévaut dans de nombreux cas pratiques.

Les applications cibles avec lesquelles le modèle devra être interfacé concernent la connexion avec différentes sources de données (pourvoyeurs de contenus, flux RSS...) à des fins de peuplement (semi) automatique du modèle. À l'autre extrémité de la chaîne, le modèle doit permettre de générer des bases de données ou de connaissance assurant la persistance des informations dans des formats variés en usage chez les utilisateurs et enfin, le modèle doit offrir des fonctionnalités d'interrogation et de traduction des données vers des applications adaptées à la diffusion, publication et visualisation à destination de l'utilisateur final.

Les aspects formels du modèle sont de nature à assurer également la correction des données manipulées et échangées : correction individuelle par rapport au modèle et cohérence mutuelle, horizontale entre événements frères, ou verticale entre événements *pères* et *fils*. Concernant des raisonnements temporels plus élaborés, il convient par exemple de traduire les concepts du modèle vers des langages spécialisés utilisant différentes logiques (LTL, LTL*) des algèbres de processus (CCS [Milner 82], CSP) ou des systèmes à horloges (LUSTRE, MARTE).

Nous avons comme objectif la gestion d'expressions temporelles exprimées en intension plutôt que de traiter des séquences de dates concrètes traduisant la suite des occurrences d'un événement. La série d'occurrences peut masquer des informations pertinentes sur la fréquence de répétition.

Notre travail s'appuie sur une modélisation objet et applique systématiquement une démarche d'Ingénierie Dirigée par les Modèles (IDM). Cette approche permet de gérer la complexité de façon efficace et organisée, d'assurer l'évolutivité et la réutilisabilité, de faciliter l'interopérabilité. L'objectif est comme nous l'avons dit, de proposer un métamodèle d'événements répétitifs avec un outillage élaboré en termes d'importation/exportation de données, de peuplement et de validation de ses instances.

Ce métamodèle, bien que supporté par une grammaire formelle, est fondé sur un vocabulaire et une syntaxe proches du langage naturel. L'existence de cette grammaire formelle facilite la construction de ponts vers les DSL des logiques temporelles citées plus haut. Dans la même perspective, nous avons tenu à rester compatibles avec les normes en vigueur dans le domaine de l'information spatio-temporelle (e.g. : ISO 19108 et ISO 8601:2004).

La section 3.2 qui suit présente et discute les différents aspects de l'IDM que nous

avons mis en œuvre. Avant de présenter le métamodèle PTOM (Periodic Temporal Object Model) nous donnons les besoins pour ce modèle (section 3.3) où nous comparons ceux-ci avec les propositions d'approches connexes. Nous détaillons ensuite le métamodèle PTOM (section 3.4) en insistant sur les apports majeurs et la grammaire formelle qui l'accompagne (section 3.5). La section 3.6 propose de relier des événements avec des classes et instances d'un modèle du domaine en UML.

3.2 L'Ingénierie Dirigée par les Modèles

L'Ingénierie Dirigée par les Modèles [Atkinson 03, Schmidt 06] est la méthodologie qui est à la base de toutes les spécifications et développements qui ont été réalisés durant cette thèse, cette section en présente les principes fondamentaux et les outils utilisés.

3.2.1 Principes

L'Ingénierie Dirigée par les Modèles est principalement utilisée pour générer ou manipuler tout ou partie d'un logiciel à partir de modèles. « Un modèle est utilisé pour abstraire la réalité, parfois complexe ». En IDM, l'objectif est d'aller au-delà de sa définition pour qu'il soit mis à profit d'un logiciel que ce soit *via* une génération de code ou une utilisation durant l'exécution [Jézéquel 12]. La modularité et la séparation des préoccupations métiers et logiciels sont le cœur des problématiques de l'IDM.

Modèle, métamodèle

Dans le vocabulaire de l'IDM, un système est donc représenté par un modèle. L'introduction du concept de métamodèle permet de définir des caractéristiques communes à un ensemble de modèles. Un métamodèle représente une spécification formelle d'une abstraction, généralement consensuelle et normative. Le concept de métamodèle permet également d'aider, pour un système donné, à la construction d'un modèle. Un modèle est lié à son métamodèle par une relation de conformité.

L'illustration de ces principes se retrouve dans l'architecture à quatre couches proposée par l'OMG dans son approche MDA (Model Driven Architecture) décrite dans [Miller 03]. Cette architecture (cf. figure 3.2.1) est constituée, au niveau le plus bas, par la couche M0 représentant le système réel. Un modèle représente ce système au niveau M1. Ce modèle se conforme à son métamodèle définit au niveau M2 et le métamodèle se conforme au méta-métamodèle au niveau M3. Le méta-métamodèle se conforme à lui-même, il est d'usage de parler « d'auto-définition » [Bézivin 05]. L'empilage des couches est ainsi stoppé ce qui a pour résultat, quel que soit le système, de limiter sa profondeur dans cet espace à quatre couches.

À partir de cette hiérarchisation des modèles, il est possible d'exprimer des relations mutuelles, elles-mêmes conformes à différents métamodèles, ceci se concrétise par des

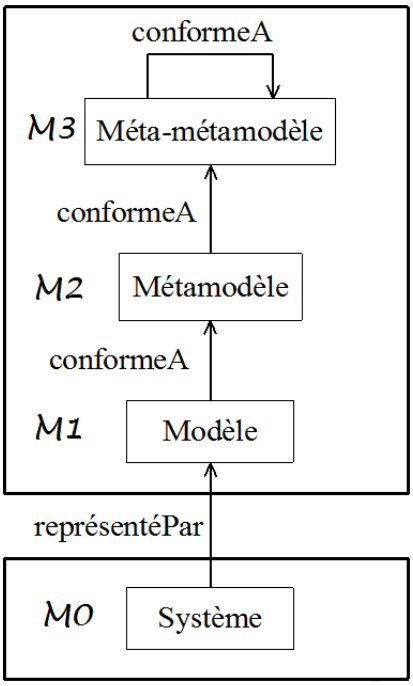


FIGURE 3.2.1: Architecture MDA en quatre couches

transformations de modèles. Dans l'approche MDA, l'objectif est de générer des modèles spécifiques à une plate-forme (PSM, Platform-Specific Model) à partir de modèles indépendants de toute plate-forme (PIM, Platform-Independent Model). Le langage QVT (Query, View, Transform) [OMG 11a] a été spécifié pour décrire des transformations et ainsi répondre aux besoins du MDA.

Le choix, dans l'approche MDA, de la recommandation MOF (Meta Object Facility) [OMG 11b] comme méta-métamodèle au niveau M3, permet l'utilisation d'un ensemble de métamodèles standardisés tels qu'UML [OMG 12b], CWM [OMG 03], SPEM [OMG 08], etc. Il permet également de définir des projections entre l'espace technique MDA et d'autres espaces techniques comme celui d'XML, par exemple le standard XMI (XML Metadata Interchange) [OMG 11c] utilisé pour l'échange de modèles.

3.2.2 Transformation de modèles

Comme il a été dit précédemment, les transformations de modèles sont un aspect important de l'Ingénierie Dirigée par les Modèles. De manière générale, la transformation de modèles utilise un ensemble de programmes prenant en entrée des modèles (accompagnés de leur métamodèle) et produisant en sortie d'autres modèles, comme illustré par la figure 3.2.2.

Il existe de nombreux outils permettant de définir des transformations. Ici nous distinguons deux types d'outils qui permettent de disposer pleinement de langages efficaces

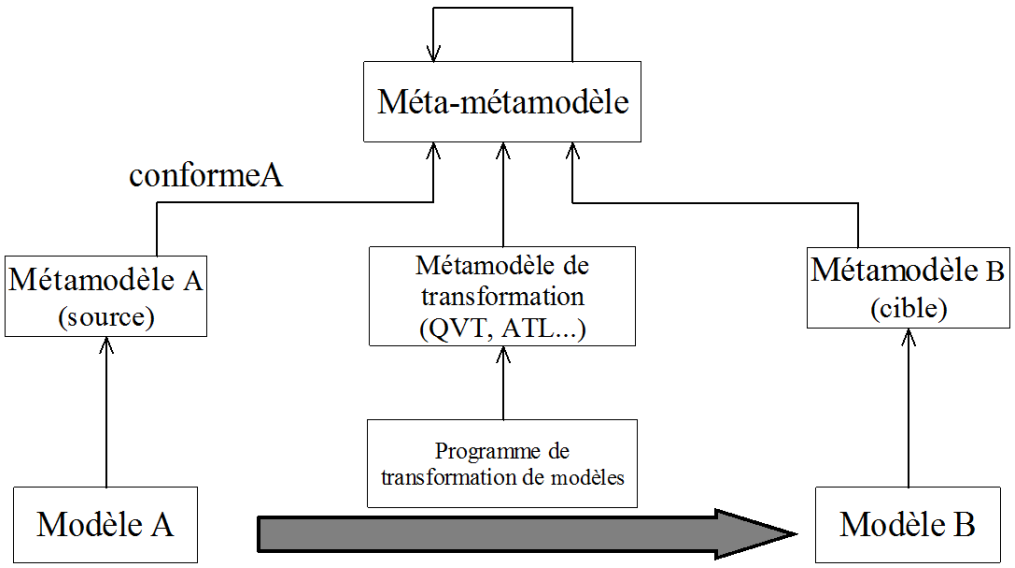


FIGURE 3.2.2: La transformation de modèles

pour réaliser des transformations, car leur syntaxe est orientée modèle :

1. les outils dédiés spécifiquement à la transformation de modèles et conçus pour être intégrables dans des environnements de développement largement diffusés comme Eclipse. Comme exemples, nous pouvons citer le langage ATL (Atlas Transformation Language) [Bézivin 03] et QVT [OMG 11a] ;
2. les outils de métamodélisation dans lesquels la transformation de modèles correspond à l'exécution d'un méta-programme. Des outils comme KERMETA [Muller 05] et Epsilon [1] manipule les modèles et métamodèles d'entrées et de sorties grâce à la réflexivité du langage. La transformation est implémentée en ajoutant aux parties structurelles des comportements grâce à un langage d'action. KERMETA (version 1.3) possède un générateur de code source JAVA qui transforme un programme KERMETA en JAVA-EMF.

3.2.3 Langage spécifique à un domaine

Définition

Un langage spécifique à un domaine (Domain Specific Language, DSL [Consel 98, van Deursen 00]) est un langage dédié à un domaine ou à un problème contrairement à un langage généraliste (General Purpose Language, GPL). Celui-ci peut se définir en utilisant un métamodèle. Un DSL doit être outillé pour aller au-delà de sa simple définition, il est utile de définir des syntaxes pour définir des instances de ce DSL. Rendre opérationnel un DSL peut être réalisé de deux manières : soit le concepteur du DSL ajoute

1. `http://www.eclipse.org/epsilon/`

dans le langage les constructions nécessaires pour pouvoir exécuter les modèles, soit les modèles sont traduits vers un autre langage qui sera chargé de l'exécution (principe de la compilation).

Syntaxe abstraite

Une syntaxe abstraite peut être associée à un métamodèle. Celle-ci est une syntaxe « par défaut » dont la structure est arborescente à l'image de la structure objet d'un modèle. Une syntaxe abstraite possède une racine et les différentes branches de cette syntaxe sont orientées par les références de type « composite ». Par exemple si une classe A possède une référence de type « composite » vers une classe B, alors nécessairement une instance de B sera contenue par une instance de A. Au chapitre 6, nous utiliserons le terme de « conteneur » pour caractériser un élément similaire à une instance de A.

Syntaxe concrète

Un des enjeux sous-jacents à de l'IDM est d'outiller les DSL que les modélisateurs ont conçus, notamment afin de proposer des syntaxes concrètes, qu'elles soient textuelles ou graphiques.

On peut noter les travaux et outils autour du projet Topcased [2] et de Graphical Modeling Project (GMP [3]) qui permettent de générer des éditeurs graphiques de modèles.

Concernant les aspects textuels, des outils proposent d'utiliser une grammaire pour instancier un métamodèle. Ainsi de nombreux outils ont vu le jour afin d'offrir un analyseur et un générateur de textes à partir d'une grammaire de type EBNF (Extended BNF [ISO/IEC 96]) qui s'appuie sur les concepts d'un métamodèle. Par exemple EMFText [4], TCS [5], Xtext [6] [Eysholdt 10] génèrent du code source implémentant un analyseur et un générateur de textes alors que Sintaks [Muller 08] propose un interpréteur pour ces deux composants. Xtext a été promu composant Eclipse et fournit des facilités pour associer à la grammaire des contraintes du type OCL (fichier de type ETL). Ceci est très utile pour intégrer une validation des modèles manipulés directement *via* un éditeur de texte.

Outre la possibilité de manipuler un modèle de manière textuelle, les grammaires permettent de filtrer une partie d'un métamodèle. En effet, seuls les éléments souhaités par le créateur de la grammaire sont rendus accessibles vis-à-vis de l'utilisateur final. Ceci permet notamment de définir différentes versions de grammaires qui couvrent plus ou moins un métamodèle.

2. `http://www.topcased.org/`
3. `http://www.eclipse.org/modeling/gmp/`
4. `http://www.reuseware.org/index.php/EMFText_CS_Language_Reference`
5. `http://www.eclipse.org/gmt/tcs/`
6. `http://www.eclipse.org/Xtext/`

3.2.4 Composition de modèles

Principes

La composition de modèles a pour objectif d'unifier deux modèles distincts en apportant leurs concepts respectifs. Elle est principalement de deux types comme le décrivent les auteurs de [Marchand 12] en faisant référence aux opérateurs de fusion (*merging*) et de tissage (*weaving*). [Blay-Fornarino 09] distingue la composition par surimposition de la composition par quantification [Apel 07]. La surimposition s'apparente à la fusion, alors que la quantification fait référence au tissage. Dans la suite nous parlerons de *fusion* et de *tissage*.

La fusion consiste dans un premier temps à identifier les éléments communs puis à fusionner les modèles. Cette identification est généralement réalisée par « pattern matching » en utilisant par exemple un libellé, un type ou des signatures [Reddy 05].

Pour le tissage, les éléments à composer sont déclarés par l'utilisateur en utilisant un langage *ad hoc* ou générique [Morin 08] notamment par des points de coupe.

Pour ce qui est de notre utilisation de la composition de modèles, nous souhaitons fusionner des modèles ainsi le tissage ne sera pas utilisé, nous décrivons ci-après l'outil choisi.

Composition par fusion

Kompose est un outil de fusion de modèles [France 07] qui est implémenté avec le langage de métamodélisation Kermeta. Le principe de Kompose est de fusionner deux modèles homogènes, i.e. conforme au même métamodèle en comparant les signatures des éléments. Il est possible de proposer sa propre implémentation pour calculer l'égalité de signatures. Les éléments identifiés avec des signatures identiques sont fusionnés alors que tous les élements sous-jacents sont quant à eux copiés dans le modèle résultant. Kompose propose également un système de pre- et post- directives pour effectuer des opérations sur les modèles avant la phase d'identification (pre) et après la fin de la fusion (post).

3.2.5 Manipulation et persistance des modèles

Outre les outils de métamodélisation et de manipulation de modèles précédemment cités, le projet Eclipse EMF (Eclipse Modeling Framework [7]) fournit des outils pour construire des métamodèles avec le langage Ecore. EMF est actuellement l'outil de métamodélisation le plus répandu dans la communauté. Les métamodèles sont instanciés pour créer des modèles, ils sont alors décrits et sérialisés à l'aide d'une syntaxe abstraite. La sérialisation des modèles EMF utilise le format XMI.

7. `http://www.eclipse.org/modeling/EMF/`

Afin de manipuler les modèles *via* un programme, ici en Java, EMF permet de générer une API Java image d'un métamodèle Ecore. Cette génération est configurable *via* un modèle de type `genmodel` pour notamment choisir les patrons de génération. Le résultat de la génération se présente sous forme de trois modules additionnels (*plugins*) Eclipse qui respectent le patron de conception MVC (Model-View-Controller). Ainsi à partir du module « model », il est possible de créer, charger, modifier et sauvegarder des modèles.

3.2.6 Génération de textes à partir de modèles

La génération de textes est un aspect de l'IDM qui permet par exemple de générer de la documentation ou bien du code source (Java, HTML, etc). Ici nous présentons les outils Jet et Ket, ce dernier sera utilisé en section 6.5.

JET

Jet (Java Emitter Templates[8]) est un moteur de patrons qui s'apparente à Jsp et Php, mais ici cette approche s'applique à l'IDM. En effet, Jet permet de prendre en paramètre un modèle EMF, de le parcourir et d'accéder à ses propriétés pour produire un texte. Le patron Jet est composé de balises. Par exemple les API Java générées par EMF sont réalisées en Jet. On peut également utiliser Jet pour générer de la documentation ou tout ou partie d'un site web.

KET

Le moteur de patrons Ket (Kermeta Emitter Templates, [Faucher 09a]) est basé sur Jet. Il permet de générer du texte à partir d'un modèle EMF. La plus-value de Ket par rapport à Jet réside dans la possibilité d'utiliser la syntaxe Kermeta lors de la définition du patron Ket et de faciliter l'application de ce patron à partir d'un programme Kermeta réalisant déjà des tâches sur le même modèle source. Ket sera utilisé dans le chapitre 6 pour générer une application web de visualisation.

3.2.7 Expression de contraintes sur des modèles

Afin d'étendre l'expressivité d'UML lors de la spécification de contraintes, au-delà des contraintes structurelles, de typage et des multplicités, l'OMG a spécifié le langage formel OCL (Object Constraint Language, [OMG 12a]). Il permet de naviguer dans les modèles (statiques et dynamiques) pour atteindre et/ou contraindre les éléments de modélisation : invariants de classes et *pre/post* conditions d'opérations. OCL intègre différents types de collections, dont les `set`, `sequence` et `bag`.

OCL est sans effet de bord et ne peut donc pas modifier les modèles eux-mêmes, seulement les enrichir et préciser leur sémantique grâce aux contraintes spécifiées. Sous

8. `http://www.eclipse.org/modeling/m2t/`

réserve de rester conforme aux recommandations de l'OMG en la matière, OCL peut être étendu. OCL s'applique également sur les métamodèles Ecore. Nous utiliserons OCL dans la suite de ce chapitre pour définir des contraintes sur le métamodèle PTOM et dans le chapitre 5 pour contraindre un modèle de calendrier.

3.3 Besoins et concepts pour PTOM

Cette section présente les concepts qui doivent être développés dans le métamodèle PTOM pour répondre aux besoins exprimés dans les sections précédentes.

Nous traitons séparément la modélisation des événements et celle de leurs propriétés temporelles. Il en résulte des modèles complémentaires dont la cohérence mutuelle (à côté naturellement de la cohérence interne) doit être assurée. En effet, la structure composite d'un événement induit des contraintes sur les propriétés temporelles relatives de ses différents composants.

Si les apports du modèle que nous proposons se manifestent surtout dans le traitement des séquences d'occurrences d'événements répétitifs exprimées en intension, il est indispensable de proposer une approche homogène qui traite aussi bien des événements ponctuels, des dates concrètes et des séquences d'occurrences exprimées en extension.

Ces différents points sont abordés dans les sections suivantes. Nous présentons et commentons ensuite le métamodèle PTOM successivement dans ses aspects dédiés aux événements (cf. section 3.3.1) puis dans ceux traitant des propriétés temporelles (cf. section 3.3.2). Nous citons et commentons d'abord brièvement les modèles et environnements existants dont nous nous sommes inspirés, leurs lacunes et leurs intérêts respectifs. Un tableau récapitulatif est fourni en section 3.3.3 pour évaluer les fonctionnalités comparées de chacun.

3.3.1 Evénément

La plupart des modèles existants dédiés à la gestion d'événements (dans différents contextes : agenda, spectacles, disponibilité de ressources) proposent des moyens de spécifier la structure composite des événements. C'est le cas d'EventsML G2 et d'iCalendar (cf. section 2.3).

Nous avons cherché à approfondir la notion d'événement complexe et à clairement distinguer deux types de composition qui sont très différents dans leur sémantique et dans leurs traitements.

Le premier point de vue, le plus simple correspond à une décomposition intrinsèquement structurelle. Un événement donné, par exemple « *les Fêtes de Gand 2012* » propose plusieurs types d'activités aux festivaliers (concert, théâtre, animation de rue,...). L'événement *parent* et chacun de ses sous-événements obéissent à leurs propriétés temporelles respectives. La structure composite induit naturellement des contraintes par exemple du type : les occurrences des *fils* se tiennent durant la période d'occurrence du *père*.

Ces règles générales sont intégrées au modèle et vérifiées lors de son peuplement.

Ces considérations soulèvent des questions sur la granularité temporelle, et sur les relations géométriques et topologiques entre événements.

Un second point de vue concerne directement le temps et les événements récurrents. Lorsqu'on énonce que les « *Francofolies de La Rochelle ont lieu chaque année en juillet* », on établit une contrainte temporelle sur une abstraction de haut niveau. En fait, les « *Francofolies de La Rochelle* » ainsi référencées, n'ont nullement lieu, ce sont les éditions annuelles de ces Francofolies qui occupent effectivement une partie du mois de juillet. Il convient de rattacher le concept de « *Francofolies de La Rochelle* », qui est utile bien qu'abstrait, aux occurrences annuelles plus concrètes. Cette fois-ci, le lien est temporel et non structurel. On doit vérifier que la règle énoncée au niveau du *père* (i.e. : le composite avec « *annuel en juillet* ») est bien vérifiée par les *fils* (i.e. : les composants qui des éditions successives). Cette vérification est mise en œuvre automatiquement.

Les premier et second points de vue ne sont pas indépendants. En effet, décomposition structurelle et temporelle coexistent dans la réalité comme dans le modèle.

Enfin, il existe entre événements des relations autres que hiérarchiques. Il est important de représenter dans nos modèles des liens sémantiques entre événements (par exemple la tenue du salon de l'agriculture et la visite qu'y effectue le ministre) ou encore des relations temporelles indiquant une position relative : « *la pêche à pied à la Telline ne peut avoir lieu plus de 3 heures avant ou après la basse mer* ».

On gardera à l'esprit que les propriétés temporelles font *in fine* référence au calendrier. Le calendrier est constitué d'événements répétitifs, le plus souvent périodiques. Nous disposons de notre propre modèle de calendrier naturellement conforme à PTOM (cf. chapitre 5), ce qui permet de traiter uniformément à la fois les événements, leurs propriétés temporelles et les références au système calendaire en vigueur.

3.3.2 Propriétés temporelles

Comme cela a été présenté dans la section 2.3, le standard EventsML G2 permet de spécifier des ensembles de dates ou de périodes, des règles de périodicité ainsi que des exceptions (non périodiques). Il fournit un ensemble de correspondances avec iCalendar. Par rapport aux besoins constatés pour RMM2, EventsML G2 possède certaines limitations que nous levons dans le modèle temporel que nous proposons. Par exemple, EventsML G2 ne permet pas de spécifier des périodes récurrentes avec des rangs sur les jours quand on spécifie des mois particuliers, e.g. : « *du 15 mars au 15 octobre* » (cf. figure A.1.4). Or, dans la vie courante, il est fréquent d'être confronté à ce genre de situation. L'expression de positions relatives entre événements ainsi que la réutilisation de propriétés temporelles existantes ne peuvent pas être spécifiées. Enfin, les propriétés temporelles de l'événement sont dispersées. Par exemple, pour la spécification d'une règle de périodicité, des informations concernant l'heure d'occurrence (`DTSTART`, `DTEND`) sont contenues directement dans l'événement et non dans la règle de périodicité (`RRULE`).

La liste ci-dessous donne le pouvoir d'expression en intension souhaité pour la partie temporelle de notre métamodèle :

- instant périodique : périodicité avec une fréquence explicite : « *2 fois durant une période d'un mois* », « *2 fois par mois calendaire* » ;
- intervalle périodique : périodicité avec une fréquence implicite : « *du 1er jour de chaque mois au 2nd jour de chaque mois de février à chaque mois de novembre* ».
- *jump* (saut sur le suivant) : « *de 22h à 2h du jour suivant* » (cf. figure B.1.5). Dans le cas où l'événement a lieu tous les jours (e.g. : « *tous les jours de 22h à 2h* »), il est nécessaire d'ajouter un attribut spécifiant que l'on parle de la prochaine occurrence du jour. Par contre *jump* n'est pas nécessaire lorsque l'on traite l'expression : « *de 22h le mardi à 2h le mercredi* », car les horaires sont spécifiés sur des jours consécutifs.
- positionnement relatif : « *de 3 heures avant la basse mer à 3 heures après la basse mer* » ; l'occurrence de « *basse mer* » est également définie comme une propriété temporelle d'un événement, les termes « *avant* » et « *après* ».
- exception périodique : « *tous les jours sauf le lundi* », « *tous les jours sauf le 1er mai et le 25 décembre* ».

La modélisation du temps dans PTOM se fonde sur les standards de l'ISO qui sont eux-mêmes très voisins de toutes les modélisations produites dans d'autres contextes : données semi-structurées (XML), relationnelles ou ontologiques. En la matière, les aspects temporels restent très limités dans le standard UML (cf. SIMPLETIME). Une présentation succincte des éléments clés du modèle présenté dans l'ISO 19108 et que nous étendons dans PTOM est fournie en section 3.4.2.1.

Les travaux existants concernent principalement les dates ou les suites de dates concrètes (calendaires). Les modèles permettant d'exprimer des événements répétitifs en intension le font uniquement lors de la saisie des faits à insérer dans un agenda. Par contre, au niveau de la persistance et surtout du traitement, on se limite à gérer des collections de dates concrètes dans lesquelles la notion de périodicité reste seulement implicite. PTOM au contraire, donne un rôle de premier rang à l'expression et au traitement (persistance, vérification, interrogation, inférence) des expressions intensionnelles des propriétés temporelles.

Nous nous sommes inspirés des publications d'experts du traitement de la langue naturelle et en particulier [Battistelli 11, Becher 06, Scherp 09, Consortium 10] pour définir une typologie des expressions de la périodicité dans le langage commun des utilisateurs non spécialistes de logiques temporelles ou d'algèbres de processus. Il en ressort trois archétypes qui structurent fortement la partie temporelle de PTOM :

- la périodicité exprimée par une fréquence d'occurrence implicite, en utilisant une référence à un élément périodique du calendrier (instant ou intervalle : « *chaque année, les jeudis du mois de mars de 2010 à 2013* ») ;
- la périodicité exprimée avec une fréquence explicite dans un intervalle de temps donné : « *deux fois par semaine entre le 15/01/2013 et le 01/02/2014* » ;
- la périodicité exprimée avec une fréquence explicite utilisant une référence à un élé-

ment périodique du calendrier : « *chaque année quatre fois durant les deux premières semaines de juillet* ».

Ces trois types d'expressions partagent des concepts de PTOM et donnent lieu à différentes spécialisations qui seront évoquées en section 3.4.2.

3.3.3 Comparatif des modèles d'événements et de temps

Le tableau 3.1 confrontent l'expressivité des différents modèles présentés dans la section 2.3. Nous avons retiré du tableau 2.2 le modèle dédié au TAL TimeML et OWL-Time tout en ajoutant PTOM. Les critères de comparaison reprennent ceux de la section 2.4 en ajoutant la possibilité de spécifier des positions relatives entre instants et intervalles convexes ou non. Nous nous intéressons également à la présence d'un syntaxe concrète textuelle pour ces différents langages ou modèles.

Modèle / Langage	Relation de parenté	Relation entre événements	Relation d'occurrence	Périodicité (expression en intension)	Relation entre instants	Relation qualitative entre intervalles	Syntaxe concrète textuelle**
Glocal	✓	✓	–	–	qualitatif	convexe	–
OGRE	–	–	✓	multiple	qualitatif et quantitatif	convexe	
iCalendar	✓	–	✓*	unique	–	–	✓
EventsML G2	✓	✓	✓*	unique	–	–	–
Event Model F	✓	✓	✓	–	qualitatif	convexe	–
UML-MARTE	✓	–	✓	multiple	qualitatif et quantitatif	–	✓
Ptom	✓	✓	✓	**multiple**	**qualitatif et quantitatif**	**convexe (non-)**	✓

Relation en intension entre instants :
– qualitatif : « A *precedes* B »
– qualitatif et quantitatif : « A *precedes* B avec un délais de 3 heures »

* dans le cas d'une exception, la propriété RECURRENCE-ID peut être assimilée à une relation liant un événement à son occurrence

** autre qu'un format de sérialisation (XML, RDF...)

Tableau 3.1: Comparatif des modèles d'événements et de temps

L'approche que nous avons choisie est voisine de celles du projet Glocal et de « Event Model F », car nous séparons les préoccupations : d'un coté le domaine et de l'autre les événements sans négliger la manière de relier les deux.

En ce qui concerne les relations entre événements, nous souhaitons offrir des fonctionnalités telles que les proposent EventsML G2 et « Event Model F », notamment en incluant des sous-événements (`structural`) et des relations du type : `management`, `affiliation`, `participation`, etc (i.e. avec des aspects plus orientés vers la « sémantique »).

Nous gérons des événements répétitifs, la partie temporelle de notre métamodèle doit proposer des facilités pour les exprimer. Nous nous inspirons d'iCalendar, OWL-Time et MARTE et nous étendons l'ISO 19108 afin de bénéficier des éléments de base pour exprimer des propriétés temporelles.

Un événement doit pouvoir porter plusieurs règles de périodicité ce que ne permet pas par exemple iCalendar.

Des exceptions sous forme périodique doivent pouvoir être spécifiées pour exprimer le fait qu'un « *musée est ouvert tous les jours sauf le 1er mai et le 25 décembre* ». Cette possibilité a été retirée d'iCalendar dans sa dernière version.

Comme dans MARTE, nous définissons des relations qualitatives et quantitatives en intension entre instants.

Nos contributions originales sont principalement :

- la mise en avant de la relation qui existe entre un événement périodique et ses occurrences : `temporal` ;
- la spécification d'exceptions périodiques ;
- des relations qualitatives et quantitatives en intension entre instants périodiques : « *chaque occurrence de A precedes chaque occurrence de B avec un délai de 3 heures* » ;
- des relations qualitatives entre intervalles non convexes : chaque jour « *l'intervalle de 10h à 12h* » *meets* « *l'intervalle de 12h à 14*h ».

Nb. Les relations *precedes* et *meets* sont issues des relations d'Allen entre des instants et des intervalles convexes. Les notations que nous introduisons pour exprimer des relations similaires entre intervalles non convexes seront décrites dans la suite et plus particulièrement au chapitre 4.

3.4 Métamodèle d'événements temporels composites

Cette section présente et commente les différents métamodèles constituant Ptom. Nous discuterons d'abord le métamodèle d'événements composites. Ensuite, après avoir rappelé les points essentiels de la modélisation du temps dans les normes ISO 19108 et 8601:2004, nous traiterons successivement des différents aspects de la modélisation des propriétés temporelles et du traitement de la périodicité des occurrences d'événements.

3.4.1 PTOM : Evénements composites

Métamodélisation

Alors qu'UML traite des événements comme des artefacts attachés aux comportements (machines à états) déclenchant des opérations/transitions, plusieurs auteurs [Olivé 06, Zaki 11] ont souligné l'intérêt de promouvoir les événements comme éléments de modélisation de premier rang. Ceci afin de pouvoir les traiter de la même façon qu'on le fait pour les classifieurs.

La partie de Ptom dédiée aux événements fournit un métamodèle qui peut être vu comme un type abstrait permettant de construire des instances d'événements aux propriétés contrôlées.

La figure 3.4.1 décrit le métamodèle que nous utilisons. Un ensemble de clauses OCL précisent les contraintes que nous évoquerons ci-dessous. Nous ne les faisons pas apparaître comme telles ici par souci de concision et parce qu'elles utilisent des notions de topologie (ALLEN*) qui ne seront introduites que plus tard (chapitre 4). Néanmoins, nous explicitons les principales contraintes en langage naturel ci-après.

De manière générale, un événement est donc représenté par la classe `Event` qui supporte deux classes d'association réflexives.

L'une `semanticLink` traduit la possibilité de rendre compte de l'appariement d'événements pour des raisons diverses dont la sémantique est spécifiée dans un libellé libre et référencée par une URI donnée (ex : « *voir aussi* »). PTOM reprend d'EventsML G2 des relations de cette nature (« item relation » [9]).

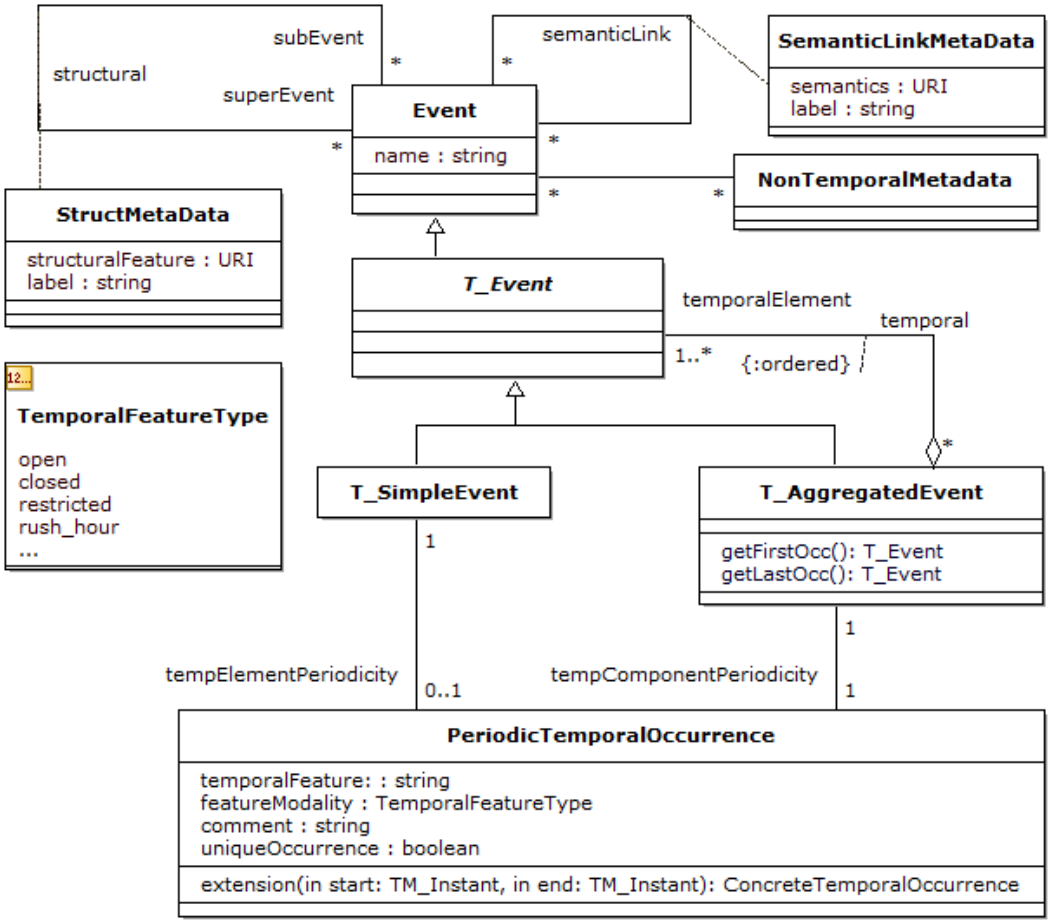


FIGURE 3.4.1: Métamodèle d'événement

L'autre (`structural`) rend compte d'une structure hiérarchique entre les événements. Le lien *père-fils* traduit le changement de niveau d'abstraction des instances de la classe

9. `http://cv.iptc.org/newscodes/itemrelation/`

`Event` considérées (e.g. : « *Les Francofolies 2012 / le concert du 14 juillet / la prestation d'un artiste de ce concert de 21h à 22h* »). L'argument qui justifie la classification, i.e. : l'association *père-fils* est indiquée, comme précédemment, par un libellé et une URI, e.g. : « *les concerts* ».

Le métamodèle prend également en compte la possibilité d'attacher à tout événement des métadonnées non temporelles (`NonTemporalMetadata`), celles-ci sont par exemple des propriétés de géolocalisation.

Un événement temporel (`T_Event`) est une spécialisation de la classe `Event`. Il s'agit d'une classe abstraite dont les instances sont celles des spécialisations apparaissant dans le patron composite utilisé. `T_SimpleEvent` représente les feuilles et `T_AggregatedEvent` les composites. Le lien qui agrège les composants est de type `temporal`. Il s'agit du lien temporel attachant un événement à ses occurrences. La relation `temporal` traduit exactement l'opérateur *dicing* du langage de Leban [Leban 86].

L'ensemble des occurrences d'un événement complexe (autrement dit composite ou `T_AggregatedEvent`) est ordonné dans le temps. Nous imposons la disjonction temporelle large (i.e. : la fin de l'occurrence précédente peut coïncider avec le début de la suivante) entre les occurrences d'un même `T_AggregatedEvent`, e.g. : les Francofolies / les éditions annuelles / les spectacles quotidiens.

La classe `PeriodicTemporalOccurrence` (`PTO`) permet d'attacher un événement de type `T_Event` à ses propriétés temporelles. La sémantique de la ressource concernée par la `PTO` est décrite avec l'attribut `temporalFeature` et la modalité de correspondance est définie à travers une énumération `TemporalFeatureType`. Par exemple pour un `T_Event` : « *Semaine de la danse* » la ressource serait « *Accès répétitions* » et la modalité : `restricted`. L'accès libre ou l'accès interdit serait éventuellement décrit dans d'autres `PTO` attachées au même événement.

Un autre exemple, « *les Fêtes de Gand* », le `temporalFeature` serait « *fréquentation* » et la modalité `rushHour`. Toujours pour « *les Fêtes de Gand* », l'attribut `uniqueOccurrence` permet d'imposer le caractère ponctuel de la `PTO`, i.e. : une seule instance, se déroulant soit à un instant unique soit durant un intervalle de temps convexe.

La méthode `extension(start, end)` permet de traduire de façon automatique et déterministe la spécification intensionnelle de la `PTO` en une série de dates concrètes (calendaires au sens de l'ISO 8601) sur une période (un intervalle de temps convexe) donnée en paramètre.

Un point important concerne la spécificité des rôles de la `PTO` selon sa connexion à un `T_SimpleEvent` ou bien à un `T_AggregatedEvent`. Le rôle `tempComponentPeriodicity` indique que la `PTO` décrit la périodicité des *fils* (`temporal`) du `T_AggregatedEvent` considéré. La propriété `tempComponentPeriodicity` est en conséquence obligatoire.

En revanche le rôle `tempElementPeriodicity` décrit la périodicité des occurrences du `T_SimpleEvent`. Ces occurrences ne sont pas implémentées dans le système mais seulement contraintes par la déclaration de la `PTO`. On peut, ou non, décrire la périodicité d'occurrence des instances d'un `T_SimpleEvent`, d'où le caractère optionnel de la propriété.

Il est important en effet de distinguer les occurrences d'un `T_Event` et les instances présentes dans le système. Il n'est pas assuré que toutes les occurrences spécifiées soient effectivement implémentées par des instances des *fils*. L'événement peut être spécifié comme annuel alors que seules quelques années sont implémentées comme instances à un moment donné du cycle de vie du système. La `PTO` est réputée exacte, et le peuplement du modèle incrémental et donc potentiellement incomplet. Ceci justifie l'attribut `uniqueOccurrence` qui ne peut, en conséquence, être un attribut dérivé. Une `PTO` peut être complétée au cours de son cycle de vie pour notamment se voir adjoindre des exceptions.

La classe `T_AggregatedEvent` possède deux méthodes qui retournent chacune un `T_Event`. Précisément, `getFirstOcc()` (respectivement `getLastOcc()`) permet d'accéder à la première (respectivement la dernière) occurrence décrite dans la `PTO` associée (non nécessairement implémentée en tant qu'instance dans le système). Il se peut qu'il n'y ait pas de dernière occurrence : cas de `PTO` avec une infinité (à droite) d'occurrences. Dans ce cas `getLastOcc()` retourne `void`.

Mise en œuvre

À ce point, il est utile de compléter la présentation précédente du modèle d'événements par l'exemple du diagramme d'instances de la figure 3.4.2 concernant les « *Francofolies de La Rochelle* ».

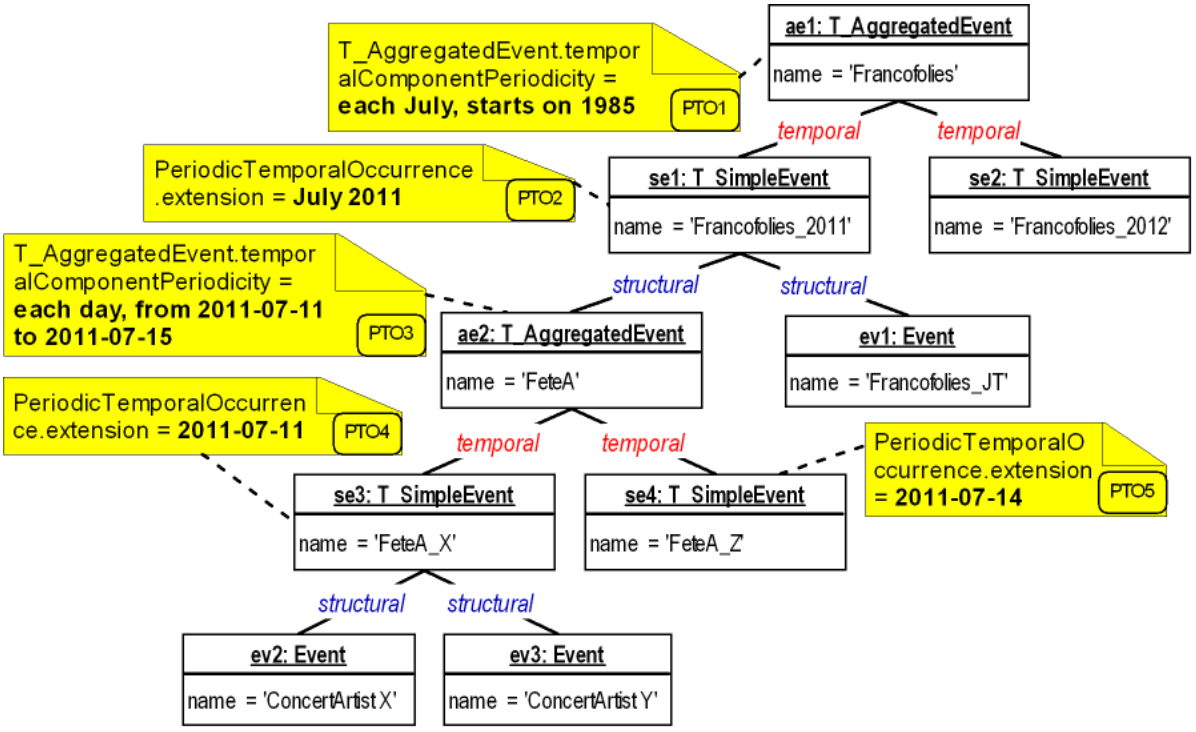


FIGURE 3.4.2: Exemple de diagramme d'objets : cas « *Francofolies* »

Pour chaque instance d'événement, la classe *parente* est indiquée. Les liens `structural` et `temporal` apparaissent, mais les `PTO` sont pour l'instant uniquement représentées par des notes en attendant que soit détaillée leur structure aux sections suivantes. De la même façon, les métadonnées non temporelles n'ont pas été considérées dans l'exemple.

La sémantique portée par le diagramme d'instance peut être résumée ainsi :

- Francofolies est le nom de l'événement composite `ae1` qui selon la `PTO(1)` se produit chaque mois de juillet depuis 1985.
- Deux occurrences de Francofolies sont décrites et respectivement nommées `Francofolies_2011 (se1)` et `Francofolies_2012 (se2)`. Selon `PTO(2)`, `Francofolies_2011` à lieu en juillet 2011. Ceci est conforme à `PTO(1)`. La temporalité de `Francofolies_2012` n'est pas décrite mais on peut inférer (d'après `PTO(1)`) que cela se passe en juillet pour une année qui reste à préciser (le nom n'étant pas significatif en la matière).
- Les Francofolies 2011 proposent différentes manifestations (`structural`) telles que la « *Fête à ...* » (`FeteA : ae2`) ou les « *SFR Jeunes Talents* » (`Francofolies_JT : ev1`).
- La `PTO(3)` indique que durant les `Francofolies_2011`, les `FeteA` sont des manifestations quotidiennes entre le 11 et le 15 juillet ce qui est cohérent avec les spécifications des événements *parents*.
- Deux occurrences de `FeteA`, nommées `FeteA_X (se3)` et `FeteA_Z (se4)` sont spécifiées pour le 11 juillet 2011 (`PTO(4)`) et le 14 juillet (`PTO(5)`).
- `FeteA_X` se compose (`structural`) d'une prestation de l'artiste qui a carte blanche (`ConcertArtiste_X : ev2`) et de celle de son invité (`ConcertArtiste_Y : ev3`).
- Les événements sans propriétés temporelles attachées instancient la classe `Event`.

Les autres informations (identification, coordonnées spatiales, qualité, métadonnées, ...) sont représentées par la classe informelle `NonTemporalMetaData`, et sont décrits précisément dans la norme ISO 19115 [ISO/TC211 10].

Contraintes additionnelles sur les éléments de métamodélisation

Une liste de contraintes complémentaires précisent la modélisation. Celles-ci doivent être vérifiées systématiquement lors de la création ou de la destruction d'instances du modèle. Ces règles sont de trois types :

- Les périodes d'occurrence d'un *fils* pour l'association `structural` sont nécessairement intégralement incluses dans l'ensemble des périodes d'occurrence du *père*.
- L'ensemble des occurrences des *fils*, selon un lien `temporal`, est conforme à la règle de périodicité du *père*. C'est-à-dire que les occurrences d'un *fils* ont lieu, dans leur ensemble, durant une et une seule des occurrences (potentiellement) répétitives du *père*. En outre, il n'existe pas d'autre *fils* partageant cette occurrence là.
- Étant donné la structure hiérarchique des événements complexes, les règles précédentes s'appliquent de façon transitive au long des branches de la hiérarchie.
- Si l'attribut `uniqueOccurrence` d'une `PTO` a pour valeur `true`, il n'existe effectivement

qu'une seule occurrence définie.
Les contraintes ainsi exprimées sur les `PTO` sont plus fortes que celles qui pourraient l'être sur des séries de dates concrètes. En effet, les `PTO` spécifient certes *in fine* une série de dates concrètes *via* l'opération `extension(start, end)`, mais y ajoutent une structure de partition qui identifie précisément chacune des occurrences répétitives.

Les contraintes induites sur les occurrences des *fils* concernent chacune de ces occurrences et non pas seulement la série non structurée des dates possibles.

3.4.2 Expressions temporelles

Nous présentons ici les métamodèles permettant d'exprimer les propriétés temporelles attachées aux événements *via* les `PeriodicTemporalOccurrence`. Nos apports s'appuient sur les concepts objets des normes ISO 19108 que nous présentons d'abord, en nous limitant aux concepts que nous réutilisons (section 3.4.2.1). Nous donnons ensuite un ensemble de points de vue sur PTOM qui constitue le corps de notre proposition dans ce domaine (sections 3.4.2.2 à 3.4.2.5).

3.4.2.1 La norme ISO 19108

Présentation introductive de la norme

La norme ISO 19108 [ISO 02] (Information Géographique : Schéma Temporel) spécifie et présente selon le paradigme objet (UML) un ensemble d'éléments de modélisation assortis de contraintes OCL utiles pour modéliser le temps et les calendriers. Les aspects géométriques du temps sont extensivement étudiés. La topologie du temps fait également partie du périmètre de la norme. Outre les relations d'ALLEN qui y sont intégrées, d'autres constructions génériques sont offertes pour modéliser les relations qualitatives mutuelles des instants et des périodes dans le temps.

Par convention, nous préfixerons systématiquement par « `TM_` » tous les noms des éléments empruntés à la norme ISO 19108.

L'ISO 19108 fait référence à l'ISO 8601:2004 pour ce qui concerne le codage des dates concrètes. Ce système de codage est repris par la plupart des systèmes implémentant des informations calendaires en particulier les SGBD, les gestionnaires d'agendas, etc. Enfin la norme ISO 19108 dans son annexe D présente certains calendriers parmi les plus usités et, en particulier, le calendrier grégorien auquel nous nous intéresserons en détail (cf. chapitre 5).

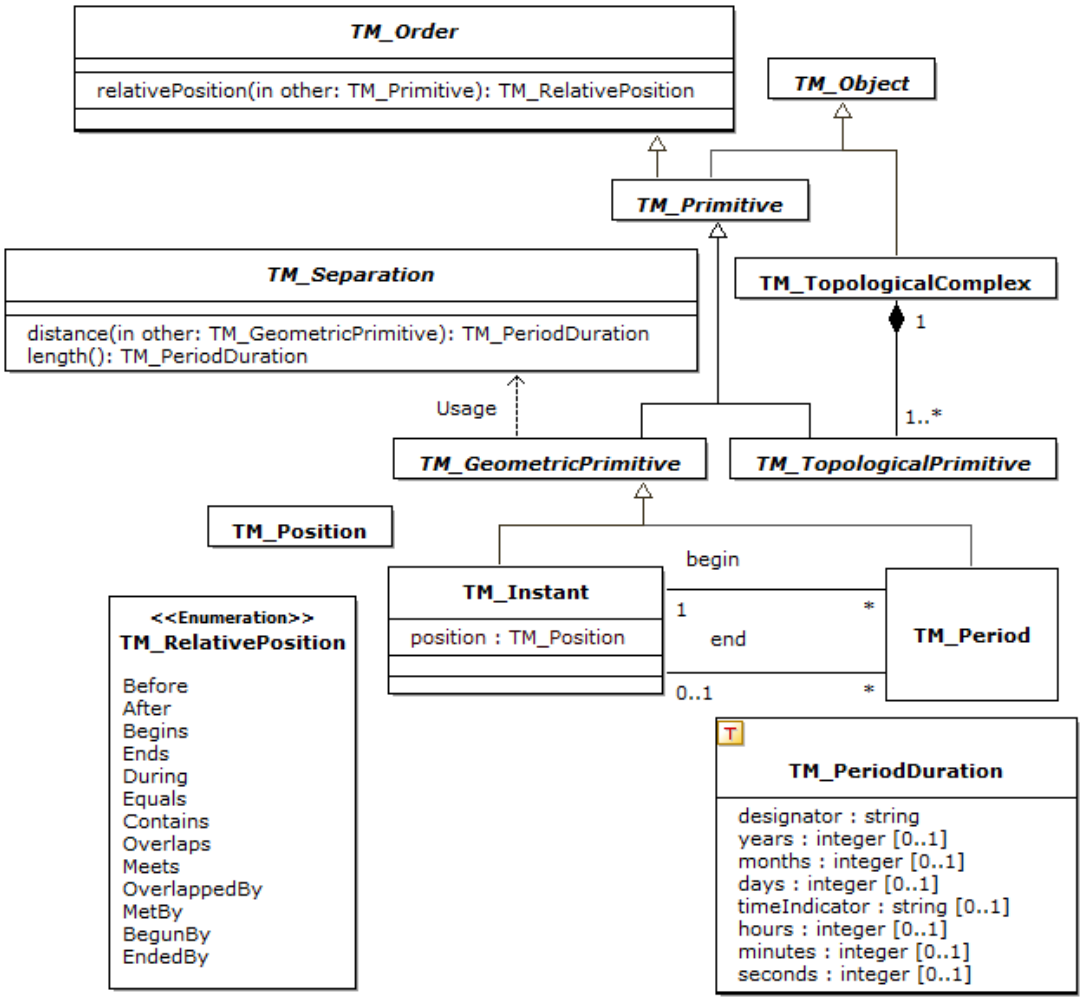


FIGURE 3.4.3: Extrait de l'ISO 19108

La figure 3.4.3 est un extrait de la norme, la classe centrale du diagramme est la classe abstraite `TM_Primitive` dont les deux spécialisations traitent respectivement des aspects géométrique et topologique des informations temporelles.

La norme suppose un ordre linéaire, préalablement donné sur une échelle numérique, qui repère et ordonne les instants. La synchronisation de cet ordre avec un calendrier particulier est prise en compte dans l'ISO 19108 (annexe C de l'ISO). La référence à l'ordre en question est représentée par la spécialisation de `TM_Order` en `TM_Primitive`. Ce point de vue sera repris et jouera un rôle central dans le chapitre 4 (traitant de la généralisation des relations d'ALLEN).

Dans ce contexte, `TM_GeometricPrimitive` possède deux spécialisations disjointes, à savoir `TM_Instant` et `TM_Period` qui indiquent respectivement un élément de dimension zéro et un élément de dimension un. Comme son nom l'indique, `TM_Instant` représente la notion classique d'instant, tandis que `TM_Period` représente un intervalle convexe, borné

par deux instants, jouant les rôles respectifs de début et de fin. La position d'un instant est fournie par l'attribut `TM_Position`.

En utilisant des services de la classe `TM_Separation`, `TM_GeometricPrimitive` offre des méthodes pour calculer des durées (appel à la classe `TM_Separation`) : écart (*single link*) entre deux de ses instances (instants, intervalles ou mixte) ou bien durée : différence positive entre fin et début pour un `TM_Period` et naturellement zéro pour un `TM_Instant`.

Les distances et les durées sont des valeurs numériques avec des unités temporelles figurant dans l'énumération `TM_PeriodDuration` qui doivent être interprétées par rapport à un référentiel donné `TM_ReferenceSystem`. L'ISO 19108 consacre une section entière à cette préoccupation. `TM_Duration` permet de spécifier une durée en additionnant les valeurs spécifiées pour les différentes unités. Par exemple une instance (`years=2`, `months=3`) indique une durée de 2 ans et 3 mois. Il est clair que la granularité est celle du mois et que la durée correspondante sera de 15 mois. Si l'on souhaite une granularité plus fine, il faut l'exprimer nécessairement en jours, heures, minutes...

Pour les aspects topologiques, la référence à la classe `TM_Order` permet, par l'intermédiaire de son opération `relativePosition(other)`, de réutiliser les relations d'Allen (cf. énumération `TM_RelativePosition`), que ce soit entre deux points (`TM_Instant`) ou entre deux intervalles convexes (`TM_Period`). La sémantique des relations d'Allen est redéfinie dans le modèle par un ensemble de contraintes formelles explicites (OCL).

Le métamodèle PTOM (Periodic Temporal Occurrence Metamodel) : une extension de l'ISO 19108

Comme nous l'avons vu précédemment, les événements traités ayant souvent un caractère temporel répétitif, il est intéressant de pouvoir décrire une série de dates, non pas en extension, mais en intension [Carnap 47] comme par exemple : « *le festival a lieu chaque 1er jeudi du mois de mai* ». L'ISO 19108 ne permet pas de définir des expressions en intension.

Ainsi nous proposons de compléter l'ISO notamment avec les concepts utiles à l'expression en intension de périodicités temporelles [Faucher 12a] tels que :

- une durée avec les notions de « *semaine* », de « *siècle* » et de « *comparateur* », permettant de spécifier que la durée entre deux instants est, par exemple : « *inférieure à 2 heures* » ;
- des descripteurs périodiques utilisant les termes du calendrier auxquels sont associés un rang, e.g. : « *1er lundi* » ;
- des positions temporelles relatives entre des occurrences, e.g. : « *2 heures avant l'ouverture du musée* ».

La création d'un nouveau métamodèle d'expressions temporelles et l'utilisation d'un langage dédié (DSL) sont ici appropriées car, d'une part, nous avons une base à étendre (i.e. : l'ISO 19108) définie sous forme d'un modèle UML, et d'autre part, nous souhaitons instancier et outiller le métamodèle notamment avec une syntaxe textuelle (cf. section 3.5). Cette extension a été réalisée en utilisant la modélisation par aspects (AOM, As-

pect Oriented Modeling) et plus particulièrement la composition de modèles (cf. section 3.2.4). Ceci permet de séparer les préoccupations liées à la norme ISO de celles liées aux concepts métier à intégrer. Cette extension a été réalisée en plusieurs étapes : en ajoutant tout d'abord le concept de règle de périodicité, puis de position relative et celui d'événement. Ces traitements ont été effectués avec les fonctionnalités d'AOM de KERMETA [Muller 05] et le moteur de composition KOMPOSE [France 07].

3.4.2.2 Expressions calendaires et périodiques (extension/intension)

Cette section présente les différents volets de la modélisation des occurrences de phénomènes périodiques. Sa structure suit celle du diagramme de la figure 3.4.4 qui détaille la structure des `PTO`.

Nous traiterons donc successivement des règles décrivant en intension des séries d'occurrences périodiques `PeriodicRule`. Il est indispensable de traiter de façon homogène de telles règles et des séries de dates concrètes apparaissant dans de nombreux cas d'utilisation. Nous présenterons donc en second lieu la classe `ConcreteTemporalOccurrence` (`CTO`) dont les instances sont des séries de dates calendaires exprimées en extension. Il est également utile de permettre de spécifier des exceptions aux règles générales ci-dessus (par exemple : « *un musée ouvert tous les jours sauf les mardis et durant le mois d'août* ». Cet aspect est pris en charge par la classe `TemporalException`.

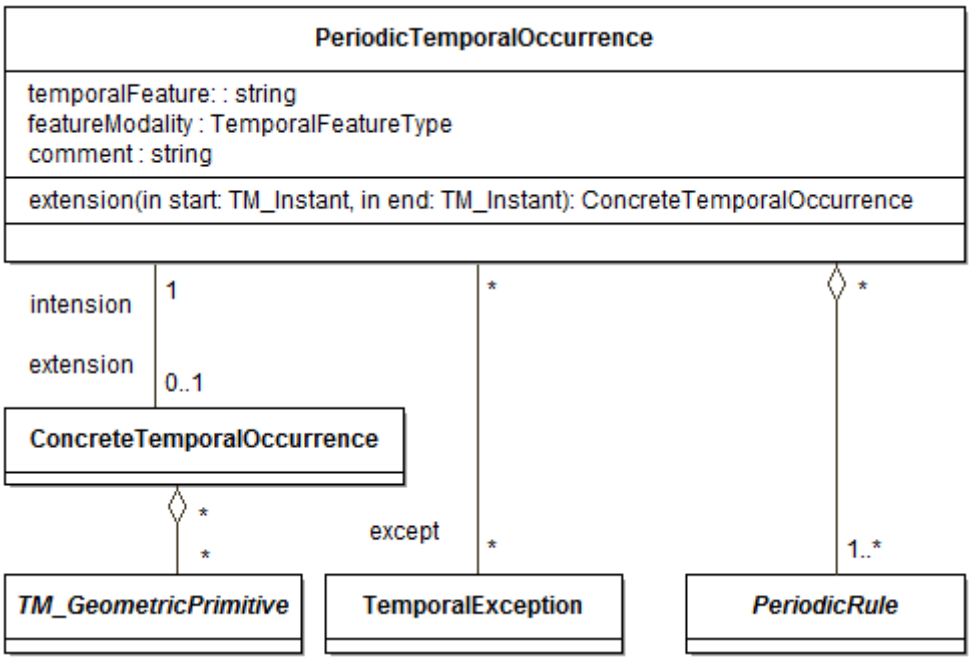


FIGURE 3.4.4: Différents types d'Occurrences temporelles

Dans notre problématique, les événements périodiques jouent un rôle prépondérant,

car ils sont fréquents dans de nombreuses applications métier et sont la base des calendriers et, plus généralement, de notre rapport au temps.

Du point de vue de PTOM, toutes les expressions temporelles exprimées en intension sont périodiques. Lorsque cela est nécessaire, on considère des expressions périodiques n'ayant qu'une seule occurrence. Ce point de vue est pragmatique et déjà admis dans d'autres communautés (traitement du signal, analyse de Fourier). Pour nous, cela représente un principe unificateur.

LEBAN [Leban 86] et TERENZIANI évoquent les différents types de périodicité, exacte ou approchée. Nos modélisations permettent de prendre en compte des phénomènes répétés non strictement périodiques (événements de durée variable dont le début est périodique).

Selon ce principe nous représenterons ces phénomènes par des `PTO`, même si le caractère « périodique » n'est pas rigoureusement vérifié.

D'ailleurs, les calendriers eux-mêmes sont construits sur des événements seulement pseudo-périodiques. Se référer aux éléments calendaires sera pour nous le moyen de gérer ces événements répétés non strictement périodiques.

3.4.2.3 Descripteur calendaire périodique

Comme il vient d'être dit, les calendriers fournissent des concepts élémentaires pour représenter les occurrences d'événements répétitifs. La figure 3.4.5 présente un diagramme de classes dédié à la représentation des concepts calendaires que nous commentons ci-dessous et réutiliserons ensuite.

La classe de base de ce premier modèle est `CalendarPeriodicDescriptor` qui sert effectivement à décrire la périodicité d'occurrences d'événements quelconques.

Les types offerts correspondent aux différentes granularités des éléments calendaires. `DayDescriptor` et `MonthDescriptor` sont des éléments calendaires nommés, identifiables par leur nom mais aussi par leur *rang* dans l'élément de granularité supérieure considéré.

On notera la similitude profonde entre cette notion de *rang* introduite dans notre modèle et l'opérateur *slicing* défini par LEBAN [Leban 86].

« *Mardi* » peut être référencé par le nom d'un `DayDescriptor` (`day = tuesday`), mais aussi comme le deuxième jour de la semaine, par un couple constitué d'un `CalendarUnitDescriptor` avec `value = day`, assorti d'un *rang* (`rank = 2`), puis d'un nouveau `CalendarUnitDescriptor` avec `value = week` sans *rang* associé. Précisément, lorsqu'on spécifie un `CalendarUnitDescriptor` sans préciser de *rang*, cela signifie que toutes les occurrences (quelque soit leur *rang*) sont considérées (i.e. : « *chaque jour* »).

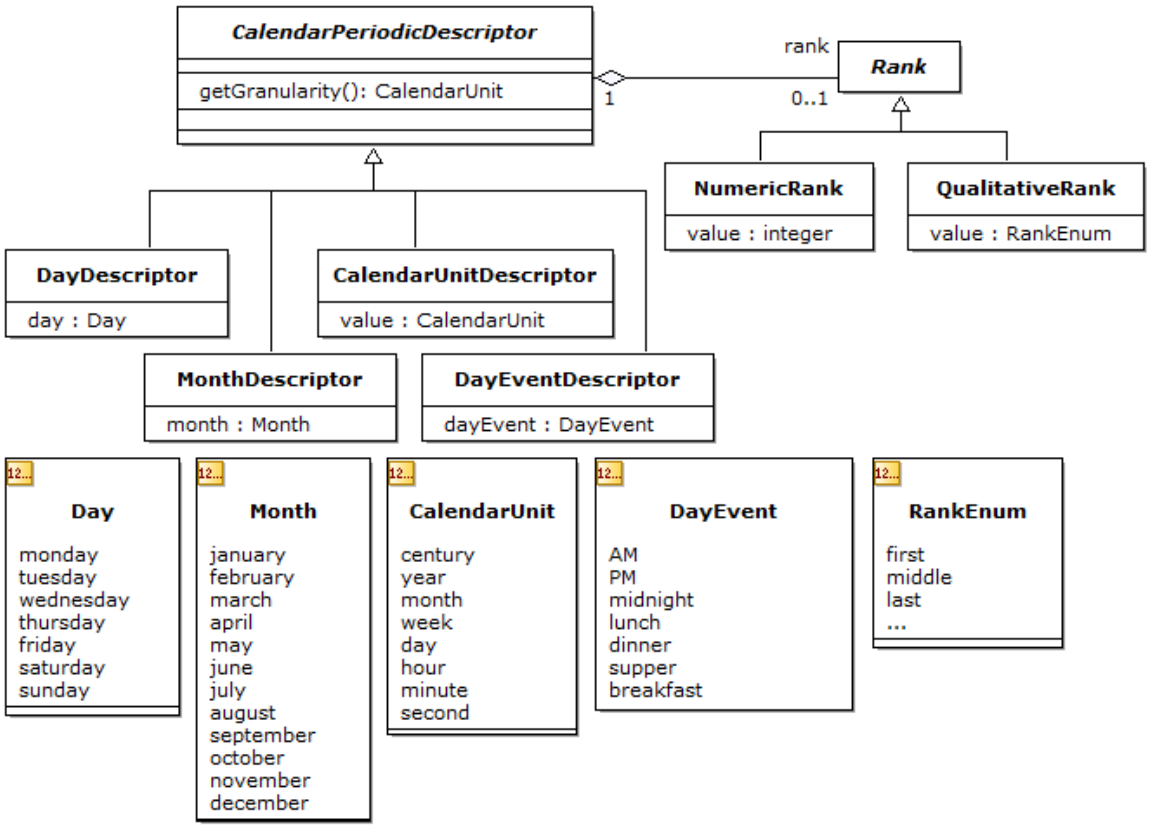


FIGURE 3.4.5: Descripteur calendaire périodique

Ainsi, avec une syntaxe intuitive (la grammaire formelle associée à PTOM est plus complexe et sera présentée à la section 3.5) on peut exprimer ce qui précède sous la forme :

« *chaque mardi* » :

```
{
  { CalendarUnitDescriptor.value = day ; NumericalRank = 2 }
  { CalendarUnitDescriptor.value = week }
}
```

de la même façon, on pourrait spécifier la règle « *chaque deuxième semaine de l'année* » :

```
{
  { CalendarUnitDescriptor.value = week ; NumericalRank = 2 }
  { CalendarUnitDescriptor.value = year }
}
```

ou enfin « *chaque vendredi de chaque semaine du deuxième mois de l'année* » :

```
{
  { DayDescriptor.day = friday }
  { CalendarUnitDescriptor.value = week }
  { CalendarUnitDescriptor.value = month ; NumericalRank = 2 }
  { CalendarUnitDescriptor.value = year }
}
```

L'énumération `CalendarUnit` est naturellement ordonnée, comme le sont également les autres : `Day` et `Month`.

Les *rangs* peuvent être codés par un nombre évidemment contraint par la nature du calendrier choisi.

Une autre façon de coder le *rang* par une spécification qualitative (utile lorsque le *rang* numérique est variable) est de se référer à des prédicats ordinaux tels que `first`, `last` et `middle`.

Il est ainsi aisé de spécifier « *le dernier jour du mois* » :

```
{
  { CalendarUnitDescriptor.value = day ; QualitativeRank = last }
  { CalendarUnitDescriptor.value = month }
}
```

Ce dernier exemple illustre et précise en outre ce que l'on entend par « modélisation de phénomènes *pseudo* périodiques en référence aux éléments calendaires » (cf. plus haut).

Le modèle permet enfin de se référer à des événements implicites étiquetés comme tels par l'énumération `DayEvent`. Cette liste peut être étendue selon les contextes applicatifs. PTOM n'est actuellement pas outillé pour gérer la sémantique de ces éléments, mais il conserve cette donnée.

La construction des descriptions est soumise à des contraintes de correction syntaxique et sémantique.

Ainsi :

- les `CalendarUnitDescriptor` apparaissant dans la spécification d'une règle de périodicité utilisant des `CalendarPeriodicDescriptor` doivent respecter l'ordre induit par l'énumération `CalendarUnit` (i.e. : par granularités croissantes) ;
- un type donné de `CalendarPeriodicDescriptor` peut apparaître au plus une fois dans la liste ;
- les rangs sont soumis aux contraintes de multiplicité énoncées dans la modélisation du calendrier au chapitre 5 ;
- pour chaque `CalendarPeriodicDescriptor` nous vérifions également qu'il ne soit ni sur- ni sous-spécifiée, i.e. :
  - sur-spécification : « *chaque 2e jour de chaque mois de chaque année* », « *de chaque année* » n'est ici pas nécessaire ;
  - sous-spécification : « *chaque 2e jour* », cette expression n'est pas suffisante ;
  - l'ensemble des cas possibles est fourni par la table de décision de l'annexe D.

On notera que le processus décrit plus haut produit des suites d'instants périodiques et ne traite pas pour l'instant de suites d'intervalles.

3.4.2.4 Expression des règles de périodicité temporelles

Les propriétés temporelles sont attachées aux instances du métamodèle d'événements *via* les `PTO`. Ces dernières portent la sémantique des propriétés temporelles concernées (`TemporalFeature`, `FeatureModality`).

La classe `PeriodicRule` spécifie une règle de périodicité « élémentaire », c'est-à-dire mono-fréquentielle (cf. figure 3.4.6). En conséquence, pour décrire des périodicités pluri-fréquentielles, nous rattachons la `PTO` à plusieurs `PeriodicRule`. la sémantique de cette agrégation est l'additivité des règles de périodicité, comme sont additionnés les signaux dans les décompositions de Fourier.

Dans ces conditions, les `PeriodicRule` peuvent aisément être réutilisées par plusieurs `PTO` et donc par plusieurs événements.

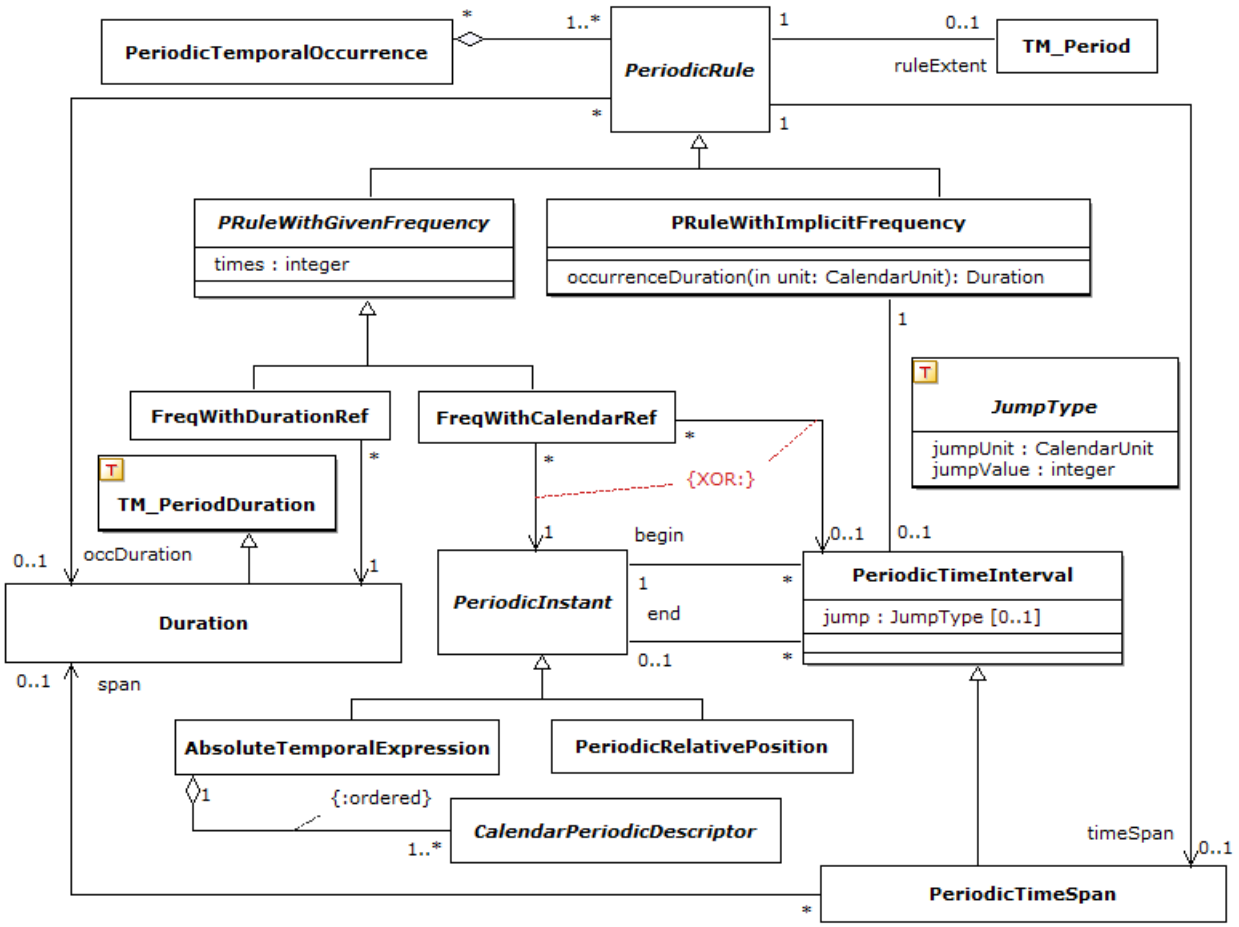


FIGURE 3.4.6: Modélisation des règles de périodicité des occurrences

Règle de périodicité : PeriodicRule

La construction des `PeriodicRule` est fondée sur les classes `PeriodicInstant` et `PeriodicTimeInterval` qui sont mutuellement associées de la même manière que `TM_Instant` et `TM Period` dans l'ISO 19108. Cependant, contrairement à l'ISO 19108, nous traitons ici de suites d'instants et d'intervalles non convexes (bien que potentiellement réduits à une seule instance si nécessaire). De plus, l'ISO 19108 définit des dates concrètes en exten-

sion alors que notre modèle permet la description d'instants (et donc d'intervalles) en intension. Ici nous soulignons que `PeriodicInstant` n'est aucunement une spécialisation de `TM_Instant`.

Un `PeriodicInstant` peut être de deux types :

- soit il est défini par sa relation avec un autre `PeriodicInstant` déjà spécifié (cas d'une `PeriodicRelativePosition`) ;
- soit il est défini individuellement, *via* la classe `AbsoluteTemporalExpression` qui agrège une ou plusieurs instances de `CalendarPeriodicDescriptor`.

Nous rappelons que cette agrégation est implicitement ordonnée conformément aux granularités calendaires croissantes et qu'un type de `CalendarUnitDescriptor` ne peut apparaître plus d'une fois dans l'agrégation.

Le cas des positions relatives sera détaillé à la fin de cette section. Ceci étant, un `PeriodicTimeInterval` se définit en spécifiant conjointement deux `PeriodicInstant`, l'un avec le rôle `begin` et l'autre avec le rôle `end`. Le début est obligatoire et la fin optionnelle. Il existe une contrainte qui assure que les `PeriodicInstant` respectivement de rôle `begin` et `end` ont la même fréquence d'occurrence. De surcroît ceci est nécessaire pour pouvoir garantir que les composants convexes d'un `PeriodicTimeInterval` sont bien disjoints deux à deux. Cette contrainte est automatiquement vérifiée dans le modèle.

Ceci étant établi, PTOM offre trois manières différentes de définir une règle périodique :

- (1) avec `PRuleWithGivenFrequency`, la fréquence de la `PeriodicRule` est implicite et résulte d'une association avec un `PeriodicTimeInterval`.
  ex. : « *du mardi au jeudi de chaque mois* » (fréquence implicite : mensuelle).
  Nb. : ici, la fréquence et la phase du phénomène sont précisément connus.

  avec `PRuleWithGivenFrequency`, deux cas sont possibles :
- (2) la fréquence est fixée en référence à une durée (`FreqWithDurationRef`).
  ex. : « *6 fois tous les deux mois* ».
  On utilise ici le type de données `Duration` qui étend `TM_Duration` de l'ISO 19108 en ajoutant des unités supplémentaires (`week`).
  Nb. : seule la fréquence moyenne du phénomène est spécifiée.
- (3) la fréquence est fixée en référence à un élément calendaire (`FreqWithCalendarRef`). L'élément calendaire référencé peut être soit un `PeriodicTimeInterval` soit un `PeriodicInstant`.
  ex. : « *2 fois du 15^{e} au dernier jour de chaque mois* ».
  ex : « *2 fois le premier dimanche de chaque mois* ».
  Nb. : comme dans le cas précédent, seule la fréquence moyenne du phénomène est spécifiée.

Dans tous les cas, une fois la `PeriodicRule` définie, on peut lui associer un `ruleExtent`, qui définit le support temporel (`TM_Period`) dans lequel s'applique finalement la `PeriodicRule`. Si la `PeriodicRule` est infinie, cela représente un moyen d'en limiter la durée. Par exemple on définira une règle « *tous les premiers lundis de chaque mois* » et un

`ruleExtent` : « *du 01/01/2012 au 31/12/2017* ».

Si la durée des occurrences est fixe, on peut la spécifier *via* le rôle `occDuration` vers `TM_Duration`. Si elle peut être calculée, la propriété est dérivée.

Une variante du `ruleExtent` est spécifiée *via* la classe `PeriodicTimeSpan` qui peut être associée à une `PeriodicRule`. Il s'agit de traiter des phénomènes périodiques qui se produisent pendant une série, elle-même périodique, de périodes, e.g. : « *entre 2010 et 2015, chaque année, une fois par semaine de juillet à août* ».

`PeriodicTimeSpan` spécialise `PeriodicTimeInterval` en imposant que la fin de l'intervalle soit spécifiée. Si ce n'est pas le cas, on définit une durée (rôle `span` vers `Duration`) qui lève l'indétermination de l'extrémité droite de l'intervalle (`end = begin + span`).

Le `PeriodicTimeSpan` doit être vu comme une fenêtre ayant des occurrences périodiques pendant lesquelles la `PeriodicRule` associée s'applique. Au contraire, elle ne s'applique pas en dehors de ces créneaux.

En conséquence, l'exemple précédent, « *une fois par semaine de juillet à août* » sera modélisé par :

```
{
  PRuleWithGivenFrequency : "une fois par semaine",
  timeSpan : "de juillet à août",
  ruleExtent : "de 2010 à 2015"
}
```

Dans des cas pratiques, il arrive fréquemment que le début et la fin d'un `PeriodicTimeInterval` ne fassent pas référence au même élément calendaire, i.e. : « *de 11h à 2h du matin* » (cf. figure B.1.5). En effet, il faut comprendre « *de 11h à 2h du jour suivant* », sans cette précision l'expression serait fausse, car la fin de l'intervalle précèderait le début. Pour permettre la spécification de telles expressions, nous proposons la notion de *jump* (saut à la prochaine occurrence) et le type `JumpType`.

Un *jump* se définit selon un des descripteurs décrivant la fin d'un intervalle. L'utilisateur spécifie un `JumpType` avec une unité (`jumpUnit`) choisie parmi les littéraux de `CalendarUnit` et une valeur (`jumpValue`) qui spécifie le nombre d'occurrences à « passer ». `jumpUnit` a la même unité que le descripteur concerné. Dans notre exemple précédent, `jumpUnit` vaudrait `Day` et `jumpValue=1`. Si `jumpValue>1`, l'utilisateur doit se prémunir des recouvrements possibles entre occurrences d'événements. Bien entendu, l'existence du *jump* est contrainte, nous fournissons ci-dessous ces conditions :

S'il n'y a pas de propriété `end`, l'attribut *jump* est `null`.

```
context PeriodicTimeInterval inv jump_1 :
  self.end->oclIsUndefined() implies
  self.jump->oclIsUndefined();
```

Si un *jump* est spécifié, alors sa valeur (`jumpValue`) est positive.

```
context PeriodicTimeInterval inv jump_2 :
  (not self.jump->oclIsUndefined()) implies
  self.jump.jumpValue >= 0;
```

Si le type de la propriété `end` est `PeriodicRelativePosition`, l'attribut *jump* est `null`.

```
context PeriodicTimeInterval inv jump_3 :
  self.end.oclIsKindOf(PeriodicRelativePosition) implies
  self.jump->oclIsUndefined();
```

Lorsque l'attribut *jump* existe, alors la propriété `end` est de type `AbsoluteTemporalExpression` et le `CalendarPeriodicDescriptor` associé n'est pas spécifié avec un `rank`.

```
context PeriodicTimeInterval inv jump_4 :
  (not self.jump->oclIsUndefined()) implies
  (
  (not self.end.oclAsType(AbsoluteTemporalExpression)->oclIsUndefined()) and
  (self.end.oclAsType(AbsoluteTemporalExpression).calendarPeriodicDescriptor->select(
  e | e.oclIsKindOf(self.jump.jumpUnit)).rank->oclIsUndefined())
  )
```

Gestion des Exceptions

Dans les applications pratiques, il est courant que les règles de périodicité des occurrences d'événements soient définies de manière générale, puis corrigées en formulant des exceptions, e.g. : « *Le musée est ouvert tous les jours de 9h à 16h sauf les mardis* ». Il est indispensable d'introduire dans PTOM cette possibilité de spécifier les règles d'occurrence en deux temps : loi générale avec des exceptions (cf. figure 3.4.7).

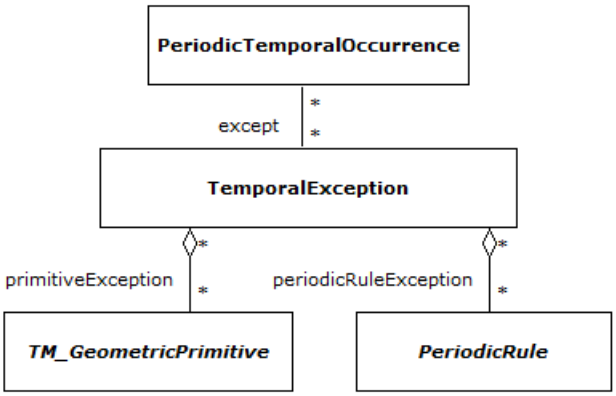


FIGURE 3.4.7: Exceptions

Dans cet esprit, la classe `TemporalException` permet de retirer des occurrences d'une `PTO` donnée. `TemporalException` est attachée aux `PTO` et non aux `PeriodicRule` car au contraire des secondes, les premières ne sont pas réutilisables. Il convient en effet que les exceptions soient définies de manière indépendante des règles qu'elles modifient. L'application des exceptions aux `PTO` (et non aux `PeriodicRule`) évite de gérer la composition d'exceptions.

Les `TemporalException` s'expriment selon les cas comme des ensembles de `TM_Geometric-Primitive` ou des `PeriodicRule`. De ce fait, les `TemporalException` ne peuvent elles mêmes supporter des exceptions.

La sémantique des `TemporalException` est ensembliste. Les `PTO` définissent en intension des ensembles d'occurrences, les `TemporalException` également. Déclarer une `Temporal-Exception` associée à une `PTO` revient à opérer une différence d'ensembles d'occurrences :

$$\{PTO\} - \{TemporalException\}$$

Ainsi, il n'est pas nécessaire que les occurrences spécifiées dans la `TemporalException` existent effectivement dans l'ensemble des occurrences de la `PTO` concernée. La sémantique ensembliste peut être paraphrasée de manière prédicative par :

$$[PTO\ et\ non(TemporalException)]$$

Positions relatives

La possibilité de spécifier des relations mutuelles (géométriques/quantitatives ou topologiques/qualitatives) entre occurrences d'événements devient nécessaire lorsqu'on se préoccupe d'applications pratiques. L'importance de ces aspects est avérée par le succès des calculs d'Allen sur les relations topologiques entre intervalles convexes. Ceci a été complété par [Meiri 96] qui étudie la combinaison des relations qualitatives (i.e. Allen) et quantitatives [Dechter 91].

Ptom offre la possibilité de spécifier de telles relations entre occurrences d'événements et nous détaillons ci-après ces aspects de notre modèle.

Cependant, il est apparu indispensable de pouvoir spécifier des relations topologiques non plus seulement au niveau des occurrences, mais également au niveau des événements eux-mêmes, c'est-à-dire déclarer des règles sur l'événement *père* (composite) d'une relation `temporal`, et non uniquement sur ses *fils* (composants).

Par exemple, considérons un modèle de calendrier, où « *mardi* » et « *mercredi* » sont deux instances de `T_AggregatedEvent` partageant la même `PTO` associée à la `Periodic-Rule` « *une fois par semaine* » avec des *fils* respectifs (`temporal`) instances de `T_Simple-Event`, indiquant la suite des mardis et mercredis hebdomadaires.

Nous proposons de déclarer une règle générale : « *mardi meets* mercredi et mercredi Met_by* mardi* ».

La sémantique de cette règle est qu'on peut apparier les mardis et mercredis hebdomadaires de façon à assurer pour chaque couple de rang `i` :

$$\forall i\ :\ I\ \bullet\ mardi(i)\ meets\ mercredi(i)$$

où *meets* est la relation d'Allen classique.

*meets**et *Met_by** sont des généralisations des relations d'Allen qui s'appliquent aux intervalles non convexes, et donc aux événements à occurrences répétées. Le chapitre 4 traite exhaustivement de ces relations et présente les généralisations que nous

avons établies et leurs propriétés. En pratique, les relations généralisées qui constituent $ALLEN^*$ permettent de spécifier des règles dans un langage dédié qui complète PTOM et permet le raisonnement et la validation des modèles.

Dans PTOM lui-même, seules les relations d'ALLEN classiques et celles de SIMPLETIME d'UML 2.0 sont autorisées. Cette séparation permet de distinguer les préoccupations : spécification, vérification et raisonnement. En outre, seuls les `PeriodicInstant` sont directement concernés par ces définitions. Ici nous nous plaçons dans un cas similaire à MARTE, i.e. : synchronisation d'horloges. Cela n'est pas une restriction du pouvoir d'expression, mais la prise en compte du fait que `PeriodicInstant` est l'élément de base pour la construction des `PeriodicTimeInterval` qui eux même s'agrègent pour former des `PeriodicRule` décrivant de intervalles non convexes.

Se limiter à définir des `PeriodicRelativePosition` sur des instants est donc justifié et efficace. Le cas où `PeriodicInstant` est défini par une `AbsoluteTemporalExpression` a été traité plus haut.

Nous envisageons maintenant (cf. figure 3.4.8) celui où on définit une position relative d'un `PeriodicRelativePosition` qui spécialise `PeriodicInstant` jouant le rôle `source` par rapport à un second `PeriodicInstant` prédéfini qui joue le rôle de cible : `target`.

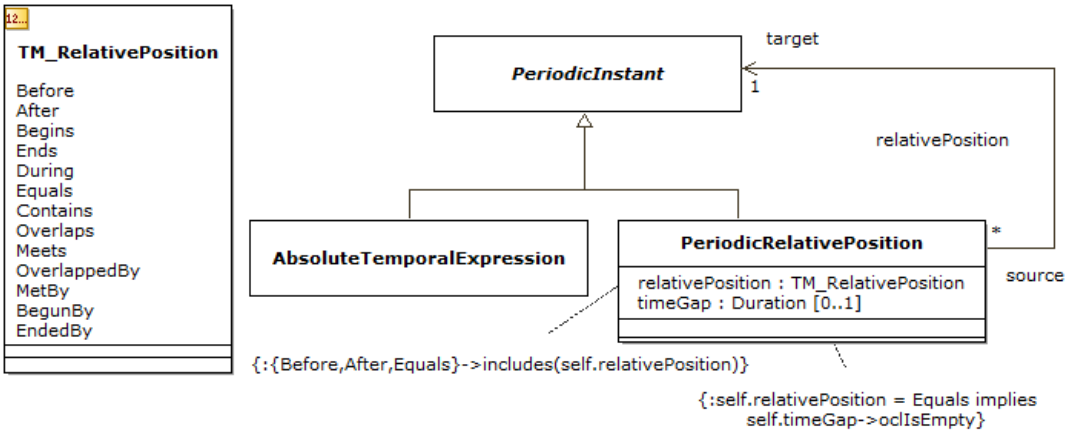


FIGURE 3.4.8: Position relative des occurrences de deux instants périodiques

Ces définitions ne vont pas sans que des contraintes soient imposées. Il est nécessaire que les deux `PeriodicInstant` `source` et `target` aient le même nombre fini d'occurrences. (Nb. : il est possible d'étendre la définition à un nombre d'occurrences infini, à condition qu'une occurrence initiale soit spécifiée pour l'un et l'autre instants). En fait il suffit que l'on puisse définir une bijection monotone entre les deux ensembles d'occurrences temporellement ordonnées.

Les occurrences étant ainsi appariées, deux attributs permettent de spécifier les aspects respectivement topologiques et géométriques de leur relation mutuelle. La relation topologique entre les deux éléments d'un couple d'occurrences est spécifiée en référence à ALLEN et s'exprime par une valeur choisie dans l'énumération `TM_RelativePosition`

de la norme. Puisqu'il s'agit d'instants, seuls $Before$, $After$ et $Equals$ sont autorisées. L'attribut `timeGap` qui représente une durée (`Duration`) est significatif lorsque les relations d'ALLEN sont choisies dans $\{Before, After\}$ et indique alors le délai entre les deux occurrences considérées. Dans le cas *equals*, le `timeGap` est forcé à la valeur zéro. La position relative ainsi définie est celle de l'instant courant (rôle `source`) par rapport à l'instant référencé par l'association `relativePosition` (rôle `target`).

3.4.2.5 Exemple d'instanciation

La figure 3.4.9 fournit un exemple de règle de périodicité sous forme d'un diagramme d'instances et sa contrepartie textuelle générée (cette forme sera discutée dans la section suivante). Celle-ci décrit l'ouverture d'un magasin : « *du lundi au vendredi, de 10h à 20h sauf le jeudi* ». Le texte généré utilise une règle de périodicité « *de 10h à 20h* » et une période de validité récurrente « *du lundi au vendredi* ». Une exception pour le jeudi est également ajoutée. La taille du modèle montre ici l'intérêt de la représentation textuelle pour sa concision.

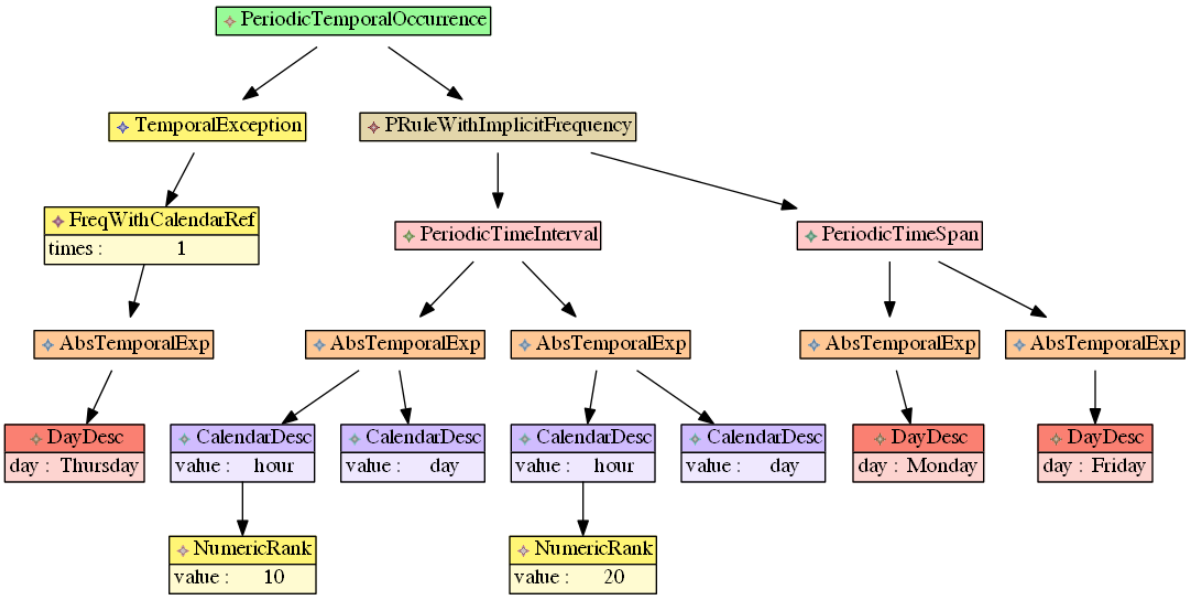


FIGURE 3.4.9: Exemple de règle de périodicité : modèle d'instances et forme textuelle

3.5 Grammaire (formelle) image de PTOM

3.5.1 Objectif

L'objectif de la grammaire est d'adjoindre au métamodèle global PTOM une contrepartie textuelle afin d'offrir à l'utilisateur une vue complémentaire à la spécification arborescente des données qui est classiquement fournie par EMF ou bien par un diagramme d'objets. Nous avons utilisé l'outil IDM XTEXT pour définir des grammaires. Un des principaux avantages est la possibilité de profiter d'une conversion bidirectionnelle entre texte et modèle. L'instanciation des métamodèles est ainsi simplifiée tout comme la maintenance de l'ensemble.

Proposer une telle grammaire qui paraphrase le modèle dans des termes et avec une syntaxe proches du langage naturel est une volonté qui répond au besoin des utilisateurs dans de nombreux cas pratiques, en particulier celui des journalistes chargés de saisir, compléter et valider un contenu de dépêche de presse relatant un événement donné et ses propriétés temporelles.

La grammaire produit un texte directement compréhensible par l'utilisateur (proche du langage naturel). Ce texte est l'écho de ce que le modèle va intégrer comme donnée et l'utilisateur peut ainsi le confronter à l'information qu'il souhaite enregistrer avant de valider la transaction. Cette phase de validation sémantique qui demande l'expertise de l'humain est complémentaire des vérifications syntaxique et sémantique (temporelle) automatiquement effectuées par le système en fonction des contraintes intrinsèques portées par le métamodèle et celles qui seront présentées au chapitre 5.

3.5.2 Mise en œuvre

La grammaire permet *de facto* de sélectionner les éléments instanciables du métamodèle en masquant certaines parties à l'utilisateur. Ceci nous a permis de définir plusieurs grammaires pour un même métamodèle, ainsi la grammaire de l'annexe E.1 concerne principalement la partie temporelle du métamodèle, alors que celle de l'annexe E.2 propose de décrire les relations structurantes entre événements (`structural`) en plus des propriétés temporelles. Nos grammaires sont pour l'heure en pseudo anglais, nous pourrions parfaitement les décliner en français et dans d'autres langues.

La figure 3.5.1 donne un exemple d'utilisation de la grammaire mettant en œuvre à la fois les relations `temporal` et `structural`. On y retrouve l'événement `Francofolies` qui porte la propriété temporelle « `1 times during each July` ». Ensuite on voit apparaître un événement `Francofolies2011` qui possède une période d'occurrence « `from "2011-07-12" to "2011-07-16"` ». L'information « `verifies the intention of Francofolies` » indique que l'événement `Francofolies2011` est une occurrence de `Francofolies` qui vérifie la propriété de récurrence portée par Francofolies *via* l'association `temporal`. De plus, nous ajoutons deux sous-événements à `Francofolies2011`, i.e. : `FeteA` (« *La fête à Xyz* ») et `JT` (« *SFR Jeunes Talents* »), par conséquent cela se traduit par l'instancia-

tion de la relation `structural` du métamodèle. Le terme `Concert` permet de qualifier la relation `structural` avec un label (cf. la classe `StructMetaData` en 3.4.1).

```
//Concept de Francofolies
The event "Francofolies" occurs periodically
    (named "Francofolies")
    according to the rule(s) below
    - rule: 1 times during each July
end of the event
//An instance of Francofolies
The event "Francofolie2011" occurs concretely
    [verifies the intention of Francofolies]
    from "2011-07-12" to "2011-07-16"
    structural as "Concert" {FeteA, JT}
end of the event

The event "FeteA" occurs periodically (named "FeteA")
    according to the rule(s) below
    - rule: 1 times during each day
end of the event

The event "JT" occurs periodically (named "JT")
    according to the rule(s) below
    - rule: 2 times during each day
end of the event
```

FIGURE 3.5.1: Exemple d'utilisation de la grammaire mettant en œuvre à la fois les relations `temporal` et `structural`

3.6 Connexion entre modèle d'événement et Modèle Temporel

3.6.1 Objectif

PTOM est un métamodèle ou encore un DSL pour les événéments composites à caractère périodique. Comme nous l'avons décrit dans l'état de l'art, UML propose de modéliser le temps avec le paquetage SIMPLETIME. Il est dit dans la spécification d'UML que lorsque le modèle du domaine doit utiliser des types liés au temps, alors il est préférable d'utiliser un modèle externe (comme PTOM) ou bien un profil UML comme MARTE qui ont été spécifiquement définis dans cet objectif. Nous montrons dans la suite de cette section comment nous pouvons relier des événements PTOM avec des classes et instances UML.

Dans la suite nous utiliserons le terme « d'instances » au lieu « d'*InstanceSpecification* du modèle UML » pour faciliter la lecture.

3.6.2 Mise en œuvre

Dans le métamodèle PTOM nous avons ajouté à la classe `T_Event` un attribut nommé `umlClass` qui est de type `uml::Class` qui permet de relier un événement PTOM à une classe d'un modèle UML. Comme nous l'avons vu précédemment nous considérons que les événements PTOM font référence à la fois à des classes UML ou bien à des instances. Ainsi afin de lier un événement PTOM avec une instance, nous avons ajouté, comme précédemment, à la classe `T_Event` un attribut `umlInstance` de type `uml::InstanceSpecification`. La liaison entre les deux modèles est navigable de PTOM vers le modèle du domaine ce qui permet de ne pas être intrusif dans le métamodèle d'UML. De plus, le besoin de navigation inverse n'a pas été exprimé.

La figure 3.6.1 montre l'utilisation de ces deux nouveaux attributs en réutilisant l'exemple précédent des Francofolies. À noter qu'il est nécessaire d'importer le fichier UML correspondant dans le fichier PTOM.

Sur la partie gauche de la figure se trouve le modèle UML du domaine et sur la partie droite le fichier PTOM. Les relations entre les deux sont symbolisées par des flèches uni-directionnelles. Ainsi la mise en correspondance s'exprime de la façon suivante :

- l'événement `Francofolies` est relié à la classe UML `Francofolies` ;
- l'événement `Francofolies_2011` est relié à l'instance `Francofolies_2011` ;
- l'événement `FeteA` est relié à l'instance `FeteA` ;
- l'événement `JT` est relié à l'instance `JT`.

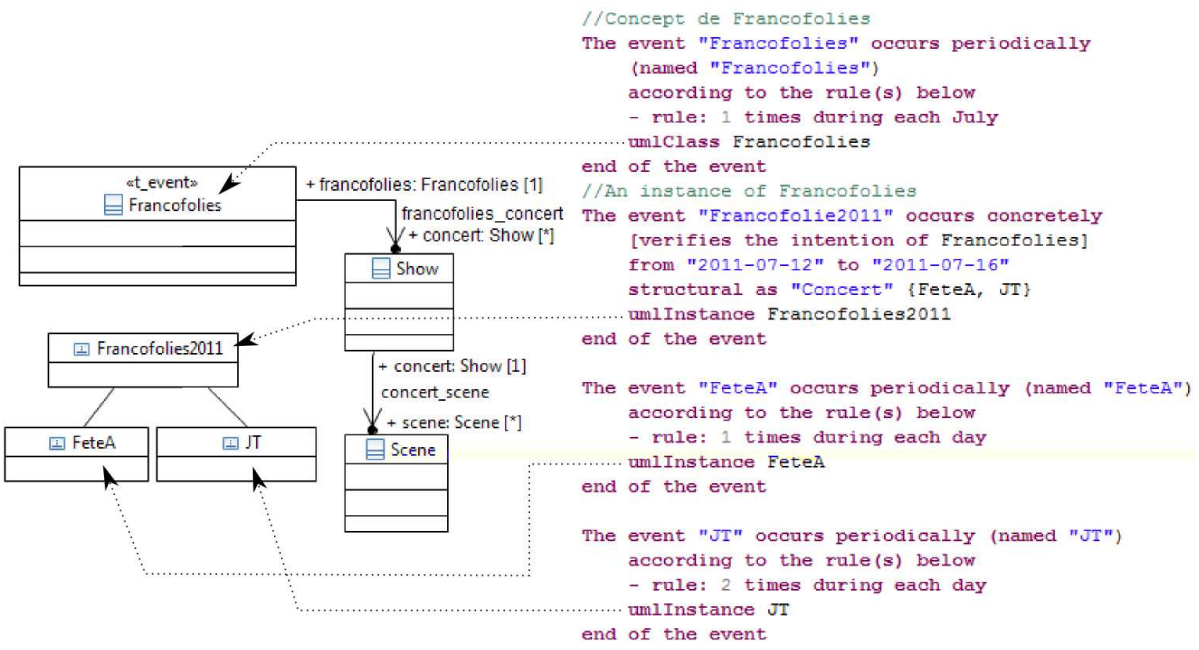


FIGURE 3.6.1: Grammaire d'événements composites et récurrents et liaison avec un modèle UML : exemple des Francofolies

3.7 Conclusion

Dans ce chapitre, nous avons structuré notre discours en présentant successivement deux types de métamodèles dont nous avons discuté les interactions. Le métamodèle d'événements composites et celui d'expressions temporelles en intension (`PTO`). Les événements sont décomposés de façon structurelle et temporelle. En conséquence, des contraintes d'intégrité entre les deux types de décomposition doivent être spécifiées et vérifiées.

Pour exprimer les aspects structurels, nous appliquons le patron composite. Pour traiter de la sémantique temporelle, nous nous fondons d'une part sur la norme ISO 19108, et d'autre part sur la sémantique du calendrier grégorien.

Nous avons présenté les différentes caractéristiques de Ptom et son pouvoir expressif (règles de périodicité, exceptions, positions relatives. . .). Nous avons également envisagé la traduction automatique d'expressions temporelles intensionnelles en leur contrepartie concrète, c'est-à-dire une suite de dates calendaires.

Selon Ptom, qui met volontairement en œuvre une approche unificatrice, toutes les expressions temporelles sont supposées représenter des phénomènes (pseudo) périodiques, en vertu du principe qu'un événement non périodique se décrivait comme une suite d'événements périodiques n'ayant qu'une occurrence.

Les contraintes liées à la norme, jointes aux extensions que nous spécifions dans Ptom et enfin la sémantique calendaire grégorienne sont les bases de la vérification syntaxique et sémantique préalable des données destinées à peupler le modèle.

Cette phase de vérification doit s'accompagner d'une phase de validation qui ne peut être effectuée sans l'aide d'un expert (s'assurer qu'un ensemble de données réputé correct pour les modèles et métamodèles représente bien ce que l'utilisateur voulait effectivement exprimer). Pour faciliter le dialogue avec l'expert, nous avons produit une grammaire formelle qui paraphrase le modèle.

Chapitre 4

Relations qualitatives entre intervalles non convexes : ALLEN*

Nous étudions des phénomènes périodiques et souhaitons définir des positions relatives entre les occurrences de ces phénomènes. Comme il a été dit précédemment, tous les phénomènes sont vus comme des événements et si un événement se répète dans le temps, nous fournissons les moyens pour définir sa périodicité. Dans certains cas, l'expression de cette périodicité est facilitée en utilisant des références à d'autres événements préalablement définis. Par exemple, on souhaite indiquer que « *chaque semaine, le mardi suit le lundi* ». Le verbe « *suit* » correspond à l'une des relations à spécifier. Notre problématique est de définir des relations topologiques homologues à celles d'Allen, mais entre des intervalles récurrents ou, autrement dit, non convexes. En particulier, la définition d'un calendrier utilise intensivement de telles relations qualitatives. Nous donnons dans le chapitre 5, une modélisation de la sémantique calendaire à l'aide des relations qualitatives d'Allen et de celles que nous allons définir entre intervalles non convexes.

4.1 Introduction

Les travaux d'Allen fournissent un cadre théorique fondamental pour le calcul sur les intervalles convexes. G. Ligozat a étendu ce travail dans le cas des intervalles non convexes [Ligozat 91, Ligozat 10]. La plupart des propriétés algébriques sont préservées. Malgré la clarté et la complétude des structures algébriques, combinatoires et logiques de l'ensemble des relations qualitatives entre intervalles non convexes, il y a très peu d'études qui se sont penchées sur la mise en œuvre de ces concepts. Pourtant, P.B. Ladkin avait à l'origine ressenti un tel besoin d'organisation et proposé une première classification des relations binaires entre intervalles non convexes [Ladkin 86].

Nous laissons volontairement de côté la somme importante de travaux menés pour spécifier des sous-algèbres calculables maximales. Nous renvoyons sur la question du raisonnement aux publications fondatrices en complément de celles que nous avons déjà évoquées dans un contexte plus général [Weida 92, Nebel 95, Drakengren 96] et le mémoire d'habilitation à diriger des recherches de A. Osmani [Osmani 12] qui présente une étude bibliographique et discute ces points en particulier en référence au concept de relation préconvexe définie par Ligozat.

Ces approches constituent une forme de taxonomie des relations élémentaires, mais sont conçues avec le souci de faciliter le raisonnement dans le cadre des recherches dans le domaine de la satisfiabilité, et non en rapport à l'objectif que nous nous sommes fixé (application aux Systèmes d'Information).

Contrairement aux treize relations d'Allen entre intervalles convexes qui portent une sémantique pertinente, il y a actuellement un manque d'abstraction lorsqu'on s'intéresse aux intervalles non convexes. Dans ce cas précis, l'ensemble des relations binaires ne présente pas nécessairement d'intérêt pratique, i.e. : elles n'ont pas toutes du sens dans la vie courante. En effet, le treillis des relations entre intervalles non convexes est trop important pour être maîtrisé et utilisé comme tel dans son intégralité dans des applications pratiques.

Comme nous l'avons dit dans l'état de l'art, Ladkin [Ladkin 87] a listé des relations d'intérêt pour spécifier des aspects comportementaux de processus concurrents, mais à notre connaissance, hormis quelques approches spécifiques à un domaine particulier, aucune étude générale n'a été effectuée. Ainsi, nous souhaitons spécifier un noyau de types de relations entre intervalles non convexes, dont les éléments expriment des concepts courants et largement usités par exemple lors de la gestion d'occurrences d'événements temporels quel que soit le domaine d'application.

Dans notre cas, nous sélectionnons un ensemble de types de relations étendues pertinentes (nommé $ALLEN^*$) lesquelles peuvent être appliquées à des intervalles non convexes tout en gardant une sémantique proche de celle d'Allen qui constituent une base solide et universellement reconnue. Nous utilisons ensuite les travaux théoriques de G. Ligozat qui concernent les relations étendues afin d'étudier les propriétés fondamentales du noyau que nous souhaitons spécifier.

Ce chapitre s'organise comme suit. La section 4.2 rappelle les formalismes d'Allen puis (section 4.3) ceux de Ligozat pour le calcul sur les intervalles généralisés. La section 4.4 liste les types de relations significatives choisies pour les intervalles non convexes constituant le noyau de notre approche. La section 4.5 est consacrée à la présentation des propriétés principales du noyau. Nous étudions la question importante du calcul de la composition des relations étendues en section 4.6, dans laquelle nous donnons un algorithme à cet effet. La section 4.7 est dédiée à la spécification d'opérateurs, dits « de filtrage », qui permettent dans un but opératoire de spécialiser les éléments d'$ALLEN^*$. Enfin, avant de conclure, la section 4.8 montre le lien entre les relations étendues et le métamodèle Ptom. Ce travail repose sur la théorie de Ligozat sur les relations étendues et des propriétés spécifiques sont énoncées pour l'ensemble $ALLEN^*$.

4.2 Algèbre d'intervalle d'Allen

La topologie du temps a pour objectif de positionner des points et des intervalles les uns par autres et d'en déduire des relations [Vilain 82]. Dans cette section nous rappelons les principaux éléments, définitions et propriétés, de l'algèbre des intervalles d'Allen [Allen 81, Allen 83]. Puis, nous proposons un raccourci de la notation, pour nommer les relations d'Allen, qui sera utilisé dans la suite de ce manuscrit. Ensuite nous abordons les intervalles non convexes. À la fin de la section, nous introduisons une règle de codage qui s'appuie sur les travaux de Ligozat qui ont servi à étendre la proposition d'Allen au cas des intervalles non convexes.

4.2.1 Définition

Dans le sens d'Allen, un intervalle convexe est défini par un couple d'éléments ordonnés distincts dans le domain ($\mathcal{D}$) d'un ordre total ($\leq$) ; d'où sa mise en œuvre relativement simple dans le domaine des intervalles temporels.

Plus précisément, un intervalle donné I est équivalent à un couple (a, b) tel que : $(a, b \in \mathcal{D} \ \wedge \ a < b)$.

Dans la suite, $(a < b)$ est un substitut pour $(a \leq b \ \wedge \ a \neq b)$.

Ainsi : $I = \{\, x \in \mathcal{D} \mid a \leq x \ \wedge \ x \leq b \,\}$

a et b sont respectivement les bornes inférieure et supérieure de l'intervalle $I = (a, b)$

Soit $I[\mathcal{D}, \leq]$ l'ensemble de ces intervalles.

Après avoir combiné toutes les positions relatives possibles entre les bornes (suivant l'ordre $\leq$, i.e. : $<$, $=$, $>$), ALLEN a identifié 13 types de relations mutuelles élémentaires entre deux intervalles convexes.

4.2.2 Terminologie et notation

Dans la littérature, plusieurs systèmes d'abréviation sont utilisés pour noter les 13 relations, nous adoptons la suivante :

$precedes$	(p) (P)	$Preceded\_by$
$meets$	(m) (M)	Met_by
$overlaps$	(o) (O)	$Overlapped\_by$
$finishes$	(f) (F)	$Finished\_by$
$during$	(d) (D)	$Contains$
$begins$	(b) (B)	$Begun\_by$
$equals$	(e)	

Chaque relation est définie par des contraintes (impliquant l'ordre $<$) sur les bornes des intervalles en question.

Ainsi, l'ensemble exhaustif des relations possibles est :

$$ALLEN = \{\, p,\ P,\ m,\ M,\ o,\ O,\ f,\ F,\ d,\ D,\ b,\ B,\ e \,\}$$

A l'exception de la relation *equals* qui est réflexive, la casse (minuscule ou majuscule) de la première lettre de chaque relation indique respectivement une relation et son inverse.

4.2.3 Représentation canonique

La relation mutuelle entre deux intervalles peut être représentée de manière concise à l'aide de correspondances entre l'ensemble des bornes des intervalles et l'ensemble des entiers naturels comme présenté ci-dessous.

Soit $X = (x_1, x_2)$ et $Y = (y_1, y_2)$, deux intervalles vérifiant $X\ R\ Y$ où $R \in ALLEN$. Supposons que les bornes de Y soient respectivement associées aux nombres impairs 1 et

3. Il en résulte la définition de cinq zones adjacentes incluant des singletons $\{ T_0, \ldots, T_4 \}$, comme suit :

$$\begin{aligned} T_0 &= \{ t \in \mathcal{D} \mid t < y_1 \} & T_1 &= \{ y_1 \} \\ T_2 &= \{ t \in \mathcal{D} \mid y1 < t \wedge t < y_2 \} & T_3 &= \{ y_2 \} \\ T_4 &= \{ t \in \mathcal{D} \mid y_2 < t \} \end{aligned}$$

Toute relation entre X et Y peut être représentée par un couple composé des deux rangs repérant respectivement les zones dans lesquelles se trouvent x_1 et x_2.

Par exemple (cf. figure 4.2.1), *precedes* est codée par $(0, 0)$, *Preceded_by* correspond à $(4, 4)$, *begins* à $(1, 2)$ et $(0, 1)$ signifie *meets*, etc. Il est à noter que l'égalité des deux éléments du codage peut seulement avoir lieu pour des entiers pairs, mais jamais pour des impairs. En effet, les intervalles vides sont interdits dans la théorie d'Allen.

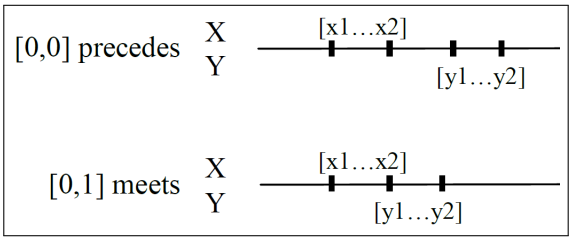


Figure 4.2.1: Exemples de relations d'Allen avec le codage de Ligozat

4.2.4 Les propriétés principales de l'algèbre d'Allen

Cette section traite de la théorie des ensembles, de la combinatoire et des aspects relationnels des propriétés de l'algèbre d'Allen.

Théorie des ensembles

Nous identifions l'ensemble $ALLEN$ avec un ensemble de parties du produit cartésien de l'ensemble des intervalles, à savoir :

$$POW(I[\mathcal{D}, \leq] \times I[\mathcal{D}, \leq])$$

Les 13 relations d'Allen définissent une partition : en conséquence tout couple d'intervalles pris dans $I[\mathcal{D}, \leq]$ satisfait une et une seule relation dans $ALLEN$.

En outre, $ALLEN$ génère une algèbre : $POW(ALLEN)$, et divers prédicats peuvent être définis comme des disjonctions d'éléments atomiques d'$ALLEN$. Ceci engendre des relations qualitatives non élémentaires intéressantes telles que « *intercept* », « *adjacent* », etc.

Propriétés combinatoires

Un ordre partiel entre les relations peut être définie sur $ALLEN$, d'où la possibilité de munir cet ensemble d'une structure de treillis distributif. La représentation sous forme de diagramme de Hasse donnée en figure 4.2.2 est la plus appropriée pour définir ce treillis, structure sur laquelle d'autres définitions peuvent s'appuyer. Cette représentation est également appelée diagramme conceptuel des voisins de Freksa [Freksa 97].

À partir d'une configuration donnée entre deux intervalles X et Y, il apparaît clairement, à condition que cela soit permis, que déplacer une borne de X vers la gauche ou vers la droite (i.e. de la zone d'origine où est défini le précédent ou le suivant), modifie nécessairement la relation mutuelle entre X et Y. Sur cette base, on peut observer qu'une topologie émerge sur les diverses configurations entre deux intervalles, par exemple : $X = (x_1, x_2)$ et $Y = (y_1, y_2)$. Ici, le déplacement de la borne « d'un pas » correspond à une sorte de continuité.

En effet, soit R appartenant à $ALLEN$. R a une représentation canonique (i, j), avec :

$$(i, j \in 0..4 \ \wedge \ i \leq j \ \wedge \ (i = j \Rightarrow even(i)))$$

Lorsque$(i < 3 \ \wedge \ j > 1)$, R a deux voisins directs, à savoir $(i, \ j + 1)$ et $(i + 1, \ j)$, il n'en a qu'un seul dans le cas contraire (cf. figure 4.2.2).

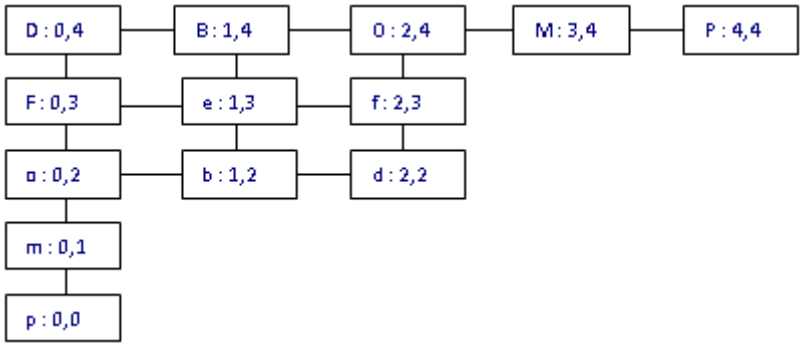


Figure 4.2.2: Structure de treillis des relations mutuelles entre intervalles d'Allen (Diagramme de Hasse)

De manière évidente, l'ordre $(<)$ qui est sous-jacent à la structure du treillis distributif, est l'ordre produit sur chacune des coordonnées de la représentation canonique.

Supposons que : $X = (i_x, \ j_x) \ \wedge \ Y = (i_y, \ j_y)$ alors :$(X < Y \ \Leftrightarrow \ i_x \leq i_y \ \wedge \ j_x \leq j_y)$

Aspects relationnels

Considéré comme un ensemble de relations binaires, $ALLEN$ est stable pour l'inversion et la composition relationnelle. L'inversion est triviale et peut être traduite en conservant une même dénomination en distinguant la relation directe de son inverse

grâce à l'usage de minuscules et de majuscules. L'étude de la composition de deux relations (notée par « ; ») conduit à la construction de tables de transitivité qui trouvent leur utilité pour le raisonnement et en ingénierie des connaissances. La composition est définie comme suit :

$$\forall\, R_1,\, R_2,\, R_3 \in ALLEN \mid R_3 = R_1\ ;\ R_2\ \bullet$$
$$R_3 = \{X,\, Y,\, Z\ :\ I[\mathcal{D}, \leq] \mid X\ R_1\ Y\ \wedge\ Y\ R_2\ Z\ \bullet\ (X,\, Z)\}$$

Une importante propriété de la composition est son indéterminisme, en effet $R\ ;\ S$ dépend de la configuration géométrique spécifique (non qualitative) des deux intervalles considérés.

Par exemple, à partir de la seule information ($X\ B\ Y\ \wedge\ Y\ m\ Z$), on ne peut pas décider entre ($X\ D\ Z$ *ou* $X\ F\ Z$ *ou* $X\ o\ Z$).

Au contraire, ($X\ s\ Y\ \wedge\ Y\ m\ Z$) implique ($X\ p\ Z$).

Cependant, l'ensemble de tous les résultats possibles pour une composition donnée est strictement contraint et nécessairement constitué d'un intervalle [1] dans le treillis.

Les propriétés de transitivité sont la base du calcul de satisfiabilité (*SAT problem*) d'un ensemble de contraintes pour une configuration d'intervalles d'Allen. Ce thème est abordé par plusieurs études dans le domaine des problèmes SAT, mais ces aspects restent hors du champs de cette thèse.

Le tableau 4.1 exprime la transitivité des relations d'Allen.

.	p	P	b	B	d	D	f	F	m	M	o	O	e
p	p	pP	p	p	pd	p	pd	p	p	pd	p	pd	p
P	pP	P	dP	P	dP	P	P	P	dP	P	dP	P	P
b	p	P	b	bB	D	pD	d	po	p	M	po	dO	b
B	pD	P	bB	B	dO	D	O	D	oD	M	oD	O	B
d	p	P	d	dP	D	pP	d	pd	p	P	pd	dP	d
D	pD	DP	oD	D	oO	D	DO	D	oD	DO	oD	DO	D
f	p	P	d	OP	D	DP	f	Ff	m	P	od	OP	f
F	p	DP	o	D	Od	D	Ff	F	m	DO	o	DO	F
m	p	DP	m	m	Od	p	od	p	p	Ff	p	od	m
M	pD	P	dO	P	dO	P	M	M	bB	P	dO	P	M
o	p	DP	o	oD	Od	pD	od	po	p	DO	po	oO	o
O	pD	P	dO	OP	dO	DP	O	DO	oD	P	oO	OP	O
e	p	P	b	B	d	D	f	F	m	M	o	O	e

Tableau 4.1: Table de transitivité d'Allen

1. Quelque soient 2 éléments comparables x et y dans le treillis, tels que $x < y$, l'intervalle $[x, y]$ consiste en tous les éléments z vérifiant ($x \leq z\ \wedge\ z \leq y$)

Pour des questions de concision, seules les bornes des intervalles du treillis ont été mentionnées. L'intervalle complet est explicitement visible sur la figure 4.2.2. Les cellules avec un fond gris sont des singletons.

4.3 Calcul sur les intervalles généralisés

La proposition d'Allen a donné lieu à diverses extensions. La contribution de M.B. Vilain [Vilain 82] consiste principalement à étendre dans un premier temps la portée de l'étude aux relations mutuelles entre des points, puis ensuite entre des intervalles convexes. Il n'est pas possible de considérer des systèmes mélangeant à la fois des points et des intervalles sans devoir ajouter de nouveaux opérateurs indiquant le type d'objets manipulés ce qui rend complexe et lourd la gestion d'une telle mixité. L'extension de Villain préserve ou adapte toutes les propriétés algébriques et relationnelles d'Allen.

4.3.1 Intervalles non convexes *vs* patron de points linéaires

Ligozat traite des relations mutuelles entre des ensembles de points fournis avec une relation d'ordre total. La dissymétrie de l'approche apparaît clairement : le second ensemble de points sépare l'espace linéaire en une série de zones adjacentes, alternativement de dimension 1 (intervalles) ou 0 (points), et constitue un système de référence. Puis, tous les membres du premier ensemble de points sont étiquetés par le rang de la zone dans laquelle ils se trouvent. Ceci est une généralisation de la technique d'étiquetage introduite en 4.2.3.

Ce passage d'un point de vue géométrique au qualitatif, donne une connaissance plus abstraite sur la situation en s'affranchissant de la valeur numérique des coordonnées. De manière similaire, le passage de systèmes de points à des systèmes d'intervalles permet un gain en abstraction.

Lorsque la taille de la série de points est paire, il y a une correspondance évidente entre une série de points et des intervalles non convexes.

Dans la suite de cette section, l'ensemble des intervalles non convexes à p élements sera noté par $NCI(p)$. NCI indique l'union de $NCI(p)$ sur toutes les valeurs p. Tout élément dans NCI est ainsi un ensemble d'intervalles convexes disjoints (dans le sens d'Allen) :

$$NCI = \{\, \Xi \in POW(I[D, \leq]) \mid (\forall X, Y \in \Xi \mid not(X\ e\ Y) \bullet X\ p\ Y\ \vee\ X\ P\ Y) \,\}$$

Par conséquent, l'ensemble des patrons de points de taille n sera noté $PP(n)$. Ainsi, il y a une correspondance directe entre les patrons de points et d'intervalles lesquels se notent $NCI(p)$ avec $PP(2p)$ quelque soit p dans $NAT1$. Nous nous intéressons en fait au traitement des patrons d'intervalles non convexes mais, dans une approche générale,

toutes les propriétés représentent des patrons de points et seront présentés comme tels dans la suite.

4.3.2 Extensions de Ligozat pour l'algèbre d'intervalle d'Allen

Ligozat a construit une théorie générale qui étend les travaux d'Allen et de Vilain à tout type de configuration linéaire de points. Nous donnons tout d'abord quelques définitions, puis nous montrons comment les travaux d'Allen sont étendus d'une manière naturelle avant de traiter le cas des intervalles non convexes dans sa généralité.

4.3.2.1 Principe de base et notation

Soit X (respectivement Y) un système de points de taille m (respectivement de taille n) tel que : $X = (x_1, ..., x_m)$ et $Y = (y_1, ..., y_n)$. Les éléments de Y sont associés à une section commençante de l'ensemble des nombres impairs : $(1, 3, \ldots, 2n-1)$, ceci amène à définir $(2n+1)$ zones adjacentes, qui sont étiquetées de 0 à $2n$ (fonction notée $label()$). X est alors représenté par la série d'étiquettes indiquant la zone correspondant à ces éléments, suivant la règle suivante :

$$\begin{array}{rc} (\quad \forall i : 1..m \bullet label(x_i) = 2k-1 \Leftrightarrow x_i = y_k & \wedge \\ label(x_i) = 0 \Leftrightarrow x_i \leq y_1 & \wedge \\ label(x_i) = 2n \Leftrightarrow x_i \geq y_n & \wedge \\ label(x_i) = 2k \Leftrightarrow x_i \in]y_k/2\ ,\ 1 + y_k/2[&) \end{array}$$

Dans ce contexte, la représentation de X est une séquence non décroissante π de m nombres, choisis entre 0 et $2n$, les doublons ne sont pas autorisés dans la séquence pour les valeurs impaires. Pour n fixé, π identifie de façon unique une configuration, et chaque configuration possède un code π. Dans l'exemple précédent, la configuration suivante (cf. figure 4.3.1), où $m = 5$ et $n = 4$ sera codée (2, 5, 6, 6, 8).

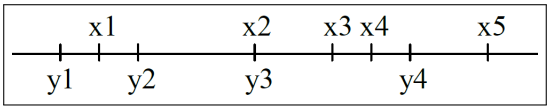


Figure 4.3.1: Configuration $\pi = (2, 5, 6, 6, 8)$ dans $\Pi(5, 4)$

Dans la suite, l'ensemble des relations possibles entre X et Y dépend seulement de m et n, et sera alors désigné par : $\Pi(m, n)$. Dans l'exemple précédent, $\pi = (2, 5, 6, 6, 8)$ est un élément de $\Pi(5, 4)$. Enfin, Π représentera l'union des $\Pi(m, n)$, pour toutes les valeurs possibles de m et n. Désormais, nous admettrons l'utilisation du codage π, de la même manière que nous l'avons fait avec les intervalles convexes, pour indiquer une relation spécifique entre deux intervalles non convexes ; ainsi nous écrirons $X\ \pi\ Y$ pour

deux intervalles dans NCI au lieu de $X\ R\ Y$ (en supposant que R soit effectivement codé par π). Comme π est une séquence d'entiers naturels, $\pi(i)$ représentera alors le $i^{ème}$ élément de π.

4.3.2.2 Propriétés

Structure du treillis

La généralisation de LIGOZAT préserve la plupart des propriétés combinatoires précédemment observées dans la théorie d'ALLEN. Plus précisément, pour tout entier p et q, $\Pi(p, q)$ est muni d'une structure de treillis distributif. L'ordre sous-jacent au treillis est homologue à celui du cas simple d'ALLEN et correspond à l'ordre produit sur les entiers naturels apparaissant dans le codage des configurations.

Inversion *vs* transposition

Du fait de la dissymétrie évoquée ci-dessus lorsque $p \neq q$, l'inversion n'est pas appropriée pour traiter des relations mutuelles entre intervalles non convexes. En revanche, la transposition est utile pour intervertir les rôles des deux intervalles associés dans une relation binaire. La définition de la transposition est la suivante :

$$\forall m,\ n\ :\ NAT1\ ;\ R \in \Pi(m,n)\ \bullet$$
$$\exists_1\ R^t \in \Pi(n,m)\ \bullet\ (\forall X \in NCI(m)\ ;\ Y \in NCI(n)\ \bullet\ X\ R\ Y \Leftrightarrow Y\ R^t\ X)$$

($\exists_1$ indique l'existence unique)

Concernant le codage de LIGOZAT, la transposée de π est notée π^t. Dans [Ligozat 91], il donne une formule pour calculer π^t à partir de π. Il apparaît clairement, selon la définition, que tous les éléments de $\Pi(p, q)$ ont une unique transposée et que l'opération de transposition est une involution.

Composition

La composition $\pi_1\ ;\ \pi_2$ de deux éléments respectivement membres de $\Pi(m_1, n_1)$ et $\Pi(m_2, n_2)$ est possible dès que $n_1 = m_2$. Le résultat peut être indéterministe (lorsque seule l'information qualitative est spécifiée) et fournit un ensemble d'éléments de $\Pi(m_1, n_2)$ qui constituent nécessairement un intervalle dans le treillis correspondant. LIGOZAT met en avant des propriétés intéressantes : la transposition est une bijection qui inverse l'ordre initial du treillis et transposer le résultat de la composition de deux éléments donnés est équivalent à la composition de leurs transposés.

Théorie des ensembles

Selon la théorie des ensembles, il peut apparaître tentant de développer un schéma similaire à Allen. Malheureusement, la taille de Π croît de manière exponentielle avec p et q. En outre, peu de relations atomiques sont d'un intérêt pratique. En effet, les cas où la relation mutuelle entre les composants d'intervalles convexes est trop hétérogène s'avère complexe et souffre d'un manque patent d'abstraction. Dans de telles situations, il est utile de diviser l'ensemble d'intervalles non convexes de manière à se référer pour chaque partie à des relations plus simples.

La section suivante a pour but l'étude des propriétés utiles en pratique et qui résultent de la spécification d'un ensemble de types de relations élémentaires significatives tout en gardant un niveau de généralité suffisant.

4.4 Ensemble de relations synthétiques

4.4.1 Objectifs et motivations

L'objectif est de spécifier un ensemble « simple » et significatif de concepts en typant les relations entre intervalles non convexes. Le terme « simple » signifie que de tels concepts correspondent à des situations communes fréquemment rencontrées par des utilisateurs. Dans [Dubois 00], les auteurs cherchent une partition de l'ensemble des relations convexes (du treillis) pour des intervalles convexes. Leur but est de simplifier la complexité des problèmes de satisfiabilité. Ils créent de nouveaux concepts (spectres et patrons) avec leur propre sémantique. Nous cherchons également des archétypes dans l'ensemble des relations, mais opérons sur des intervalles non convexes. Les solutions de Dubois *et al* ne sont pas applicables. Nous sacrifions le côté « partition » (nos relations ne sont pas exclusives et il y a des relations non exploitables, cf. section 4.5) et également le côté « convexe » (nos relations ne le sont pas), mais nous privilégions la sémantique initiale d'Allen qui est au mieux conservée.

De manière à illustrer notre propos, nous nous réfèrerons au cours de la présentation de notre travail à un cas d'utilisation type qui concerne la constitution, la gestion, et l'interrogation d'une base de connaissance dédiée à l'enregistrement d'occurrences d'événements et de disponibilité de ressources.

Le système de Ligozat fournit un cadre de développement très efficace pour indexer de telles informations grâce à son calcul de composition. Il reste à réaliser le choix des sous-ensembles réduits fondamentaux et facilement calculables de Π et la constitution d'une base opérationnelle pour exprimer et gérer la connaissance et les processus de l'utilisateur final. Bien sûr, le cas simple des intervalles convexes, que traite Allen, inclut clairement les concepts de première importance qui doivent nécessairement être considérés et adaptés pour le traitement des intervalles non convexes.

Nous préconisons tout d'abord un principe fondateur auquel nous nous référons lors de l'extension des relations d'Allen aux intervalles non convexes. Ainsi, chaque relation

étendue dans notre proposition est spécifiée de deux manières équivalentes :

- par un ensemble de contraintes imposées sur des couples de composants convexes pour chaque intervalle non convexe impliqué dans la relation étendue. Ces contraintes sont des prédicats qui se réfèrent uniquement aux relations classiques d'ALLEN.
- en contraignant directement le codage de LIGOZAT pour les intervalles non convexes manipulés.

Ensuite nous discutons les propriétés de ces extensions, le traitement de la composition et d'autres aspects tels que le filtrage des intervalles non convexes. Ce dernier point permet de reconsidérer des relations complexes en des plus simples.

4.4.2 Principe de base pour étendre les relations d'Allen aux intervalles non convexes

Nous sommes guidés à la fois par le souhait de profiter de l'expertise des utilisateurs finaux et par les propriétés optimales des relations d'ALLEN. Notre approche vise à fournir une sémantique cohérente des relations d'ALLEN dans le cadre des intervalles non convexes. Ainsi ce que nous proposons peut être vu comme une extension de la terminologie d'ALLEN. Une attitude naïve serait de construire cette extension en imposant une bijection entre les composants de deux intervalles non convexes considérés, puis d'affirmer que chaque couple d'éléments ainsi constitué vérifie uniformément une même relation d'ALLEN donnée. Ceci est trop restrictif, et ne pourrait s'appliquer sauf dans des cas triviaux imposant, entre autres, la condition $(p = q)$.

Notre proposition consiste en une relaxation des contraintes naïves évoquées ci-dessus, imposant, dans le cas général, une injection (voire une simple relation binaire) entre X et Y et non plus une bijection. Plus précisément, pour définir l'extension R^* d'une relation d'ALLEN R donnée, nous mettons en avant le premier intervalle non convexe (cf. : dissymétrie déjà présente dans le codage de LIGOZAT), et assurons que pour chacun de ses composants convexes (e.g. : i_1), il existe au moins un composant convexe dans le second intervalle non convexe (e.g. : i_2) tel que $(i_1 R\ i_2)$ soit satisfait. Notre interprétation fondamentale est que R reste uniformément vraie pour tous les composants convexes du premier intervalle non convexe. Par contre, il peut exister dans le second des composants convexes non concernés par la relation étudiée. Ce point sera discuté dans le paragraphe suivant.

Auparavant, à titre d'exemple, nous donnons en figure 4.4.1 deux instances du type $meets^*$ dans les cas $p = 2$ et $q = 3$.

Ce genre de connaissance est en fait suffisante dans de nombreux cas d'utilisation. Par exemple lors de l'interrogation d'une base de connaissance recherchant les activités auxquelles un utilisateur peut participer. Le premier intervalle non convexe indique les disponibilités de l'utilisateur, et le second représente la période d'accès de l'activité (répétitive). Si une ou plusieurs périodes de l'un et l'autre se recouvrent, peu importe qu'il existe des périodes d'accès supplémentaires qui ne correspondent pas aux disponibilités

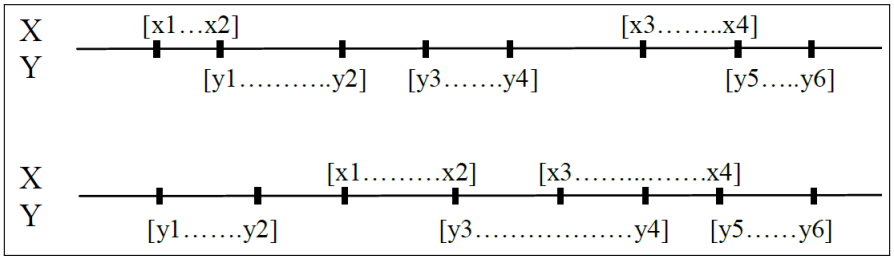


FIGURE 4.4.1: Deux cas spécifiques où X et Y satisfont $X\ meets^*\ Y$

de l'utilisateur.

Dans n'importe quel autre cas, des contraintes additionnelles peuvent être spécifiées, particulièrement en permutant les rôles des deux intervalles non convexes considérés, ce qui autorise l'expression de propriétés plus restrictives, et permet par exemple de reformuler le cas naïf évoqué plus haut, pour lequel à la fois le premier et le second intervalles non convexes sont strictement contraints.

On notera enfin que l'application du principe énoncé plus haut et que nous allons mettre en œuvre ci-après, préserve la stricte coïncidence des relations d'ALLEN avec leur extensions ALLEN* lorsque l'on se limite au traitement des relations entre couples d'intervalles convexes (ou de points).

4.5 Relations élémentaires étendues entre intervalles non convexes : ALLEN*

Cette section énumère l'ensemble des extensions proposées, puis énonce leurs principales propriétés. Nous spécifierons tout d'abord le contexte de la proposition de façon plusPacrmelle. Toutes les relations d'ALLEN s'appliquent sur le produit cartésien $I[\mathcal{D}, \leq] \times I[\mathcal{D}, \leq]$, et retournent une valeur dans $ALLEN$. L'ensemble des intervalles non convexes (NCI) peut s'exprimer comme suit :

$$NCI \in POW(I[\mathcal{D}, \leq]) \ \wedge\ \forall\, X \in NCI\ ;\ i_1, i_2 \in I[\mathcal{D}, \leq] \mid i_1 \in X\ \wedge\ i_2 \in X\ \wedge\ i_1 \neq i_2 \bullet i_1\, p\, i_2\ \vee\ i_1\, P\, i_2$$

Chaque élément de $ALLEN$ est une abstraction d'une série d'instances d'intervalles convexes appariés, toutes les instances partageant la même relation qualitative. La spécification d'$ALLEN^*$, permet de gagner en abstraction. Il n'est pas suffisant de reproduire la définition qui affirme que « chaque type de relation d'$ALLEN^*$ est une abstraction pour une série de couples d'instances d'intervalle non convexes couplées, tous les couples partageant la même relation étendue qualitative ». En effet, le niveau d'abstraction approprié n'est plus une relation étendue, mais un ensemble de telles relations.

Ainsi, les éléments d'$ALLEN^*$ sont des ensembles (cohérents) de relations étendues conformes à : $ALLEN^* \subseteq POW(\Pi)$.

Pour la suite, il faudra garder à l'esprit qu'en spécifiant $(X\ R^*\ Y)$, nous signifions qu'il existe un élément R dans Π tel que R soit de type R^* et que $(X\ R\ Y)$ soit vérifié. $ALLEN^*$ ne recouvre pas totalement Π. De nombreuses relations étendues de Π ne correspondent à aucune des relations de $ALLEN^*$. Nous appelons « *Garbage* » l'ensemble de telles relations ne faisant pas parties d'$ALLEN^*$. Le fait que par nature, la sémantique de ce genre d'éléments ne puisse s'exprimer simplement ou naturellement, justifie cette dénomination. Par exemple (cf. figure 4.5.1), la configuration $(0, 0, 4, 6, 6, 9)$ est considérée comme *Garbage* dans $\Pi(6, 6)$.

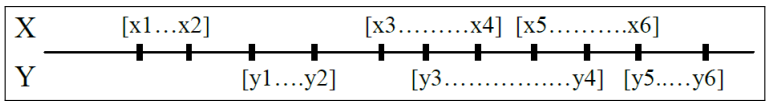


FIGURE 4.5.1: Un exemple de *Garbage* en $\Pi(6, 6)$: $(X, Y) \notin ALLEN^*$

4.5.1 Les relations ALLEN* : contexte et remarques générales

Ici nous traitons successivement des extensions que nous spécifions lorsqu'on applique le principe de base énoncé ci-dessus à chaque élément d'$ALLEN$. Le processus de généralisation conduit à étudier la relation mutuelle entre intervalles non convexes. Pour cela, nous avons besoin d'une définition préalable :

Soit X (respectivement Y) un intervalle non convexe constitué de p (respectivement q) composants convexes, par conséquent : $X \in NCI(p)$ et $Y \in NCI(q)$

Dans la suite, nous supposons que $inc\_seq(p, q)$ représente l'ensemble des séquences strictement croissantes de p entiers naturels non nuls inférieurs ou égaux à q :

$$\forall\, p, q\ :\ NAT1 \mid p \leq q \bullet inc\_seq(p, q) = \{u : seq_{1..q} \mid size(u) = p \wedge \forall\, i \in dom(front(u)) \bullet u(i) < u(i+1)\} \tag{4.5.1}$$

Ces séquences permettront de spécifier une relation canonique entre les composants convexes de X et ceux de Y comme cela a été implicitement évoqué dans la section 4.4.2. Sauf exceptions (qui seront mentionnées en détail en section 4.5.2), la relation canonique est une injection monotone au regard de la relation d'ALLEN *precedes*. Le domaine de u se réfère aux rangs des composants convexes de X, alors que son codomaine se réfère aux rangs des composants convexes de Y.

Par exemple, dans la figure 4.4.1 les séquences correspondant aux parties hautes et basses sont respectivement $< 1, 3 >$ et $< 2, 3 >$. Dans l'exemple du haut (respectivement celui du bas), les composants #2 (respectivement #1) sont hors du codomaine de la séquence.

Du fait des correspondances sous-jacentes au principe de base (cf. section 4.4.2), la plupart des définitions des relations étendues $ALLEN^*$ supposent une contrainte de cardinalité, à savoir $p \leq q$. Une conséquence est qu'ALLEN identifie six relations fondamentales, chacune accompagnée de son homologue inverse, plus une relation symétrique (e.g. : *equals*), tandis qu'$ALLEN^*$ spécifie quatorze relations individuelles (i.e. : 14 types de relation étendues).

La relation *equals* provenant d'ALLEN peut évidemment n'être obtenue que lorsque $p = q$. Par conséquent, lors du processus de généralisation, *equals* doit être séparée en ses contre-parties, i.e. : $subset^*$ lorsque $p \leq q$ et $Superset^*$ lorsque $p \geq q$. Ainsi, la traduction directe d'$equals^*$ peut s'exprimer par la conjonction : $equals^* = (subset^* \wedge Superset^*)$. De même, $precedes^*$ et $Preceded\_by^*$ doivent être spécialisées de manière à s'adapter au schéma étendu, ce qui nous a conduit à la spécification de la relation nommée : $interleaves^*$.

Sauf pour $subset^*$, $Superset^*$ et $interleaves^*$ qui sont nouvelles, tous les autres types de relations étendues sont nommés par l'adjonction d'une étoile à la dénomination originale d'ALLEN. Ainsi pour aider la compréhension tout au long du texte qui suit, nous noterons par $ALLEN^*[R_1]$ l'extension R_1^* de R_1 et inversement $ALLEN[R_1^*]$ la relation de base R_1 étendue par R_1^*. Ce point de vue n'est que lexical.

Nous devrons utiliser les crochets lorsque le paramètre est un ensemble de relations au lieu d'un singleton. La valeur alors retournée est l'union des images, par exemple :

$$ALLEN[\{R_1^*, R_2^*\}] = ALLEN(R_1^*) \cup ALLEN(R_2^*) \text{ et}$$
$$ALLEN^*[\{R_1, R_2\}] = ALLEN^*(R_1) \cup ALLEN^*(R_2)$$

4.5.2 Relations d'ALLEN* étendues : définitions

La liste des relations étendues est donnée ci-dessous. Selon la structure de treillis attachée aux représentations de LIGOZAT, il est possible, pour n'importe quel p et q donné, de déterminer l'élément minimal et maximal (dans le treillis) pour chaque définition. Ces éléments extrémaux seront systématiquement spécifiés ainsi que le type de relation canonique induite par la relation étendue traitée.

Définition 1 : $begins^*$ **/ code : b* (précondition :** $p \leq q$**)**
$X\ begins^*\ Y \Leftrightarrow \forall\, x \in X \bullet \exists\, y \in Y \bullet x\ begins\ y$
– Contrainte selon le formalisme de LIGOZAT :

$$\forall\, \pi \in \Pi \mid X\ begins^*\ Y \wedge X\, \pi\, Y \bullet \exists\, s \in inc\_seq(p,q) \bullet$$
$$(\forall\, i \in 1..2p \mid EVEN(i) \bullet \pi(i) = 4s(i/2) - 2 \wedge \pi(i-1) = \pi(i) - 1)$$

– Elément minimal : $\forall\, i \in 1..2p \mid EVEN(i) \bullet \pi(i) = 2i - 2 \wedge \pi(i-1) = \pi(i) - 1$
– Elément maximal : $\forall i \in 1..2p \mid EVEN(i) \bullet \pi(i) = 4q - 4p + 2i - 2 \wedge \pi(i-1) = \pi(i) - 1$
– Relation canonique : injection totale de X vers Y.

Définition 1a : $Begun\_by^*$ **/ code : B* (précondition :** $p \leq q$**)**
$X\ Begun\_by^*\ Y \Leftrightarrow \forall\, x \in X\ \bullet\ \exists\, y \in Y\ \bullet\ y\ Begun\_by\ x$
– Contrainte selon le formalisme de LIGOZAT :

$$\forall\, \pi \in \Pi \mid X\ Begun\_by^*\ Y\ \wedge\ X\pi Y\ \bullet\ \exists\, s \in inc\_seq(p,q)\ \bullet$$
$$(\forall\, i \in 1..2p \mid EVEN(i)\ \bullet\ \pi(i-1) = 4s(i/2)\text{–}3\ \wedge\ \pi(i) \geq \pi(i-1)+3)$$

– Elément minimal : $\forall\, i \in 1..2p \mid EVEN(i)\ \bullet\ \pi(i) = 2i\ \wedge \pi(i-1) = \pi(i) - 3$
– Elément maximal : $\forall\, i \in 1..2p \mid EVEN(i)\ \bullet\ \pi(i) = 4q - 4p + 2i\ \wedge\ \pi(i-1) = \pi(i) - 3$
– Relation canonique : injection totale de X vers Y.

Définition 2 : $finishes^*$ **/ code : f* (précondition :** $p \leq q$**)**
$X\ finishes^*\ Y \Leftrightarrow \forall\, x \in X\ \bullet\ \exists\, y \in Y\ \bullet\ x\ finishes\ y$
– Contrainte selon le formalisme de LIGOZAT :

$$\forall\, \pi \in \Pi \mid X\ \pi\ Y\ \wedge\ X\ finishes^*\ Y\ \bullet\ \exists\, s \in inc\_seq(p,q)\ \bullet$$
$$(\forall\, i \in 1..2p \mid EVEN(i)\ \bullet\ \pi(i) = 4s(i/2) - 1\ \wedge\ \pi(i-1) = \pi(i) - 1)$$

– Elément minimal : $\forall\, i \in 1..2p \mid EVEN(i)\ \bullet\ \pi(i) = 2i - 1\ \wedge\ \pi(i-1) = \pi(i) - 1$
– Elément maximal : $\forall i \in 1..2p \mid EVEN(i) \bullet \pi(i) = 4q - 4p + 2i - 1\ \wedge\ \pi(i-1) = \pi(i) - 1$
– Relation canonique : injection totale de X vers Y.

Définition 2a : $Finished\_by^*$ **/ code : F* (précondition :** $p \leq q$**)**
$X\ Finished\_by^*\ Y \Leftrightarrow \forall\, x \in X\ \bullet\ \exists\, y \in Y\ \bullet\ x\ Finished\_by\ y$
– Contrainte selon le formalisme de LIGOZAT :

$$\forall\, \pi \in \Pi \mid X\ \pi\ Y\ \wedge\ X\ Finished\_by^*\ Y\ \bullet\ \exists\, s \in inc\_seq(p,q)\ \bullet$$
$$(\forall\, i \in 1..2p \mid EVEN(i)\ \bullet\ \pi(i) = 4s(i/2) - 1\ \wedge\ \pi(i-1) \leq \pi(i) - 3)$$

– Elément minimal : $\forall\, i \in 1..2p \mid EVEN(i)\ \bullet\ \pi(i) = 2i - 1\ \wedge\ \pi(i-1) = \pi(i) - 3$
– Elément maximal : $\forall i \in 1..2p \mid EVEN(i) \bullet \pi(i) = 4q - 4p + 2i - 1\ \wedge\ \pi(i-1) = \pi(i) - 3$
– Relation canonique : injection totale de X vers Y.

Définition 3 : *meets** **/ code : m* (précondition :** $p \leq q$**)**
$X\ meets^*\ Y \Leftrightarrow \forall\, x \in X\ \bullet\ \exists\, y \in Y\ \bullet\ x\ meets\ y$
– Contrainte selon le formalisme de LIGOZAT :

$$\forall\, \pi \in \Pi \mid X\ meets^*\ Y\ \wedge\ X\, \pi\, Y \bullet \exists s \in inc\_seq(p,q)\ \bullet$$
$$(\forall\, i \in 1..2p \mid EVEN(i)\ \bullet\ \pi(i) = 4s(i/2)–3)$$

– Elément minimal : $\forall i \in 1..2p \mid EVEN(i) \bullet \pi(i) = 2i-3 \wedge \pi(i-1) = max(0; \pi(i)-3)$
– Elément maximal :

$$\forall\, i \in 1..2p \mid EVEN(i)\ \bullet$$
$$\pi(i) = 4q - 4p - 3 + 2i\ \wedge\ \pi(i-1) = \pi(i-1) - 1$$

– Relation canonique : injection totale de X vers Y.

Définition 3a : *Met_by** **/ code : M* (précondition :** $p \leq q$**)**
$X\ Met\_by^*\ Y \Leftrightarrow \forall\, x \in X\ \bullet\ \exists\, y \in Y\ \bullet\ x\ Met\_by\ y$
– Contrainte selon le formalisme de LIGOZAT :

$$\forall\, \pi \in \Pi \mid X\ Met\_by^*\ Y\ \wedge\ X\, \pi\, Y\ \bullet\ \exists\, s \in inc\_seq(p,q)\ \bullet$$
$$(\forall\, i \in 1..2p \mid EVEN(i)\ \bullet\ \pi(i-1) = 4s(i/2) - 1)$$

– Elément minimal : $\forall\, i \in 1..2p \mid ODD(i)\ \bullet\ \pi(i) = 2i+1 \wedge \pi(i+1) = min(4q\,;\, \pi(i)+3)$
– Elément maximal :

$$\forall\, i \in 1..2p \mid ODD(i)\ \bullet$$
$$\pi(i) = 4q - 4p + 2i + 1\ \wedge\ \pi(i+1) = min(4q\,;\, \pi(i) + 3)$$

– Relation canonique : injection totale de X vers Y.

Définition 4 : *overlaps** **/ code : o* (précondition :** $p \leq q$**)**
$X\ overlaps^*\ Y \Leftrightarrow \forall x \in X\ \bullet\ \exists\, y \in Y\ \bullet\ x\ overlaps\ y$
– Contrainte selon le formalisme de LIGOZAT :

$$\forall\, \pi \in \Pi \mid X\, \pi\, Y\ \wedge\ X\ overlaps^*\ Y\ \bullet\ \exists\, s \in inc\_seq(p,q)\ \bullet$$
$$(\forall\, i \in 1..2p \mid EVEN(i)\ \bullet\ \pi(i) = 4s(i/2) - 2 \wedge \pi(i-1) \leq \pi(i) - 2)$$

– Elément minimal : $\forall i \in 1..2p \mid EVEN(i) \bullet \pi(i) = 2i-2 \wedge \pi(i-1) = max(0; \pi(i)-4)$
– Elément maximal : $\forall i \in 1..2p \mid EVEN(i) \bullet \pi(i) = 4q-4p+2i-2 \wedge \pi(i-1) = \pi(i)-2$
– Relation canonique : injection totale de X vers Y.

Définition 4a : *Overlapped_by** **/ code : O* (précondition :** $p \leq q$**)**
X *Overlapped_by** $Y \Leftrightarrow \forall\, x \in X \bullet \exists\, y \in Y \bullet x$ *Overlapped_by* y
– Contrainte selon le formalisme de LIGOZAT :

$$\forall\, \pi \in \Pi \mid X\ \pi\ Y \ \wedge\ X\ \mathit{Overlapped\_by}^*\ Y \bullet \exists\, s \in \mathit{inc\_seq}(p,q) \bullet$$
$$(\forall\, i \in 1..2p \mid \mathit{EVEN}(i) \bullet \pi(i-1) = 4s(i/2)–2 \ \wedge\ \pi(i) \geq \pi(i-1)+2)$$

– Elément minimal : $\forall\, i \in 1..2p \mid \mathit{ODD}(i) \bullet \pi(i) = 4i-2 \ \wedge\ \pi(i+1) = \pi(i)+2$
– Elément maximal : $\forall\, i \in 1..2p \mid \mathit{ODD}(i) \bullet \pi(i) = 4q-4p+2i \ \wedge\ \pi(i+1) = \pi(i)+2$
– Relation canonique : injection totale de X vers Y.

Définition 5 : *during** **/ code : d*** $(p \leq q)$

$$X\ \mathit{during}^*\ Y \Leftrightarrow \forall\, x \in X \bullet \exists\, y \in Y \bullet$$
$$x\ \mathit{during}\ y \ \wedge\ (\forall\, x_1, x_2 \in X\ ;\ y \in Y \mid x_1\ \mathit{during}\ y \ \wedge\ x_2\ \mathit{during}\ y \bullet x_1 = x_2)$$

– Contrainte selon le formalisme de LIGOZAT :

$$\forall\, \pi \in \Pi \mid X\ \pi\ Y \ \wedge\ X\ \mathit{during}^* Y \bullet \exists\, s \in \mathit{inc\_seq}(p,q) \bullet$$
$$(\forall\, i \in 1..2p \mid \mathit{EVEN}(i) \bullet \pi(i) = 4s(i/2) - 2 \wedge \pi(i+1) = \pi(i))$$

– Elément minimal : $\forall\, i \in 1..2p \mid \mathit{EVEN}(i) \bullet \pi(i) = 2(i-1) \ \wedge\ \pi(i+1) = \pi(i)$
– Elément maximal : $\forall\, i \in 1..2p \mid \mathit{EVEN}(i) \bullet \pi(i) = 4q-4p+2i-1 \ \wedge\ \pi(i+1) = \pi(i)$
– Relation canonique : injection totale de X vers Y.
Dans le cas de *during*$*$, une contrainte additionnelle a été spécifiée, pour éviter la configuration dégénérée lorsque tous les composants convexes de X devraient se dérouler durant un même composant convexe de Y. Le résultat est une injection de X vers Y qui, contrairement au cas dégénéré, a du sens lors d'un traitement pratique d'occurrences d'événements répétitifs.

Définition 5a : $Contains^*$ / code : D* (précondition : $p \leq q$)
$X\ Contains*\ Y \Leftrightarrow \forall\, x \in X \bullet \exists\, y \in Y \bullet x\, Contains\, y$
– Contrainte selon le formalisme de LIGOZAT :

$$\forall\, \pi \in \Pi \mid X\, \pi\, Y\ \wedge\ X\, Contains^*\, Y \bullet$$
$$(\forall\, i \in 1..2p \mid ODD(i) \bullet$$
$$((\pi(i) = 0\ (mod\ 4)\ \wedge\ \pi(i+1) \geq \pi(i) + 4)\ \vee$$
$$(\pi(i) = 1\ (mod\ 4)\ \wedge\ \pi(i+1) \geq \pi(i) + 7)\ \vee$$
$$(\pi(i) = 2\ (mod\ 4)\ \wedge\ \pi(i+1) \geq \pi(i) + 6)\ \vee$$
$$(\pi(i) = 3\ (mod\ 4)\ \wedge\ \pi(i+1) \geq \pi(i) + 5)\ \vee))$$

– Elément minimal : $\forall\, i \in 1..2p \mid EVEN(i) \bullet \pi(i) = 2i\ \wedge\ \pi(i-1) = \pi(i) - 4$
– Elément maximal : $\forall\, i \in 1..2p \mid EVEN(i) \bullet \pi(i) = 4q - 4p + 2i\ \wedge\ \pi(i-1) = \pi(i) - 4$
– Relation canonique : surjection partielle de Y vers X.

Du fait de la contrainte plus restrictive, imposée à la relation $during^*$ on vérifie que l'assertion ($X\ Contains^* Y\ \wedge\ Y\ during^* X$) nécessite $p = q$, et implique qu'il existe une stricte bijection entre X et Y.

Les deux définitions suivantes sont issues de l'extension de la relation d'ALLEN *equals* pour les cas où p diffère de q. Ils traduisent un point de vue ensembliste : $X\ subset^*\ Y$ signifie que tous les composants de X sont aussi (i.e. : *equals*) des composants de Y. En revanche, $X\ equals^*\ Y$ signifie une stricte identité entre les composants X et Y, ainsi une unique instance du type de relation (comme Y lui-même) qui est évidemment à la fois un élément minimal et maximal. $Superset^*$ et $subset^*$ sont mutuellement exclusifs sauf si ($p = q$), dans ce cas elles coïncident avec $equals^*$.

Définition 6a : $subset^*$ / code : s* (précondition : $p \leq q$)
$X\ subset^*\ Y \Leftrightarrow \forall\, x \in X \bullet \exists\, y \in Y \bullet x\, equals\, y$
– Contrainte selon le formalisme de LIGOZAT :

$$\forall\, \pi \in \Pi \mid X\, \pi\, Y\ \wedge\ X\, subset^*\, Y \bullet \exists\, s \in inc\_seq(p, q) \bullet$$
$$(\forall\, i \in 1..2p \mid EVEN(i) \bullet \pi(i) = 4s(i/2) - 1\ \wedge\ \pi(i-1) = \pi(i) - 2)$$

– Elément minimal : $\forall\, i \in 1..2p \mid EVEN(i) \bullet \pi(i) = 2i - 1\ \wedge\ \pi(i-1) = \pi(i) - 2$
– Elément maximal : $\forall i \in 1..2p \mid EVEN(i) \bullet \pi(i) = 4q - 4q + 2i - 1\ \wedge\ \pi(i-1) = \pi(i) - 2$
– Relation canonique : injection totale de X vers Y.

Définition 6b : *Superset** **/ code : S* (précondition :** $q \leq p$**)**
X *Superset** $Y \Leftrightarrow Y$ *subset** X
– Elément minimal :

$$\forall\, i \in 1..(2q - 2p) \bullet \pi(i) = 0) \ \wedge$$
$$(\forall i \in (2p - 2q)..2q \mid ODD(i) \bullet \pi(i) = 2i - 1 \ \wedge \ \pi(i+1) = \pi(i) + 2)$$

– Elément maximal :

$$\forall\, i \in 2q..2p \bullet \pi(i) = 4q) \ \wedge$$
$$(\forall\, i \in 1..2q \mid ODD(i) \bullet \pi(i) = 2i - 1 \ \wedge \ \pi(i+1) = \pi(i) + 2)$$

– Relation canonique : injection de X sur Y.

Corollaire 6 : *equals** **/ code : e* (précondition :** $p = q$**)**
X *equals** $Y \Leftrightarrow X$ *subset** $Y \ \wedge \ X$ *Superset** Y
– Contrainte selon le formalisme de LIGOZAT :
$\forall\, \pi \in \Pi \mid X\ \pi\ Y \ \wedge \ X$ *equals** $Y \bullet (\forall\, i \in 1..2q \bullet \pi(i) = 2i - 1)$
– Elément minimal : $\forall\, i \in 1..2q \bullet \pi(i) = 2i - 1$
– Elément maximal : $\forall\, i \in 1..2q \bullet \pi(i) = 2i - 1$
– Relation canonique : bijection totale (identité) entre X et Y.

Il n'y a aucun moyen d'étendre les relations d'ALLEN *precedes* et *Preceded_by* d'une manière directe et satisfaisante, i.e. : qui préserve le schéma répétitif des occurrences des composants convexes dans des intervalles non convexes. Pour cette raison, nous adoptons d'abord une définition grossière donnée ci-dessous, l'affirmation X *precedes** Y *si et seulement si* tous les composants de X *precedes* tous les éléments de Y. *Preceded_by** est définie en conséquence. Par ailleurs, nous définissons une nouveau type de relation, nommé *interleaves** qui, en pratique représente une configuration répétitive, reconnue comme bien adaptée au traitement des intervalles non convexes.

Définition 7 : *precedes** **/ code : p* (précondition : True)**
X *precedes** $Y \Leftrightarrow \forall\, x \in X\ ;\ y \in Y \bullet x$ *precedes* y
– Contrainte selon le formalisme de LIGOZAT :
$\forall\, \pi \in \Pi \mid X\ \pi\ Y \ \wedge \ X$ *precedes** $Y \bullet (\forall i \in 1..2p \bullet \pi(i) = 0)$
– Elément minimal : $\forall\, i \in 1..2p \bullet \pi(i) = 0$
– Elément maximal $\forall\, i \in 1..2p \bullet \pi(i) = 0$
– Relation canonique : relation binaire correspondant à un graphe biparti complet entre X et Y.

Définition 7a : *Preceded_by** **/ code : P* (précondition : True)**
$X\ Preceded\_by^*\ Y \Leftrightarrow \forall x \in X\ ;\ y \in Y\ \bullet\ x\ Preceded\_by\ y$
– Contrainte selon le formalisme de LIGOZAT :
$\forall\, \pi \in \Pi \mid X\ \pi\ Y\ \wedge\ X\ Preceded\_by^*\ Y\ \bullet\ (\forall\, i \in 1..2p\ \bullet\ \pi(i) = 4q)$
– Elément minimal : $\forall\, i \in 1..2p\ \bullet\ \pi(i) = 4q$
– Elément maximal $\forall\, i \in 1..2p\ \bullet\ \pi(i) = 4q$
– Relation canonique : relation binaire correspondant à un graphe biparti complet entre Y et X.

Les définitions 7 et 7a sont évidemment mutuellement exclusives.

Définition 8 : *interleaves** **/ code : i* (précondition :** $p > 1\ \wedge\ p \leq q + 1$**)**

$$X\ interleaves^*\ Y \Leftrightarrow \forall\, x_1, x_2 \in X \mid x_1\ precedes\ x_2\ \bullet$$
$$(\exists\, y \in Y\ \bullet\ x_1\ precedes\ y\ \wedge\ y\ precedes\ x_2)$$

– Contrainte selon le formalisme de LIGOZAT :

$$\forall\, \pi \in \Pi \mid X\ \pi\ Y\ \wedge\ X\ interleaves^*\ Y\ \bullet$$
$$(\forall\, i \in 1..2p-2 \mid EVEN(i)\ \bullet$$
$$((\pi(i) = 0\ (mod\ 4)\ \wedge\ \pi(i+1) \geq \pi(i) + 4)\ \vee$$
$$(\pi(i) = 1\ (mod\ 4)\ \wedge\ \pi(i+1) \geq \pi(i) + 7)\ \vee$$
$$(\pi(i) = 2\ (mod\ 4)\ \wedge\ \pi(i+1) \geq \pi(i) + 6)\ \vee$$
$$(\pi(i) = 3\ (mod\ 4)\ \wedge\ \pi(i+1) \geq \pi(i) + 5)\ \vee))$$

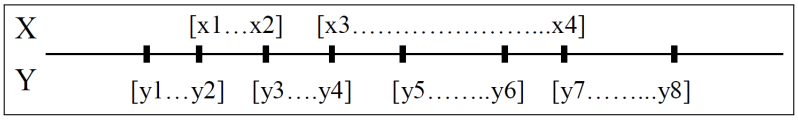
– Elément minimal : $\forall\, i \in 1..2p \mid EVEN(i)\ \bullet\ \pi(i) = 2i - 4\ \wedge\ \pi(i-1) = \pi(i)$
– Elément maximal : $\forall\, i \in 1..2p \mid EVEN(i)\ \bullet\ \pi(i) = 4q - 4p + 2i\ \wedge\ \pi(i-1) = \pi(i)$
– Relation canonique : relation binaire totale de X vers Y (sauf pour le premier ou le dernier élément de X qui peut ne pas avoir d'image).

Ce dernier cas est quelque peu différent des précédents puisque trois composants convexes au lieu de deux sont impliqués dans le prédicat définissant *interleaves** (x_1 et x_2 dans X, et y dans Y). En général, la définition peut être simplifiée ; il n'y a pas de réel besoin de contraindre tous les couples de composants convexes de X comme écrit ci-dessus. Il suffit de vérifier qu'il y a au moins un élément y de Y positionné entre un élément x_1 quelconque de X et son successeur direct x_2. Ainsi tout élément x de X (hormis le premier et le dernier) peut être exactement associé à deux éléments de Y comme suit : x sera associé au dernier élément y_1 vérifiant : (y_1 *precedes* x), et au premier élément y_2 vérifiant (x *precedes* y_2). Bien entendu, le premier (respectivement le dernier) composant convexe de X peut ne pas posséder d'image à sa gauche (respectivement à sa droite).

Un strict entrelacement, i.e. : une séquence alternant les éléments de X et de Y, peut être spécifiée par : X *interleaves** $Y \wedge Y$ *interleaves** X. Dans ce cas, p et q sont tels que $abs(pq) \leq 1$. La relation *interleaves** n'apporte pas d'information sur quel intervalle débute et termine la série combinée.

4.5.3 Calcul de relation étendue

La contre-partie du gain sémantique obtenu lors de la définition synthétique des relations étendues, est une forme d'indéterminisme. Plus précisément, dans certains cas, deux relations étendues différentes peuvent être vérifiées à la fois. Nous appellerons ce phénomène la « concomitance ». Considérons par exemple $X = (3, 5, 7, 13)$ dans $\Pi(4, 8)$: les relations X *Met_by** Y et X *meets** Y sont conjointement satisfaites. Dans ce cas, toute propriété de l'une ou l'autre des deux relations s'applique à X et Y.



Le tableau 4.2 donne un aperçu de la concomitance possible des deux différentes relations étendues. Les résultats présentés dépendent des valeurs relatives de p et q. Ceci étant établi, nous discuterons ensuite de la composition de deux relations étendues puis du filtrage de composants *via* la relation canonique évoquée précédemment. Traiter de la composition nécessite de différencier deux types de relations étendues, celles issues directement de la théorie originale d'Allen et les autres. Chaque cas sera traité successivement.

Il peut être facilement vérifiable que dans aucun cas, sauf lorsque *contains** est impliqué, plus de deux relations étendues concomitantes peuvent coexister. Par conséquent, les 14 relations étendues peuvent être séparées afin de construire l'ensemble des relations canoniques sans recouvrement qui constituent la base de $ALLEN^*$. L'ensemble des éléments de base peuvent être classifiés selon les valeurs relatives de p et q. Si $p = q$, les 13 relations d'Allen sont alors obtenues.

Concomitance	p*	P*	b*	B*	d*	D*	f*	F*	m*	M*	o*	O*	s*	i*	e*
p*		F	F	F	F	F	F	F	F	F	F	F	F	F	F
P*	F		F	F	F	F	F	F	F	F	F	F	F	F	F
b*	F	F		F	F	F	F	F	F	F	F	F	F	T2	F
B*	F	F	F		F	T3	F	T3	T3	F	T3	F	F	T2	F
d*	F	F	F	F		F	F	F	F	F	F	F	F	T2	F
D*	F	F	F	T3	F		F	T3	T3	T3	T3	T3	F	T2	F
f*	F	F	F	F	F	F		F	F	F	F	F	F	T2	F
F*	F	F	F	T3	F	T3	F		F	T3	F	T3	F	T2	F
m*	F	F	F	T3	F	T3	F	F		T1	F	T1	F	T2	F
M*	F	F	F	F	F	T3	F	T3	T1		T1	F	F	T2	F
o*	F	F	F	T3	F	T3	F	F	F	T1		T1	F	T2	F
O*	F	F	F	F	F	T3	F	T3	T1	F	T1		F	T2	F
s*	F	F	F	F	F	F	F	F	F	F	F	F		T2	F
i*	F	F	T2	T2	T2	T2	T2	T2	T2	T2	T2	T2	T2		F
e*	F	F	F	F	F	F	F	F	F	F	F	F	F	F	

$(F = False),\ (T1 = True) \Rightarrow q > p,\ (T2 = True) \Rightarrow q \geq 2p - 1,$
$(T3 = True) \Rightarrow q \geq 2p$

TABLEAU 4.2: Prédicat = « Relations étendues pouvant coexister pour au moins un couple d'intervalles non convexes »

Tous les autres cas sont énumérés ci-dessous. Nous avons volontairement omis le caractère « * » pour des questions de lisibilité. Par ailleurs, les relations doivent être comprises ici comme « strictes », i.e. : sans recouvrement avec une autre.

– cas ($q > p$ et $q < 2p - 1$), le résultat du processus de séparation est composé de 17 éléments disjoints :

$$\{\ \ p, P, b, B, f, F, m, (m \wedge M), (m \wedge O), \\ M, (M \wedge o), o, (o \wedge O), O, d, D, s \qquad \}$$

– cas ($q = 2p - 1$), la base contient 28 éléments :

$$\{\ \ s, D, d, O, o, (s \wedge i), (D \wedge i), (d \wedge i), (o \wedge i), (O \wedge i), \\ (o \wedge O), M, (M \wedge i), (M \wedge o), m, (m \wedge M), (m \wedge i), \\ (m \wedge O), F, (F \wedge i), f, (f \wedge i), B, (B \wedge i), b, (b \wedge i), P, p \ \ \}$$

– cas ($q \geq 2p$), la base contient 53 éléments :

$$\{ \quad s, D, (s \wedge i), (D \wedge i), (D \wedge B), (D \wedge B \wedge i), (D \wedge F), (D \wedge F \wedge i),$$
$$(D \wedge m), (D \wedge m \wedge i), (D \wedge M), (D \wedge M \wedge i), (D \wedge o), (D \wedge o \wedge i),$$
$$(D \wedge O), (D \wedge O \wedge i), d, (d \wedge i), O, (O \wedge i), o, (o \wedge i), (o \wedge O),$$
$$(o \wedge O \wedge D), M, (M \wedge i), (M \wedge o), (M \wedge o \wedge D), m, (m \wedge i), (m \wedge M),$$
$$(m \wedge O), (m \wedge M \wedge D), (m \wedge O \wedge D), F, (F \wedge i), (F \wedge M), (F \wedge O),$$
$$(F \wedge M \wedge D), (F \wedge O \wedge D), f, (f \wedge i), B, (B \wedge i), (B \wedge F), (B \wedge m),$$
$$(B \wedge o), (B \wedge F \wedge D), (B \wedge m \wedge D), (B \wedge o \wedge D), b, (b \wedge i), P, p \qquad \}$$

Pour toute position relative spécifique de p et q (supposons $p \leq q$), alors l'ensemble $(ALLEN^* \cap \Pi(p, q))$ peut être entièrement construit en considérant toutes les unions possibles des relations « atomiques » énumérées ci-dessus. Ces ensembles de base peuvent constituer une entrée à des environnement de SAT qui généralement supposent que les relations prises en considération sont disjointes [Wallgrün 06] et [Westphal 09].

Un autre résultat préliminaire utile concerne la relation mutuelle entre les éléments extrémaux donnés pour chaque relation étendue (cf. le paragraphe introduisant $ALLEN^*$ 4.5.1). Pour tout p donné inférieur ou égal à q, l'ensemble de ces éléments hérite de la structure de treillis sur $\Pi(p, q)$. La figure 4.5.2 (cas de $p < q$) et la figure 4.5.3 (cas $p = q$) donnent les diagrammes de Hasse pour les sous-treillis restreints aux éléments extrêmes. Les éléments sont codés avec un signe (soit « – » ou « + ») suivi d'une lettre. La lettre est l'initiale de la relation étendue (l'étoile est ici omise pour une meilleure lisibilité). Le signe « – » est utilisé pour un élément minimal et le « + » pour un maximal. Par exemple, $-P$ est un raccourci pour « l'élément minimal de $Preceded\_by^*$ ».

Dans le cas de $p = q$, les éléments minimaux et maximaux sont égaux pour toutes les relations qui induisent une correspondance canonique bijective. Dans ce cas, aucun signe n'est ajouté à la lettre servant de code.

4.5.4 Transposition

La dénomination des éléments d'$ALLEN^*$ rend compte de la sémantique des relations d'Allen et permet en particulier d'exprimer simplement la correspondance entre une relation et son inverse. Dans la section 4.3.2, nous évoquions comment, dans le cas des relations étendues de Π, la transposition se substituait à la relation inverse. De manière générale, $ALLEN^*$ n'est pas stable pour la transposition dans le sens qui suit.

Supposons $X\ R^*\ Y$, ainsi, nous inférons qu'il existe des R dans Π tel que $X\ R\ Y$. Par conséquent R^t appartient aussi à Π, mais il n'est pas garanti que R^t soit de type R^{*t} ni même que R^t appartienne à $ALLEN^*$.

Par exemple : $\pi = (1, 2, 9, 10) \in \Pi(4, 6)$ et $\pi \in begins^*$,

alors $\pi^t = (1, 4, 4, 4, 5, 8) \in \Pi(6, 4)$ mais $\pi^t \notin Begun\_by^*$ et $\pi^t \notin ALLEN^*$.

La stabilité pour la transposition s'applique seulement pour des couples particuliers tels que : ($precedes^*$, $Preceded\_by^*$) ou lorsque $p = q$. Dans ce dernier cas, toutes les

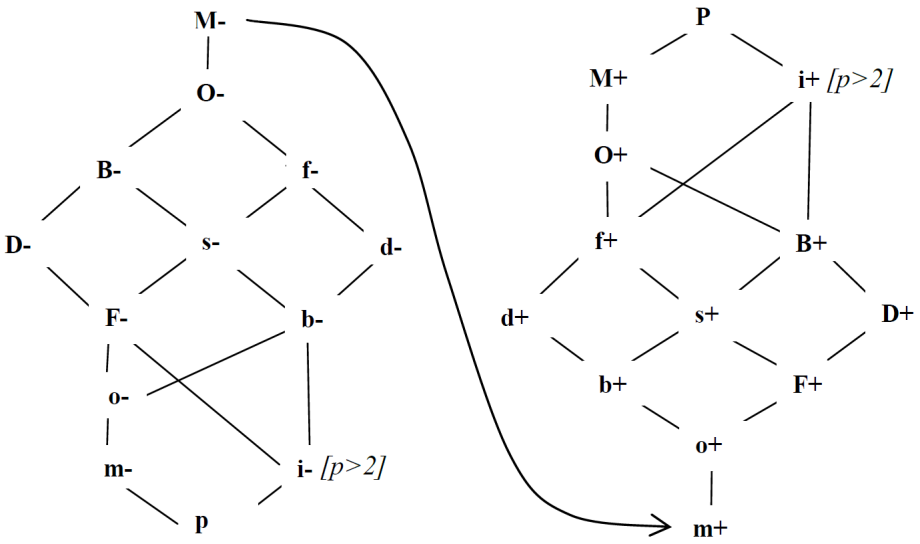


FIGURE 4.5.2: Diagrammes de HASSE pour la structure de treillis des éléments extrêmes des relations étendues d'$ALLEN^*$ (cas : $p < q$)

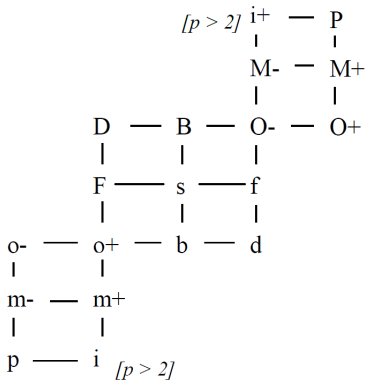


FIGURE 4.5.3: Diagrammes de HASSE pour la structure de treillis des éléments extrêmes des relations étendues d'$ALLEN^*$ (cas : $p = q$)

relations couplées sont mutuellement inverses.

Dans la section 4.7 qui est dédiée au filtrage, plusieurs résultats et propriétés à propos de la transposition seront exposés. Auparavant, la composition sera étudiée plus en détail.

4.6 Composition

Sous réserve de vérifier des contraintes de cardinalité évidentes, les relations étendues $ALLEN^*$ peuvent être composées entre elles. Dans notre cas la composition est plus complexe que pour les relations originales d'$ALLEN$, puisque les éléments d'$ALLEN^*$ sont des ensembles de plusieurs relations binaires et non des singletons. Par composition de deux éléments d'$ALLEN^*$, soit R_1^* ; R_2^*, nous indiquons que nous composons n'importe quel R_1 représentatif de R_1^* avec n'importe quel R_2 représentatif de R_2^*. Ainsi, il apparaît clairement que $ALLEN^*$ n'est pas stable par composition. Le calcul de R_1^* ; R_2^* peut donner soit des éléments $ALLEN^*$, soit des éléments du « *Garbage* ».

Néanmoins, en se référant au principe fondamental qui régit les définitions $ALLEN^*$ et les propriétés algébriques générales énoncées par Ligozat [Ligozat 91], les tables de transitivité peuvent être construites en indiquant tous les résultats possibles de la composition dans $ALLEN^*$. Pour des raisons de taille, il n'est pas raisonnable de donner ici l'extension des tables de transitivité dans le cas des relations étendues. Cependant, nous fournissons un algorithme permettant de les calculer .

Calcul des résultats possibles pour la composition en Allen* [2]

Considérons deux éléments R_1^* et R_2^* dans $ALLEN^*$ et R_1, R_2 dans $ALLEN$ tels que $R_1^* = ALLEN^*[R_1]$ et $R_2^* = ALLEN^*[R_2]$. Dans la suite, nous supposerons que : $subset^* = ALLEN^*[equals]$. Quelques cas spécifiques comme $interleaves^*$, $precedes^*$ et $Preceded\_by^*$, qui n'ont pas de contre-partie directe en $ALLEN$ seront discutés plus tard. Soit Γ l'ensemble des résultats possibles pour R_1^* ; R_2^*, i.e. : pour toute configuration π de Γ, il y a au moins deux configurations π_1 de R_1^* et π_2 de R_2^* telles que : π_1 ; $\pi_2 = \pi$. Il est clair que tout résultat de composition calculé à l'aide des tables de transitivité d'$ALLEN$ est également un résultat dans le calcul en $ALLEN^*$, ainsi : $ALLEN^*[R_1 ; R_2] \subseteq \Gamma$. De plus, des résultats additionnels peuvent être obtenus, constituant un sur-ensemble qui peut être calculé de la manière suivante.

Compte-tenu du fait que pour p et q fixés (cf. tableau 4.2), les types de relations $ALLEN^*$ ne sont pas exclusives, R_1^* et R_2^* peuvent être associées à leurs concomitants respectifs $C_1 = \{R_{11}^*, \ldots, R_{1k}^*, \ldots, R_{1K}^*\}$ et $C_2 = \{R_{21}^*, \ldots, R_{2m}^*, \ldots, R_{2M}^*\}$.

Notons $Conc$ la fonction qui retourne l'ensemble des concomitants d'une relation $ALLEN^*$. Ici, par exemple, nous pouvons dire que :

2. Des exemples de résultats de composition sont fournis à : `http://relaxmultimedia2.univ-lr.fr/resources.html`

$$C_1 = Conc(R_1^*) \; et \; C_2 = Conc(R_2^*)$$

Nous conserverons la même notation en remplaçant simplement les parenthèses par des crochets lorsque le paramètre de *Conc* est un sous-ensemble et non un élément de $ALLEN^*$.

Ainsi nous pouvons conclure que :

$$UNION_{k:1..K\,;\,m:1..M}\{ALLEN^*[ALLEN(R_{2k}^*)\,;\,ALLEN(R_{2m}^*)]\} \subseteq \Gamma \qquad (4.6.1)$$

ou, de façon équivalente :

$$ALLEN^*[ALLEN[Conc(R_1^*)]\,;\,ALLEN[Conc(R_2^*)]] \subseteq \Gamma$$

Enfin, pour terminer le processus, nous devons considérer comme éléments possibles du résultat de la composition les concomitants de la partie gauche de la formule 4.6.1. Ceux-ci sont effectivement des candidats, cependant, en raison de contraintes topologiques intrinsèques, certains ne peuvent en aucun cas être atteints. Par exemple, il est facile de vérifier que X *begins** Y composé avec Y *Overlapped_by** Z ne peuvent pas produire X *Begun_by** Z ni X *Met_by** Z. Bien que ces derniers figurent parmi les concomitants candidats potentiels au sens précédent.

Nous avons envisagé tous les cas possibles et produisons ci-après la liste des combinaisons interdites parmi les candidats potentiels. Précisément, considérons la composition dans Allen* de deux éléments R_1^* ; R_2^* avec R_1^* dans $\Pi(p,q)$ et R_2^* dans $\Pi(q,r)$, les interdits sont de deux types, permanents ou contextuels. Le tableau 4.3 donne la liste des interdits permanents, résultats inaccessibles quelles que soient les valeurs de p, $q\,et\,r$. Le tableau 4.4 donne la liste des interdits contextuels inaccessibles seulement lorsque l'inégalité $(p+q<r)$ est vérifiée. Ils complètent dans ce cas la liste des interdits permanents.

Nb. : pour des questions de clarté, le caractère « * » a été retiré dans les deux prochains tableaux.

$R_1 \backslash R_2$	**b**	**B**	d	**D**	**f**	**F**	**m**	**M**	**o**	**O**	**s**
b								B,O		B,M	
B								B,O		B,M	
d											
D											
f							F,o		F,m		
F							F,o		F,m		
m	F,o,d,f	F,o	f,F	F,m	F,m			m,b,d,o		m,f,F,s	F,o
M	B,M		b,B,M	B,M	B,O,b,d	B,O	M,d,f,O		M,b,B,s		B,O
o	f,F,m,d		f,F,m		F,m,b,d			b,d,f,s			F,m
O	B,M		b,B,M		B,M,b,d		b,d,f,s				B,M
s							F,o	B,O	F,m	B,M	

TABLEAU 4.3: Interdits permanents dans la composition R_1^* ; R_2^*

$R_1 \backslash R_2$	**b**	**B**	d	**D**	**f**	**F**	**m**	**M**	**o**	**O**	**s**
b		F,m,o,D		B,M,O		B,M,O,D	F,m,o,D	F,D	B,F,D	F,D	
B				B,M,O		B,M,O	B		B		
d		F,m,o,D				B,M,O,D	F,m,o,D	B,M,O,D	B,F,D	B,F,D	
D											
f		F,m,o,D		F,m,o		B,M,O,D	B,D	B,M,O,D	B,D	B,F,D	
F		F,m,o		F,m,o				F		F	
m			m	o		F,m,o	m		m		
M		B,M,O		O				M		M	
o		m				F	m	F			
O		B				M	B	M			
s		F,m,o,D		B,F,m,M,o,O		B,M,O,D	B,D	F,D			

TABLEAU 4.4: Interdits contextuels dans la composition R_1^* ; R_2^*
avec $R_1^* \in \Pi(p,q) \ \wedge \ R_2^* \in \Pi(q,r) \ \wedge \ (p+q<r)$

Si l'on convient d'appeler *Purge* l'opération qui élimine les combinaisons interdites, il résulte de ce qui précède que :

$$Purge(Conc[ALLEN^*[ALLEN[Conc(R_1^*)]\ ;\ ALLEN[Conc(R_2^*)]]]) \subseteq \Gamma \qquad (4.6.2)$$

Inversement, supposons que R_3^* est un élément de Γ. Alors, il existe au moins X , Y et Z dans NCI tels que les trois assertions suivantes soient avérées : $X\ R_3^*\ Z$, $X\ R_1^*\ Y$ et $Y R_2^* Z$. En conséquence de quoi, en conservant la notation générique $R = ALLEN([R^*])$:

$$\forall\, x : X \bullet (\exists\, y : Y \bullet x\, R_1\, y) \quad (1)$$
$$\forall\, y : X \bullet (\exists\, z : Z \bullet x\, R_2\, y) \quad (2)$$
$$\forall\, x : X \bullet (\exists\, w : Z \bullet x\, R_3\, w) \quad (3)$$

Supposons maintenant, que $S_3 = R_1\,;\, R_2$ dans $ALLEN$. Alors de (1) et (2), il est déduit que $\forall\, x : X \bullet (\exists s : Z \bullet x\, S_3\, s)$, i.e. : $x\, S_3^*\, Z$ dans $ALLEN^*$.

En rapprochant ce résultat de (3), nous concluons que S_3^* et R_3^* sont nécessairement des concomitants dans $ALLEN^*$, ce qui prouve :

$$\Gamma \subseteq Conc[ALLEN^*[ALLEN[Conc(R_1^*)]\,;\, ALLEN[Conc(R_2^*)]]]$$

ou encore plus précisément

$$\Gamma \subseteq Purge(Conc[ALLEN^*[ALLEN[Conc(R_1^*)]\,;\, ALLEN[Conc(R_2^*)]]])$$

puisque $Purge$ ne concerne aucun élément de Γ.

Ces deux raisonnement successifs (cf. Formules 4.6.1 et 4.6.2) permettent finalement de spécifier complètement le résultat cherché en ce qui concerne la composition des éléments de $ALLEN^*$ qui ont des contre-parties directes dans $ALLEN$.

$$\Gamma = Purge(Conc[ALLEN^*[ALLEN[Conc(R_1^*)]\,;\, ALLEN[Conc(R_2^*)]]]) \qquad (4.6.3)$$

Cas particuliers (éléments de $ALLEN^*$ sans contre-partie directe dans $ALLEN$) :

– *interleaves** est clairement hors du champ de la discussion précédente. En effet, toute composition peut produire *interleaves** comme résultat dès que la contrainte $r \geq p + q - 1$ est vérifiée.
– *precedes** ne peut être obtenu comme résultat de la composition sauf lorsque *precedes** est l'un des termes de la composition. Sous cette condition, *precedes** fait toujours partie des résultats possibles.

Les mêmes propriétés sont valides pour *Preceded_by**. Les arguments présentés ci-dessus se traduisent dans l'algorithme 4.1 qui permet de calculer les tables de transitivité dans $ALLEN^*$.

Algorithme 4.1 Calcul des tables de transitivité dans $ALLEN^*$

Pré-conditions:
- p, q, r : INPUT ;
- RES : OUTPUT ;
- //On traite du cas où un élément de $\Pi(p, q)$ est composé avec un élément de $\Pi(q, r)$
- $\{p, q, r\} \in NAT1 \wedge p \leq q \wedge q \leq r$;

Post-conditions:
- $RES \subseteq ALLEN^*$;

début
pour tout couple de types de relations étendues : soit (R_1^*, R_2^*) **faire**
Calculer l'ensemble des concomitants de $R_1^* = Conc\_R_1^*$
Calculer l'ensemble des concomitants de $R_2^* = Conc\_R_2^*$
$RES \leftarrow \oslash$;
pour tout couple dans le produit cartésien ($Conc\_R_1^* \times Conc\_R_2^*$) **faire**
Calculer la composition selon les tables de transitivité de ALLEN
Ajouter le résultat à RES
fin pour
Compléter RES en ajoutant les concomitants de chacun de ses éléments
Retirer de RES tous les éléments interdits (cf. tableau 4.4)
Traiter les cas spécifiques : *precedes**, *Preceded_by**, *interleaves**
fin pour
fin

Un calcul simple (et approximatif) montre que le nombre maximal d'itérations nécessaire au calcul d'un table de transitivité complète peut être majorée par :

$$14 \times 14 \times 7 \times 7 \times 13 = 124852.$$

Il est à noter que ce résultat est indépendant de p, q et r.

Il est possible de quantifier la complexité maximale de l'algorithme précédent, par exemple en termes d'itérations élémentaires exécutées. Si on exclut dans un premier temps les cas particuliers (*precedes**, *Preceded_by** et *interleaves**), pour un couple de relations étendues (R_1^*, R_2^*), on exécutera au pire $7 \times 7 = 49$ itérations. Cas ($D^* =$ *contains**) qui possède 7 concomitants. Chacun de ces 49 cas intermédiaires peuvent donner lieu à 13 possibilités *via* la composition d'ALLEN classique, soit 637 cas au total. Ces 637 cas peuvent eux même posséder chacun 7 concomitants. La complexité maximale peut donc être à ce stade majorée par $7 \times 637 = 4459$. Il convient également de chercher les configurations interdites dans les résultats potentiels, ce qui multiplie par deux la complexité.

En traitant les cas particuliers de la même façon, on peut au final, majorer la complexité maximale de l'algorithme par 10000 (nb. d'itérations élémentaires). En première

approximation, ce calcul est indépendant de la taille des intervalles non convexes considérés. On pourrait certes affiner la majoration en tenant compte du nombre variable de concomitants selon les valeurs de n et p comme cela est indique dans le tableau 4.2, mais l'effet reste marginal.

Nous présentons ci-dessous un exemple de calcul de « $b\,;O$ » avec l'algorithme décrit ci-dessus 4.1.

Soit $p = 2$, $q = 3$ et $r = 4$. Par rapport au tableau 4.2 donne, T1 et T2 sont vraies pour (p, q) et T1 seule est vraie pour (q, r).

Les concomitants obtenus *via* le tableau sont :

- $Conc\_R_1^* = \{b\}$
- $Conc\_R_2^* = \{m, o, O\}$

La composition des éléments de ces deux ensembles donne :

- $b\,; m \rightarrow p$
- $b\,; o \rightarrow p, m, o$
- $b\,; O \rightarrow d, f, O$

La relation p n'étant pas un résultat valide, il n'est pas retenu et donc on obtient :

$$RES = \{m, o, O, d, f\}$$

Puis on ajoute les concomitants de chaque relation contenue dans RES. Comme on se place ici dans le cas (p, r) les conditions T1, T2 et T3 sont vraies, ce qui permet d'obtenir :

- $m \rightarrow m, B, D, M, O$
- $o \rightarrow o, B, D, M, O$
- $d \rightarrow d$
- $f \rightarrow f$
- $O \rightarrow m, D, o, O, F$

Donc, après complément, $RES = \{m, B, o, O, d, D, f, F\}$.

Puis on retire de RES les relations interdites (cf. tableaux 4.3 et 4.4) qui sont B et M, ce qui donne au final $RES = \{m, o, O, d, D, f, F\}$.

4.7 Filtrage

Les sections précédentes traitent des aspects relationnels et logiques des intervalles non convexes. Dans cette section, nous proposons d'adopter une approche ensembliste à la fois constructive et opérationnelle. Nous nous focalisons sur les éléments X et Y dans NCI considérés comme des ensembles de composants convexes $X = (x_i)_{i=1:p}$ et $Y = (y_j)_{j=1:q}$. Selon la définition d'$ALLEN^*$, nous insistons sur la « relation canonique » (cf. formule 4.5.1) qui désigne l'ensemble des composants appariés (respectivement de X et Y) étant effectivement des instances de la relation étendue donnée. Nous fournissons ci-dessous plusieurs opérateurs qui permettent de sélectionner ces éléments spécifiques

appariés lesquels sont au cœur de la relation mutuelle entre X et Y, et éliminent les autres. Nous appelons ceci le filtrage (ou bien *filtering*). Un calcul peut être développé sur cette base pour montrer qu'il est possible de raisonner plus efficacement sur des intervalles non convexes. Nous donnons des définitions préliminaires, puis nous énonçons quelques propriétés qui en découlent.

4.7.1 Définition : Filtres d'ALLEN

Étant donnée une relation d'Allen quelconque, définissons un filtre comme suit :

$$\forall\, R \in ALLEN\ ;\ X, Y \in NCI\ \bullet$$
$$R[X, Y] = \{x \in X\ ;\ y \in Y \mid x\, R\, y\ \wedge\ y\, R_{\sim}\, x\}$$

où la notation $R_{\sim}$ représente l'inverse de R dans $ALLEN$.

$R[\_, \_]$ est le filtre lié à la relation d'Allen R. $R[X, Y]$ est une partie de $NCI \times NCI$ et représente le résultat de l'application du « filtre R » à X et Y.

4.7.2 Définition : Projecteurs

Il est également utile de définir deux projecteurs de manière à accéder à l'un ou l'autre des éléments filtrés, i.e. : $Pr_1.R[\_, \_]$ et $Pr_2.[\_, \_]$ qui sont spécifiés par :

$$\forall\, R : ALLEN\ ;\ X, Y : NCI\ \bullet$$
$$Pr_1.R[X, Y] = \{x \in X\ ;\ y \in Y \mid (x, y) \in R[X, Y]\ \bullet\ x\}\ \wedge$$
$$Pr_2.R[X, Y] = \{x \in X\ ;\ y \in Y \mid (x, y) \in R[X, Y]\ \bullet\ y\}$$

4.7.3 Définition : Filtres d'ALLEN*

Sauf pour la relation *interleaves**, les définitions ci-dessus peuvent être aisément étendues à $ALLEN^*$. Le résultat de l'application d'un filtre ($R^*[\_, \_]$) ou des projecteurs ($Pr_1.R^*[\_, \_]$, $Pr_2.R^*[\_, \_]$) se concrétise par le retrait de tous les éléments convexes n'étant pas impliqués dans la relation canonique, les autres étant conservés.

Le cas d'*interleaves** est spécial du fait que la propriété sous-jacente est ternaire et non binaire comme dans les autres cas. Ainsi, aucun filtre n'est attaché à *interleaves**.

4.7.4 Propriétés

Les filtres et les projecteurs sont idempotents. Les propriétés suivantes s'appliquent pour tout R dans $ALLEN$, et pour tout X et Y dans NCI :

Soit $R^* = ALLEN^*[R]$ et $R^*_{\sim} = ALLEN^*[R_{\sim}]$.

Propriété 1

$$\begin{aligned}(Pr_1.R[X,Y])\ R^*\ (Pr_2.R[X,Y]) \quad &\wedge \\ (Pr_2.R[X,Y])\ R^*_\sim\ (Pr_1.R[X,Y]) \quad &\wedge \\ (Pr_1.R[X,Y])\ R^*\ Y \quad &\wedge \\ (Pr_2.R[X,Y])\ R^*_\sim\ X\end{aligned}$$

Propriété 2

$$X\ R^*\ Y \Rightarrow X\ R^*\ (Pr_2.R^*[X,Y])\ \wedge\ (Pr_2.R^*[X,Y])\ R^*_\sim\ X$$

4.8 Relations ALLEN* et PTOM

Les relations $ALLEN^*$ que nous venons de définir doivent pouvoir s'exprimer avec le métamodèle PTOM. $ALLEN^*$ est conçu pour les relations entre intervalles non convexes, ces relations s'établissent par conséquent entre deux ensembles d'intervalles. Nous considérons que les propriétés temporelles d'un événement produisent un ensemble d'intervalles. Ainsi, les relations $ALLEN^*$ peuvent être utilisées pour spécifier un lien entre deux événements.

Par conséquent, nous proposons d'ajouter au métamodèle PTOM (cf. figure 4.8.1) une classe d'association `NC_TopologicRel` entre des `T_Event`.

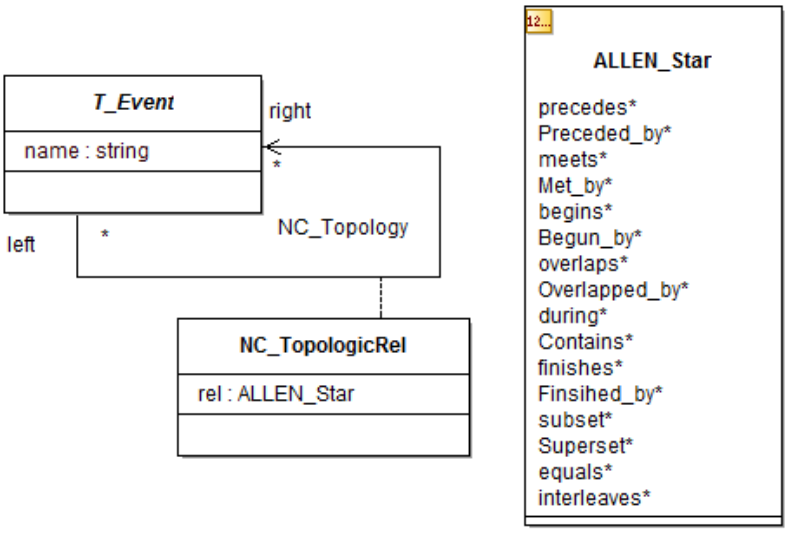


FIGURE 4.8.1: Relations ALLEN* dans le métamodèle PTOM

L'énumération `ALLEN_Star` contient l'ensemble des relations accessibles et définies dans ce chapitre. Ces relations seront spécifiées par l'utilisateur lorsqu'il va définir des événements avec la syntaxe textuelle de PTOM.

4.9 Conclusion

Pour répondre à des besoins et à des objectifs opérationnels, nous avons développé une extension pratique des relations classiques d'ALLEN, sans pour autant délaisser les aspects théoriques. En effet, nous fondons notre proposition sur une restriction pragmatique des travaux théoriques de LIGOZAT.

Un langage formel de relations qualitatives entre intervalles non convexes a été défini afin de permettre la définition de positions relatives entre événements comme par exemple : « *chaque semaine le mardi suit le lundi* ». Ces relations sont nommées $ALLEN^*$.

Parmi l'ensemble extrêmement important des relations possibles entre intervalles non convexes, nous avons retenu des types spécifiques qui représentent une large part de la sémantique d'ALLEN. Ainsi, notre proposition couvre la connaissance commune utile tout en gardant une base théorique rigoureuse.

Une relation qui est utilisée communément dans la pratique a été spécifiée (e.g. : *interleaves**) et trois autres ont été transformées de manière à garder du sens dans un contexte d'intervalle non convexe (*precedes**, *Preceded_by** et *contains**). La relation *equals* a été séparée en *subset** et *Superset**, car elle prouve la dégénérescence dans la mesure où les intervalles non convexes sont considérés.

Le calcul de composition entre les relations étendues en $ALLEN^*$ a été étudié et des tables de transitivité ont été calculées. La structure distributive du treillis se présente comme un cadre général pour comparer les relations entre elles. Le filtrage effectue une sorte de projection de l'espace d'$ALLEN^*$ sur $ALLEN^*$.

La vérification de la cohérence des données temporelles repose clairement sur la connaissance topologique telle que celle traitée par ALLEN. Les périodes d'accès et les occurrences d'événements sont uniformément périodiques ou au moins répétitives, d'où la nécessité des extensions $ALLEN^*$ pour les prendre en considération.

La connaissance fondamentale des données de type temporel est incluse dans les calendriers. Notre métamodèle d'événement, accompagné des définitions $ALLEN^*$, nous permet d'offrir un moyen cohérent et compact pour modéliser les calendriers vus comme un ensemble d'événements périodiques connectés entre eux à l'aide des relations $ALLEN^*$. Ce cas d'application est détaillé dans le chapitre 5.

Chapitre 5

Modèle de calendrier et vérification sémantique

5.1 Introduction

Les métamodèles introduits aux chapitres précédents permettent de spécifier des événements et leur structure ainsi que des règles décrivant leurs propriétés temporelles en précisant les contraintes portant sur leurs occurrences : dates, périodicités et relations mutuelles. Au-delà de la géométrie et de la topologie du temps, des éléments de modélisation font explicitement référence à des informations calendaires. C'est en particulier

le cas pour la classe `CalendarUnit` de PTOM et les éléments qui lui sont attachés. La spécification du temps n'est pas propre à un calendrier particulier, mais les calendriers sont indispensables pour ancrer les concepts dans le concret, et en particulier pour fixer une origine et spécifier un système d'unités partagées. Le calendrier grégorien servira de référence à notre travail, mais tout autre calendrier du même type (julien, japonais moderne, républicain, babylonien ancien, GPS calendar...) pourrait lui être substitué.

Nous donnerons une suite de spécifications sous formes de modèles de classes complétés par des clauses exprimées en OCL pour spécifier successivement la structure des éléments calendaires, leur topologie, les multiplicités et cardinalités et enfin certains aspects spécifiques comme par exemple la prise en compte des années bissextiles et l'affectation d'une semaine à une année unique. Les structures et contraintes que nous proposons sont minimales mais suffisent à définir formellement les concepts fondamentaux du calendrier. Naturellement d'autres règles et contraintes peuvent être ajoutées pour enrichir la spécification.

L'utilité de définir formellement les éléments et la sémantique du calendrier est importante lorsqu'on se préoccupe des aspects suivants :

- vérification et validation individuelle des instances d'événements introduits *via* le métamodèle pour peupler un système d'information ;
- cohérence globale du système vérification de la cohérence inter-instances ;
- raisonnement sur les occurrences (interrogation du système).

Un point fondamental qui a déjà été souligné par plusieurs auteurs [Terenziani 02, Leban 86, Niezette 92] est qu'en général les modèles d'événements temporels (et PTOM en particulier) sont naturellement indiqués pour exprimer la sémantique calendaire. En effet, les éléments calendaires sont essentiellement répétitifs et pour la plupart périodiques. Les éléments de modélisation de PTOM joints aux relations d'ALLEN initiales et étendues constituent donc une base naturelle pour modéliser les calendriers.

La seconde partie de ce chapitre indique et illustre par des exemples la manière dont les métamodèles décrits auparavant peuvent efficacement être implémentés et rendus opérationnels. Une grammaire formelle permet de rattacher les éléments de modélisation (événements et règles de périodicité) aux relations topologiques d'$ALLEN^*$, et par conséquent de spécifier les éléments calendaires et leur contraintes.

Un système de règles fondé sur les mêmes éléments permet de vérifier la conformité aux modèles mais aussi la cohérence entre les contraintes stipulées sur les événements et celles, intrinsèques aux calendriers. Nous utilisons OCL pour exprimer ces règles de correction et de cohérence mais d'autres langages (PROLOG ou d'autres DSL) pourraient tout aussi bien être utilisés. La difficulté majeure est de naviguer entre les métamodèles (événements, modèle temporel, modèle de calendrier).

Ces différents aspects sont étudiés dans la suite de ce chapitre. En revanche, les questions de traduction automatique d'un système calendaire vers un autre qui, bien que résolues en théorie et en pratique, restent complexes et sont hors de notre propos.

5.2 Modèle de calendrier

5.2.1 Approche Structurelle

Les principaux éléments de modélisation du calendrier grégorien sont représentés par les classes `Century`, `Year`, `Month` et `Day` (cf. figure 5.2.1). Ils traduisent des concepts liés à des granularités décroissantes. Pour des raisons de concision, le modèle présenté ici ne montre pas les divisions plus fines (heures, minutes, secondes et fractions décimales de secondes), mais il est facile d'imaginer la modélisation correspondante en tous points similaire. Par contre, la notion de semaine - hormis le fait qu'elle agrège une séquence de sept jours - est complexe car elle rompt la relation naturelle d'agrégation, une même semaine pouvant contenir des jours qui appartiennent à deux années différentes. Dans ces conditions, la détermination de la première et dernière semaine d'une année est certes déterministe, mais reste affaire de convention. Ainsi, le premier jour d'une année n'appartient pas nécessairement à la première semaine de l'année en question.

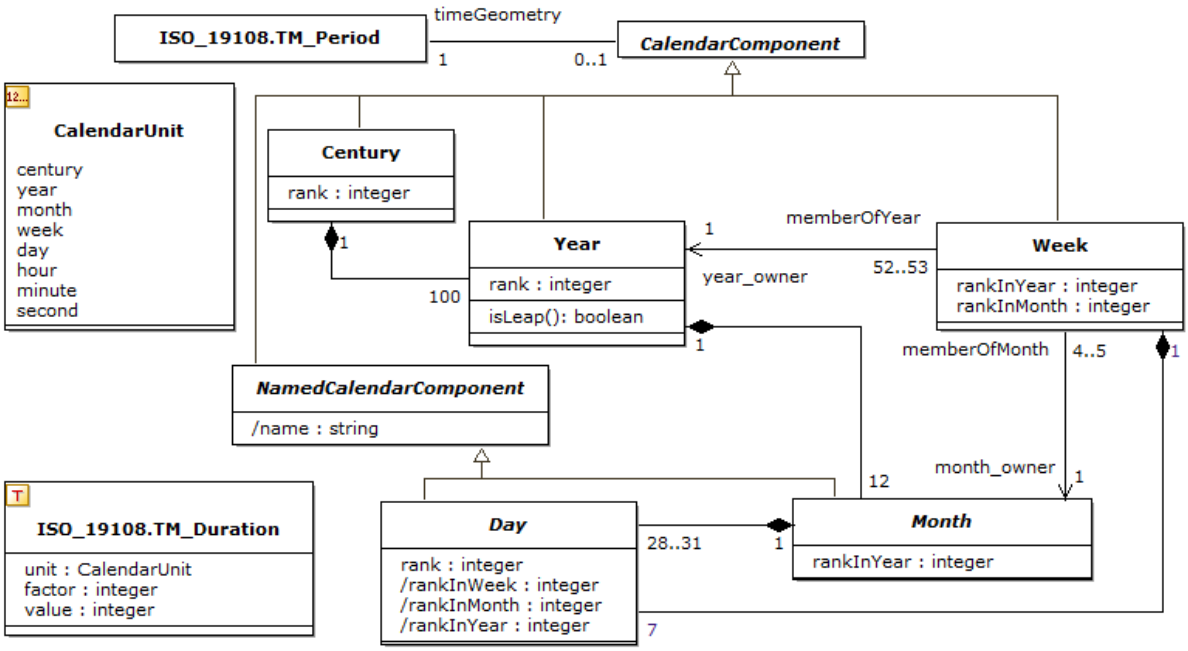


FIGURE 5.2.1: Structure du calendrier grégorien

Spécifier de telles règles nécessite l'usage d'un langage formel spécifique plus puissant que le formalisme semi formel d'UML. Nous donnons ci-dessous des commentaires et un ensemble de contraintes OCL qui précisent les points essentiels de la sémantique liée à la structure du calendrier grégorien présenté en figure 5.2.1. Selon les pays, des

différences existent quant à la définition de certaines propriétés du calendrier grégorien. Par exemple, le premier jour de la semaine est le dimanche dans les pays anglo-saxons. Plus gênant, le calendrier grégorien débute le 4 octobre 1582 et non le 15 dans les états pontificaux. La synchronisation pose de nombreux problèmes. Quant à la Russie, elle n'a adopté le calendrier grégorien qu'en 1918.

L'ensemble de la spécification du calendrier grégorien s'appuie sur des concepts de base hérités de la norme ISO 19108. Le point d'entrée est la notion d'intervalle temporel (`TM_TimeInterval`) ainsi que toutes les classes connexes avec leurs propriétés qui modélisent la géométrie du temps (en particulier le type abstrait `Duration` et les types associés utiles pour modéliser les durées). Le rôle `timeGeometry` prend en charge l'import de ces éléments.

Les attributs dénommés `rank` désignent le rang global de l'instance considérée dans le calendrier. Pour les mois, on indique seulement le rang dans l'année parente, tandis que les jours sont repérés par leurs rangs respectifs dans le calendrier, dans l'année, le mois et la semaine de rattachement. Une opération (usuelle) permet de déterminer le caractère bissextile d'une année donnée.

Nous donnons ci-dessous la liste des principales contraintes du modèle exprimées en OCL et suivies d'un commentaire ou d'une paraphrase en langage naturel.

Le calendrier grégorien débute en l'an 1582. Il peut être prolongé (calendrier grégorien rétroactif) mais cela pose des questions ardues aux réponses ambigües, qui ne sont pas envisagées ici.

Le rang le plus faible dans l'ensemble des années a donc la valeur 1582.

```
context Century
inv C1_1 :
  self.year->forAll(rank >= 1582);
```

Cet état de fait induit des cas particuliers pour les multiplicités qui contredisent le cadre général adopté dans la figure 5.2.1. En effet, en toute rigueur, le XVI^{e} siècle ne comporte que 20 années puisque l'on ignore le temps avant 1582.

Ces aspects sont traités dans le système de contraintes dont nous donnons un aperçu ci-dessous.

Le premier jour du calendrier a pour rang 1. Il s'agit du 288^{e} jour de l'année 1582 qui se trouve être le vendredi 15 octobre.

```
context Day
inv C1_2 :
  self.rank = 1 implies
  (self.name = 'Friday' and self.rankInMonth = 15 and self.rankInYear = 288
  and self.month.rankInYear = 10 and self.month.year.rank = 1582
  );
```

L'année 1582 commence avec le mois d'octobre.

```
context Year
inv C1_3 :
  self.rank = 1582 implies self.month->forAll(rank >= 10);
```

Le mois d'octobre 1582 commence avec son 15ème jour.

```
context Month
inv C1_4 :
  (self.rankInYear = 10 and self.year.rank = 1582) implies
  self.day->forAll(rankInMonth >= 15);
```

D'autres contraintes de portée générale sont présentées dans ce qui suit :

1. Le nombre de jour du mois de février dépend du caractère bissextile de l'année en cours.

```
context Year
inv C2_1 :
  self.isLeap() implies
  (self.month->select(rankInYear = 2).day->size() = 29);
inv C2_2 :
  not(self.isLeap()) implies
  (self.month->select(rankInYear = 2).day->size() = 28);
```

2. Le premier jour de l'année est le premier jour de son premier mois.

```
context Day
inv C3_1 :
  (self.rankInMonth = 1 and self.month.rankInYear = 1) implies
  self.rankInYear = 1;
```

3. Le nom d'un élément calendaire donné est identique à son type (cf. figures 5.2.3 et 5.2.4 pour examiner les sous-types de `Month` et `Day`).

```
context NamedCalendarComponent
inv C4_1 :
  self.name = self.oclType();
```

4. Une semaine est affectée au mois qui contient son jeudi.

```
context Day
inv C5_1 :
  self.name = 'Thursday' implies
  (self.week.month_owner = self.month);
```

5. Une semaine est affectée à l'année qui contient le mois auquel la semaine considérée est affectée, c'est-à-dire : une semaine est affectée à l'année qui contient son jeudi.

```
context Week
inv C5_2 :
  self.year_owner = self.month_owner.year;
```

6. La première semaine du mois est celle de son premier jeudi.

```
context Day
inv C5_3 :
  self.name = 'Thursday' and self.rankInMonth = 7 implies
  self.week.rankInMonth = 1;
```

Les aspects multiplicité sont également abordés dans le modèle. Le nombre de semaines par année est calculable *a posteriori* en vertu de l'invariant C5_1 qui spécifie la règle d'affectation d'une semaine à une année.

Comme il a été vu, le nombre de jours par mois est défini pour février selon le caractère bissextile de l'année en cours. Pour les autres mois, des contraintes seront ajoutées lorsque le sous-typage de `Month` aura été défini (cf. figure 5.2.3).

5.2.2 Approche Topologique

Les contraintes structurelles ne sont pas suffisantes pour spécifier le calendrier, et nous abordons maintenant des contraintes topologiques qui mettent en jeu les relations d'Allen entre les instances des classes respectives `Year` et `Century`.

Topologie des siècles et des années

La succession des siècles, et par conséquent, des années qui les constituent est spécifiée par le diagramme de classes de la figure 5.2.2. Cela se traduit dans le système de numérotation (de nommage) des éléments calendaires considérés.

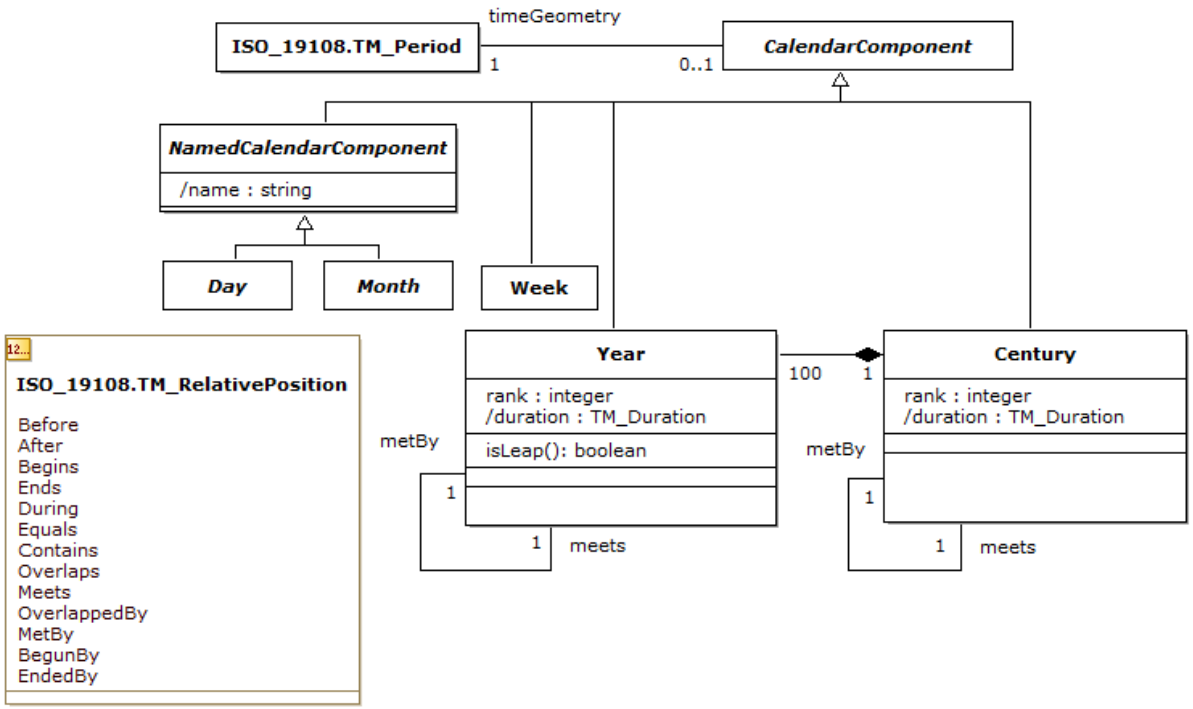


Figure 5.2.2: Topologie des années et siècles

L'invariant C6_0 spécifie que le rang d'une année détermine le siècle auquel elle appartient.

```
context Year
inv C6_0 :
  self.century.rank = (self.rank/100)+1;
```

Les deux invariants suivants (C6_1 et C6_2) expriment que la sémantique des relations *meets* et *metBy* introduites dans le modèle coïncide avec celle d'Allen telle qu'elle est formellement définie dans l'ISO 19108.

La sémantique liée à `ISO_19108::TM_RelativePosition::Meets` se déduit par la spécification de la relation inverse avec *metBy*.

```
context Year
inv C6_1 :
  self.metBy.meets = self implies
  self.timeGeometry.relativePosition(self.metBy.timeGeometry) =
  ISO_19108::TM_RelativePosition::Meets;
```

```
context Century
inv C6_2 :
  self.metBy.meets = self implies
  self.timeGeometry.relativePosition(self.metBy.timeGeometry) =
  ISO_19108::TM_RelativePosition::Meets;
```

Les rangs des siècles, années, mois et jours ont été initialisés dans le modèle structurel. On spécifie ici la relation de récurrence pour calculer le rang des éléments suivants (selon la relation *meets*).

```
context Year
inv C7_1 :
  self.meets.rank = self.rank+1;
```

```
context Century
inv C7_2 :
  self.meets.rank = self.rank+1;
```

Topologie des mois

Le calendrier reconnaît la périodicité annuelle et mensuelle. Les mois sont des éléments nommés alors que les années et les siècles, pour des raisons de cardinalité, sont repérés par leur rang. Le modèle de la figure 5.2.3 indique la succession des mois, leur nommage, leurs rangs, et le nombre de jours qu'ils contiennent. Nous avons volontairement omis la représentation dans le diagramme des mois de mars à octobre qui sont traités de façon totalement homologue à ceux explicitement mentionnés.

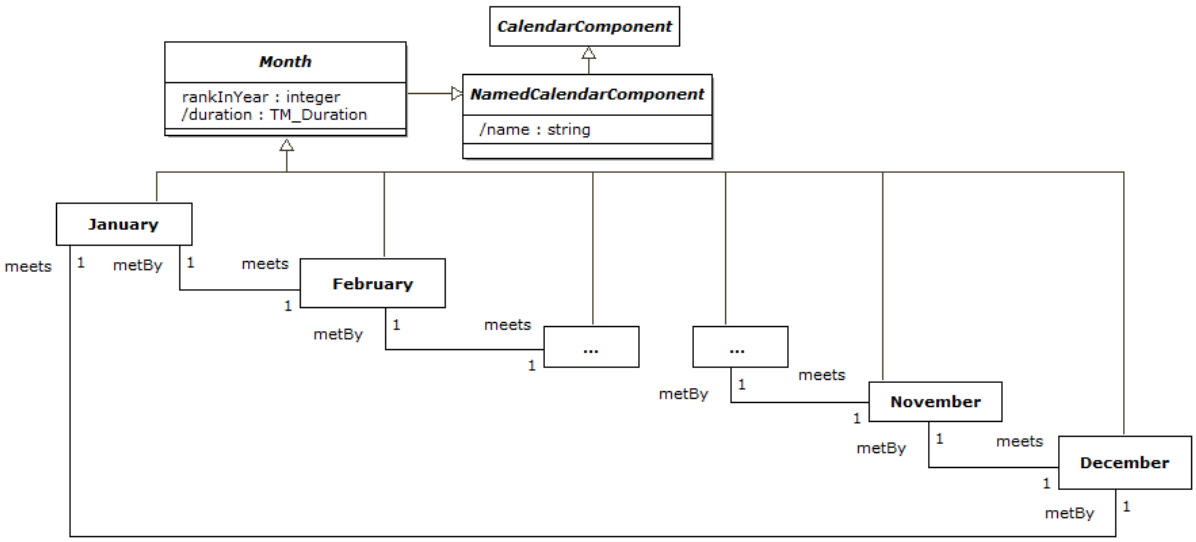


FIGURE 5.2.3: Topologie des mois

Le nommage des mois a déjà été traité par une contrainte OCL exprimée au niveau de la super classe `NamedCalendarComponent` dans le modèle structurel (inv : C4_1).

La double contrainte suivante (inv : C8_1 et C8_2) est spécifiée sur `January` pour définir la sémantique de *meets* et *Met_by*, elle doit l'être sur tous les éléments spécialisant la classe générique `NamedCalendarComponent` , ainsi cela vaudra également pour chaque jour de la semaine (`Monday`, `Tuesday`, etc).

```
context January
inv C8_1 :
  self.meets.metBy = self implies
  self.timeGeometry.relativePosition(self.oclAsType(January).meets.timeGeometry) =
  ISO_19108::TM_RelativePosition::MetBy;
```

```
context January
inv C8_2 :
  self.metBy.meets=self implies
  (self.timeGeometry.relativePosition(self.oclAsType(January).metBy.timeGeometry) =
  ISO_19108::TM_RelativePosition::Meets);
```

L'ensemble de contraintes suivant détermine le nombre de jours du mois considéré (ici janvier) et précise les rangs successifs dans les différents référentiels (année, calendrier). De telles contraintes doivent être spécifiées et adaptées pour les autres mois. Nous ne donnons pas ici la liste correspondante dont l'écriture ne présente pas de difficulté particulière.

```
context Month
inv C9_1 :
  self.oclIsTypeOf(January) implies
  self.rankInYear = 1 and
  self.oclAsType(January).meets.rankInYear = (self.rankInYear+1).mod(12) and
  self.duration.unit=CalendarUnit::day and self.duration.factor=0 and
  self.duration.value=31;
```

Le mois de janvier qui suit un mois de décembre appartient à l'année suivante.

```
context December
inv C10_1 :
  self.meets.year.rank = self.year.rank+1;
```

Topologie des jours

Comme précédemment, nous décrivons dans le modèle de la figure 5.2.4 la succession des jours de la semaine, leurs noms et leurs rangs.

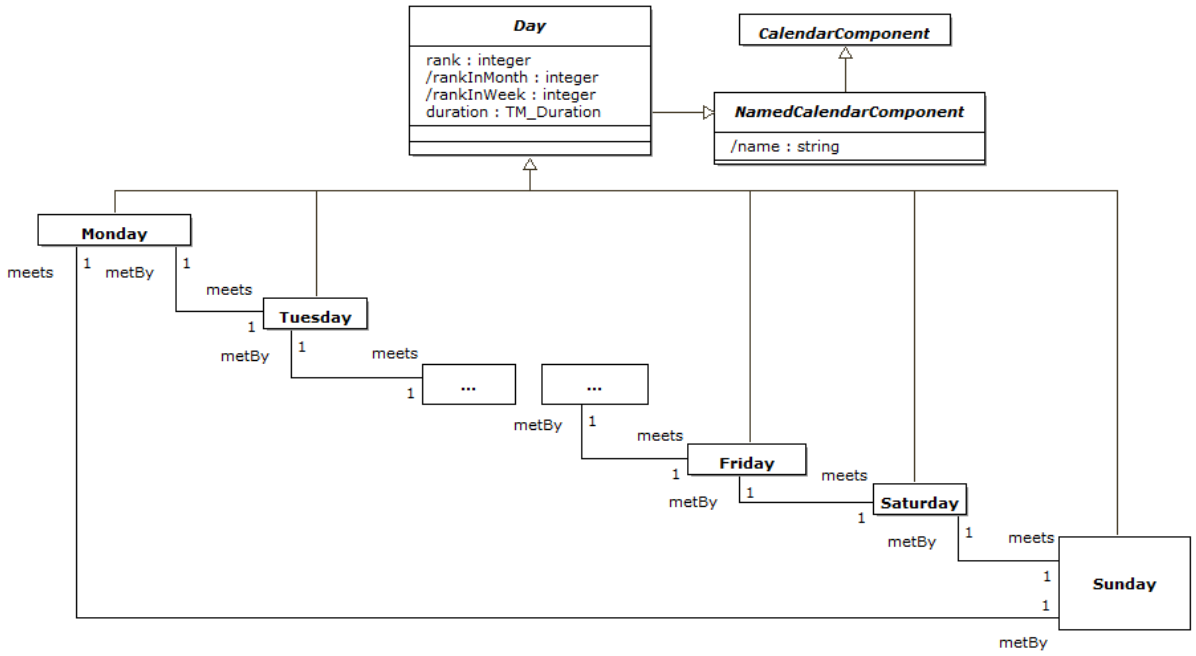


FIGURE 5.2.4: Topologie des jours

Le nommage des jours a déjà été traité au niveau de la super classe `NamedCalendarComponent` (inv : C4_1).

Il en va de même de la sémantique de *meets* et *metBy* (inv C8_1 et C8_2).

Les contraintes ci-dessous terminent la spécification en précisant les règles d'initialisation et de calcul incrémental des rangs. Les invariants C11_1 et C11_3 sont définis ci-dessous comme spécifiques à un jour nommé (i.e. : `Monday`), mais ces deux contraintes sont génériques. Pour être complet il est nécessaire de définir ces deux contraintes pour l'ensemble des jours de la semaine.

```
context Day
inv C11_1 :
  self.oclAsType(Monday).meets.rank = self.rank+1;
```

```
context Day
inv C11_2 :
  self.oclIsTypeOf(Monday) implies self.rankInWeek = 1;
```

```
context Day
inv C11_3 :
  self.oclAsType(Monday).meets.rankInWeek = (self.rankInWeek+1).mod(7);
```

le lundi qui suit un dimanche appartient à la semaine suivante

```
context Sunday
inv C12_1 :
  self.meets.week.rank = self.week.rank+1;
```

L'ensemble des modèles donnés dans ce chapitre pose un problème classique de mise en œuvre. La définition de relations cycliques (*meets* et *metBy*) entre les composants rend le modèle non implémentable dans un univers fini.

Nous avons délibérément choisi de ne pas intégrer cet aspect dans le modèle général de façon à conserver sa simplicité et sa lisibilité. Cette attitude est régulièrement adoptée en modélisation de systèmes d'information. Il convient donc lors de la mise en œuvre de traiter de façon spécifique le début et la fin du calendrier qui doit nécessairement porter sur un intervalle de temps fini (`timeExtent` de PTOM). Hormis pour ces situations initiales et finales, le modèle et sa mise en œuvre coïncident rigoureusement.

5.2.3 Apport des relations étendues Allen*

Les modèles présentés en section 5.2 expriment les aspects structurels, géométriques et topologiques des propriétés temporelles des éléments calendaires sont spécifiés *via* des éléments de modélisation UML et des contraintes OCL portant sur les instances des classes spécifiées.

Les classes du modèle de calendrier peuvent avantageusement être (stéréo-)typées par le type abstrait `T_Event` de PTOM, ce qui permet de bénéficier de la puissance expressive

de PTOM pour représenter les propriétés temporelles, en particulier dans le domaine des événements périodiques.

Il est alors possible d'accéder à un espace de raisonnement plus global portant directement sur les événements répétitifs (associés à des intervalles non convexes) et non sur la suite de leurs occurrences. Ce changement de référentiel logique diminue la complexité des raisonnements.

Le gain est le même que celui dont on bénéficie lorsque l'on raisonne sur les types de fonctions et les opérateurs associés (injection, bijection surjection, domaine, codomaine...) plutôt que sur la définition de ces propriétés à partir de l'ensemble des valeurs des fonctions considérées, par exemple, spécifier $f\ injective$ en place de :

$$\forall x, y\ :\ dom(f)\ \bullet\ f(x) = f(y) \Rightarrow x = y$$

Ainsi, une fois les éléments de modélisation définis, nous pouvons spécifier de façon concise dans $ALLEN^*$:

$meets^*$(January,February) and Met_by^*(February,January)
$meets^*$(February,March) and Met_by^*(March,February)
...
$meets^*$(December,January) and Met_by^*(January,December)

et de manière similaire pour les jours :

$meets^*$(Monday,Tuesday) and Met_by^*(Tuesday,Monday)
...
$meets^*$(Sunday,Monday) and Met_by^*(Monday,Sunday)

5.3 Un modèle de calendrier capturant la sémantique

5.3.1 L'aspect temporel

Les spécifications précédentes ont été données de manière à capitaliser, sous une forme claire et indépendante de toute autre préoccupation, les concepts de base des éléments calendaires et leurs propriétés. Ces modélisations abstraites ne sont pas implémentées telles quelles.

Comme nous l'avons annoncé, l'implémentation des modèles précédents se fait dans le cadre du métamodèle PTOM. Chaque élément calendaire étant un événement temporel dont la périodicité est exprimée par une règle de type `PTOM::PeriocicTemporalOccurrence`.

À titre d'exemple, nous donnons ci-dessous un extrait de la spécification de l'élément `YearEvent` qui représente par un événement périodique, la suite des instances de la classe `Year` du modèle présenté en figure 5.2.1.

Pour cette spécification nous utilisons la grammaire formelle image de PTOM. Elle stipule que :

– L'année possède cent occurrences par siècle, qu'elle commence le 1er jour de janvier et se termine le dernier jour de décembre. On notera que nous ne spécifions pas de relation qualitative entre une année et le 1er jour de janvier, mais nous indiquons que le début d'une année est calculable avec la règle périodique « *le 1er jour de chaque janvier* » et qu'il est possible d'obtenir des extensions pour celle-ci afin d'obtenir une date ancrée dans le calendrier pour chaque début d'année. Il en est de même pour la fin de l'année.
– On notera l'avant-dernière clause de la spécification qui assure explicitement la correspondance (`TemporalDefinitionMapping`) entre l'événement `YearEvent` ici spécifié, et l'élément cible de modélisation `CalendarUnit.year` de PTOM.

```
//événement année
The event "YearEvent" occurs
  periodically (named "YearEvent") according to the rule(s) below
    - rule: 100 times during each century
    - rule: from each 1st day of each January
      to each last day of each December
  Temporal Definition Mapping = {"CalendarUnit.year"}
end of the event
```

FIGURE 5.3.1: Spécification PTOM : événement temporel « année »

5.3.2 L'aspect structurel

Outre ce qui concerne l'expression des propriétés de périodicité, les aspects structurels sont également pris en compte. Nous en donnons une illustration ci-dessous *via* la spécification des mois de l'année. La figure 5.3.2 fournit un exemple de relations structurelles entre l'année et les mois, celles-ci sont définies à l'aide de notre grammaire.

```
//événement année
The event "YearEvent" occurs
  periodically (named "YearEvent") according to the rule(s) below
    - rule: 100 times during each century
    structural as "MonthByLabel" {
      JanuaryEvent, FebruaryEvent,
      MarchEvent, AprilEvent,
      MayEvent, JuneEvent,
      JulyEvent, AugustEvent,
      SeptemberEvent, OctoberEvent,
      NovemberEvent, DecemberEvent
    }
end of the event
```

FIGURE 5.3.2: Spécification PTOM : liste des mois constituant une année

Ainsi une année est composée des 12 mois nommés du calendrier grégorien. La liste ci-dessous est ordonnée ce qui permet d'accéder aux mois par un numéro d'ordre si besoin.

5.3.3 Relations qualitatives entre intervalles non convexes

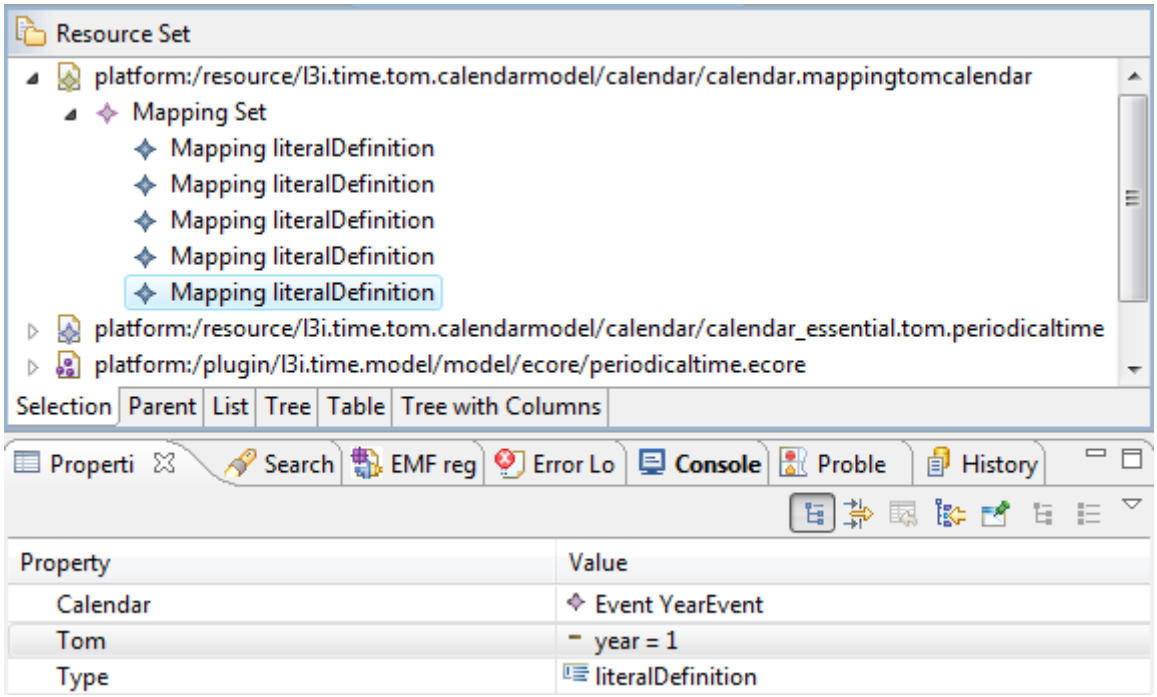
En complément des règles de périodicité qui ont été exprimées en section 5.3.1, nous proposons de définir textuellement des relations qualitatives entre événements, par exemple que chaque occurrence d'année commence en même temps que le mois de janvier (« *Begun_by** JanuaryEvent ») et se termine par un mois de décembre (« *Finished_by** DecemberEvent »).

```
Topology:
Begun_by*(YearEvent,JanuaryEvent) and contains*(YearEvent,AprilEvent) and
Finished_by*(YearEvent,DecemberEvent)
```

Grâce à ces relations, il est directement possible de savoir que l'année commence en même temps que janvier et se prolonge ensuite, alors qu'avec la seule expression de début de la section 5.3.1, nous ne pouvions seulement indiquer que leurs instants de début sont communs sans information complémentaire. Il est ainsi nécessaire de regarder l'expression de fin pour en savoir plus. Il est possible d'ajouter ici d'autres informations telle « *qu'une année contient (contains*) le mois d'avril* » par exemple.

5.3.4 Modèle de correspondance entre PTOM et le modèle de calendrier

La mise en correspondance des événements du modèle de calendrier avec des éléments de Ptom permet de donner une sémantique aux éléments du métamodèle. Ci-dessus nous avons introduit l'élément de grammaire `TemporalDefinitionMapping` spécifiant la mise en correspondance entre l'événement traité et un littéral de l'énumération définissant les unités calendaires avec les événements du modèle de calendrier correspondants. Celui-ci diffère des références `umlClass` et `umlInstance` introduites en section 3.6, puisqu'il fait référence ici aux éléments du métamodèle Ptom et non à un élément d'un modèle du domaine. La figure 5.3.3 montre le modèle de correspondance issu de l'analyse du modèle de calendrier (sous forme textuelle). Ainsi, on peut remarquer que l'événement `YearEvent` est mis en correspondance avec le littéral `year` de l'énumération `CalendarUnit` .



Les attributs :
- `calendar` réfère un événement du modèle de calendrier
- `tom` spécifie un littéral de l'énumération `CalendarUnit` du métamodèle PTOM

FIGURE 5.3.3: Modèle de correspondance de PTOM avec le modèle de calendrier

5.4 Vérification sémantique

5.4.1 Vérification de la sémantique des expressions temporelles

Lors de la phase de peuplement du modèle, c'est-à-dire lors de la constitution d'une base d'événements temporels enrichis de leurs propriétés temporelles (et accessoirement de tout autre type de méta-données), il est classique, et sans difficulté particulière, de vérifier les aspects structurels des données saisies (conformité au méta-modèle de PTOM, complétude, unicité etc.). Dans la mesure où les échanges avec les sources de données se font par l'intermédiaire d'une grammaire formelle développée de manière ad hoc, la correction syntaxique équivaut à la correction structurelle.

Par contre, la correction sémantique est beaucoup plus complexe. Il s'agit d'abord de vérifier la correction interne, c'est-à-dire la conformité des occurrences d'événements (périodiques) avec leur spécification dans le modèle. Ceci garantie la correction individuelle de chaque événement du système.

Il convient également d'assurer la cohérence mutuelle des définitions pour tous les événements en interrelation. L'enjeu de ce dernier point est la garantie de la cohérence

globale du système. Ces derniers événements ne sont pas seulement ceux dont la spécification s'appuie explicitement sur la relation `PTOM::PeriodicRelativePosition`, mais bien tous les événements sans exception, puisque chacun met en jeu des éléments appartenant à la spécification du calendrier grégorien. En conséquence, la distinction entre les deux types de correction (interne, cohérence mutuelle) n'est pas toujours nette, mais ces deux points de vue doivent systématiquement être envisagés.

La vérification sémantique doit donc être supportée par un système de règles dont chaque élément pertinent sera systématiquement vérifié lors de l'introduction de tout nouvel élément dans le modèle.

Notre propos n'est pas de construire un tel moteur de règles, ni même de spécifier un système de règles complet. Notre objectif est uniquement de montrer que nos méta-modèles portent la connaissance nécessaire pour mettre en œuvre le processus de vérification sémantique et de démontrer ce qui doit être fait.

Nous avons présenté en section 3.5 la grammaire de PTOM, qui joue le rôle de vérification « syntaxique », nous présentons maintenant des exemples de vérification sémantique pour illustrer notre démarche qui met en jeu le métamodèle de PTOM avec les contraintes additionnelles OCL et les concepts importés de l'ISO 19108, la sémantique des relations d'$ALLEN^*$ et le modèle de calendrier.

Considérons un événement spécifié avec une contrainte temporelle simple :

« *l'événement a lieu chaque 8^e jour de chaque semaine* »

Cette expression, bien que conforme au métamodèle, est sémantiquement incorrecte. En revanche, une expression voisine telle que « *a lieu chaque 8^e jour de chaque mois* » doit être réputée sémantiquement correcte.

Pour détecter automatiquement l'erreur, il convient de reconnaître les faits suivants :

1. la spécification de l'événement périodique référence deux `CalendarUnit` de PTOM (jour et semaine) ;
2. l'élément semaine de PTOM est associé à l'événement semaine du calendrier. Il en va de même de l'élément jour ;
3. dans le modèle de calendrier, la composition de semaine (`WeekEvent`) vers jour (`DayEvent`) spécifie une multiplicité d'exactement 7 (cf. figure 5.4.1).

Le système de vérification sémantique doit reconnaître que le *rang* 8^e (`NumericRankValue = 8`) est incompatible avec la multiplicité de la composition (semaine/jour). Il faut donc spécifier une règle générique en ce sens et déclarer une erreur en cas de violation de la contrainte.

```
//événement day
The event "DayEvent" occurs
  periodically (named "DayEvent") according to the rule(s) below
    -rule: 7 times during one 1 weeks period
  Temporal Definition Mapping = {"CalendarUnit.day"}
end of the event
```

FIGURE 5.4.1: Extrait du modèle de calendrier modélisant jour (`DayEvent`)

Équivalence en OCL

En OCL, pour vérifier cette expression il serait nécessaire de définir la contrainte de la figure 5.4.2, i.e. pour chaque unité donner l'ensemble des correspondances avec les autres unités calendaires, ce qui rend leurs écritures fastidieuses.

```
context AbsoluteTemporalExpression inv :
self.descriptors->forAll( desc |
if desc.OCLTypeOf(CalendarUnitDescriptor) then
 if desc.value="second"
  and self.descriptors.at(self.descriptors.indexOf(desc)).value="minute" then
   if desc.rank.OCLTypeOf(NumericRank) then
    desc.rank.value<= 60
   endif
 else
  if desc.value="minute"
   and self.descriptors.at(self.descriptors.indexOf(desc)).value="hour" then
    if desc.rank.OCLTypeOf(NumericRank) then
     desc.rank.value<= 60
    endif
  else
   if desc.unite="hour"
    and self.descriptors.at(self.descriptors.indexOf(desc)).value="day" then
     if desc.rank.OCLTypeOf(NumericRank) then
      desc.rank.value<= 24
     endif
 ...
   endif
  endif
 endif
endif
)
```

FIGURE 5.4.2: Vérification du *rang* d'une expression temporelle en OCL

La section suivante propose une approche qui s'appuie notamment sur le modèle de calendrier présenté précédemment.

5.4.2 Vérification sémantique à l'aide de contraintes et du modèle de calendrier

Nous considérons la vérification sémantique d'un modèle comme le résultat de l'évaluation d'un ensemble de contraintes. La figure 5.4.3 montre les différents modèles mis en jeu pour évaluer une contrainte. Une `Contrainte` s'applique sur le `Modèle à vérifier` ainsi elle est définie sur une des classes du `Métamodèle`.

Cette contrainte nécessite de retrouver une information se trouvant dans le `Modèle de calendrier`. Par exemple, il est nécessaire de connaître le nombre de jours composant une semaine pour ensuite évaluer la contrainte. Pour faire le lien entre le métamodèle et le modèle de calendrier, nous utilisons le modèle de correspondance (cf. section 5.3.4) qui permet d'identifier l'événement associé à un « littéral » de `enum` (Enumération des unités calendaires : de la seconde au siècle).

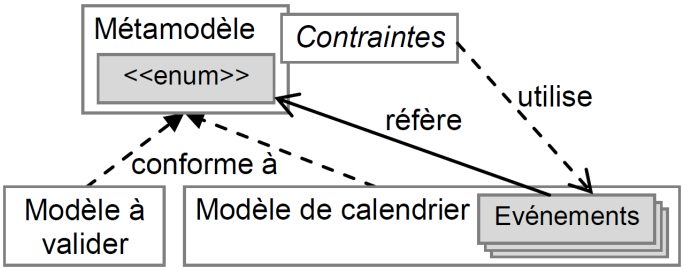


FIGURE 5.4.3: Ressources mises en œuvre dans le processus de vérification

L'utilisation de ces différents modèles et la navigation à travers plusieurs niveaux du MDA ne nous ont pas permis d'utiliser OCL. Ainsi, nous avons préféré utiliser KERMETA et ses fonctionnalités permettant de manipuler plusieurs modèles à différents niveaux de modélisation en même temps.

La figure 5.4.4 montre la mise en œuvre de la vérification de la 1re expression. Nous définissons une contrainte sur la classe `AbsoluteTemporalExpression` pour déterminer la validité des expressions telle que : $desc_n.rang \leq freq(desc_n.unité, desc_{n+1}.unité)$.

La contrainte possède une fonction $freq$ prenant en paramètres les unités de deux descripteurs successifs. Cette fonction retourne la fréquence définie dans le modèle de référence et associée au couple d'unités de descripteurs. Pour le couple (jour, semaine) : $freq(day, week) = 7$. Ainsi le *rang* de l'expression à vérifier doit être inférieur ou égal à 7. Dans l'exemple, le *rang* de l'expression à vérifier a pour valeur 8, donc une erreur sera signalée car $8 > freq(day, week)$.

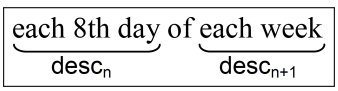


FIGURE 5.4.4: Expression à vérifier sémantiquement

5.5 Conclusion

Dans ce chapitre, nous avons successivement posé les fondements d'une modélisation explicite des concepts essentiels d'un calendrier, selon les points de vue structurel, géométrique et topologique. L'aspect structurel traite des différentes granularités des concepts constituant un calendrier. Les contraintes à caractère géométrique rattachent ces éléments à des repères temporels concrets et spécifient les durées. La topologie, contrôlée *via* un système original $ALLEN^*$ étendant le calcul classique d'Allen, traduit la périodicité et les relations mutuelles de précédence ou de coïncidence entre éléments calendaires périodiques représentant les dates et les durées.

Notre modèle est générique et peut s'appliquer à différents types de calendrier, et en particulier au calendrier grégorien que nous étudions en détail. Notre modélisation est volontairement compatible avec les définitions figurant dans les normes ISO 19108 et ISO 8601.

Notre modélisation de la sémantique du calendrier permet de mettre en œuvre des processus de vérification des données peuplant le modèle. On détecte des incohérences par rapport au métamodèle Ptom mais aussi des erreurs violant la sémantique propre au calendrier en usage.

Pour ce qui est de la validation des informations instanciant le modèle d'événement, une reformulation faite par l'intermédiaire d'une grammaire formelle *ad hoc* permet de se situer au plus près du langage naturel employé par les acteurs effectuant la validation (journalistes), ou accédant à la publication des résultats (utilisateurs finaux).

Notre propos est illustré par des exemples utilisant des contraintes OCL et la grammaire formelle.

Troisième partie

Exploitation des modèles d'événements et applications

« Il n'y a pas une vérité concernant l'IDM, ni un langage de modélisation mais in fine il y aura la votre qui sera intimement liée à vos objectifs et à vos contraintes industrielles, académiques, règlementaires ... »

(Patrick FARAIL, préface du livre "Ingénierie Dirigée par les Modèles - Des concepts à la pratique", ellipse, 2012)

Chapitre 6

De la construction à l'exploitation des modèles d'événements

6.1 Introduction

Nous avons précisé dans le chapitre 3 le rôle pivot du métamodèle Ptom, celui-ci jouant un rôle prépondérant dans la gestion des événements.

Ce chapitre a pour objectif de détailler les différents usages de Ptom. La figure 6.1.1 résume les approches et applicatifs que nous avons mis en œuvre pour créer et exploiter des modèles de Ptom.

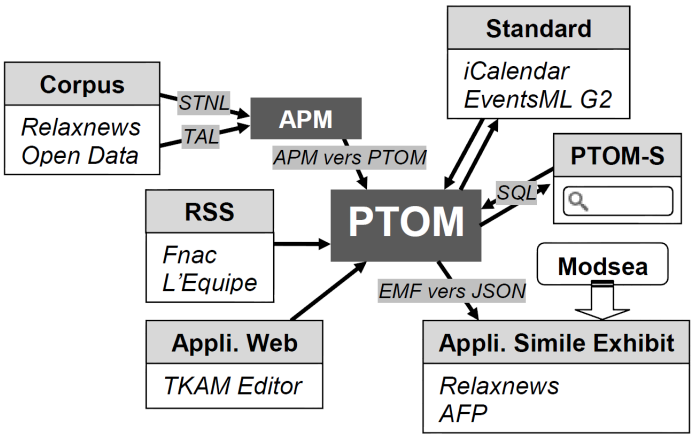


Figure 6.1.1: Approches et applicatifs mis en œuvre pour construire et exploiter des modèles de Ptom

Le rôle central de Ptom est de permettre la création d'événements qui peut être réalisée principalement de deux manières :

- Par une saisie directe pour laquelle nous fournissons une application permettant la création d'événements, à partir d'un formulaire de saisie dont les champs sont directement issus de Ptom (TKAM Editor). La structure des champs présentés dans l'interface de cette application est le reflet direct de la structure de Ptom. Les contrôles exercés lors de la saisie étant issus des contraintes OCL décrites dans les chapitres 3 et 5 complétées par des contrôles exploitant les informations contenues dans le modèle de calendrier grégorien.
- Par importation de données (modèles) conformes à différents métamodèles : grammaire Stnl, annotations TAL, flux RSS, iCalendar et EventsML G2. Un ensemble de transformation de modèles automatise cette phase d'importation, dont certaines sont fondées sur des métamodèles intermédiaires comme APM (concernant les périodes d'accessibilité). Pour tester ces transformations nous avons utilisés différents jeux de données, certains provenant de nos partenaires au sein du projet RMM2, d'autres étant choisis dans des répertoires de données ouvertes pour leur caractère temporel répétitif.

Associé à ce rôle de création, PTOM permet également d'exporter les événements sous différents formats soit pour les diffuser (iCalendar, EventsML G2), soit les visualiser, soit pour les interroger.

Pour l'interrogation d'événements (PTOM-S), à des fins de comparaison de performances, nous avons étudié deux types de requêtes : forme en intension et en extension. Nous montrons que PTOM et les relations ALLEN* nous servent à créer des types de requêtes pour interroger des expressions temporelles en intension stockées dans une base de données relationnelle.

Les besoins provenant de cas pratiques liés à la visualisation d'événements nous ont conduits à nous intéresser aux bibliothèques de composants graphiques. Ces dernières, étant relativement structurées, peuvent être gérées *via* l'IDM. Ainsi nous fournissons une méthode de génération d'applications de visualisation spatio-temporelle d'événements utilisant la bibliothèque de composants graphiques Simile Exhibit (MODSEA). Cette approche permet notamment de sélectionner les attributs pertinents à visualiser en établissant des correspondances entre un métamodèle de contenu de l'application de visualisation et celui de PTOM.

La suite de ce chapitre est organisée en quatre sections, la première section 6.2 présente les différents modes d'instanciation, ceci dans un contexte d'interopérabilité avec des normes et standards (section 6.3). Puis nous proposons en section 6.4 une approche exploratoire et comparative pour interroger des modèles de PTOM. Enfin en section 6.5, nous décrivons l'approche IDM destinée à générer des applications web dédiées à la visualisation d'événements.

6.2 Création de modèles conformes à PTOM

Dans cette section, nous montrons différentes manières d'instancier le métamodèle PTOM. De manière classique en IDM, pour instancier un métamodèle, EMF permet la génération d'une API JAVA fournissant les classes et méthodes nécessaires pour créer des modèles qui peuvent être sauvegardés au format XMI. Cette méthode nécessite une très bonne connaissance du métamodèle et n'est pas destinée aux utilisateurs finaux.

EMF propose également de manipuler des modèles en XMI *via* un éditeur arborescent, ce qui s'avère compliqué et générateur d'erreurs dont la correction est souvent laborieuse.

Comme nous l'avons discuté dans le chapitre 3, XTEXT propose de générer un analyseur et un générateur de texte basé sur un métamodèle. Ceci permet d'offrir à l'utilisateur final un texte en langage contrôlé proche du langage naturel et de déléguer la gestion des modèles XMI à XTEXT.

Dans cette section nous proposons différentes méthodes de création d'événements et de propriétés temporelles, soit unitaire à l'aide d'une interface web, soit multiple et automatisée à partir de sources externes.

Dans la section 6.2.1 nous présentons l'application web TKAM Editor destinée à des utilisateurs finaux et adaptée à la consultation et à l'édition d'un modèle de proprié-

tés temporelles. Ensuite nous développons le cas de la création automatisée ou semi-automatisée d'événements à partir de sources externes : annotations d'un outil du TAL (section 6.2.2), données ouvertes (section 6.2.3), flux RSS (section 6.2.4).

6.2.1 Formulaire de saisie web

Objectif

Afin de spécifier des événements et leurs propriétés temporelles nous avons développé une interface graphique web permettant la saisie d'expressions temporelles conformes à Ptom, ainsi qu'un module de validation d'expressions [Faucher 11]. Cette application se nomme TKAM Editor (Temporal Knowledge Acquisition and Modeling Editor), elle a fait partie d'un travail de stage de fin d'étude de Master [Faucher 11].

Mise en œuvre

L'instanciation d'un modèle de Ptom est réalisée *via* l'interface web interagissant avec l'utilisateur. Celle-ci a été spécifiée (composants et positionnement) à partir de la grammaire d'expressions temporelles (cf. section 3.5) afin que l'enchaînement des zones de saisie soit ordonné comme des phrases proches du langage naturel. Le contenu de l'interface graphique, notamment certaines listes de choix, comme `CalendarUnit`, est issu directement de l'interrogation du métamodèle Ptom.

Le module de vérification offre à l'utilisateur une saisie ergonomique, et la capacité de vérifier la sémantique des instances du métamodèle temporel avant qu'elles ne soient sauvegardées. Il a comme entrée le modèle contenant la sémantique du calendrier grégorien (cf. chapitre 5). Par exemple, les valeurs de rang sont vérifiées grâce aux propriétés temporelles définies sur les éléments du calendrier incluses dans ce modèle.

Pour faciliter le développement de notre approche et plus particulièrement l'intégration des technologies IDM comme EMF, nous avons choisi une plate-forme de conception d'applications web compatible (GWT) qui permet de développer en Java et donc rend possible l'utilisation d'EMF. L'application permet ainsi de saisir des instants et des intervalles périodiques qui utilisent des expressions temporelles absolues, mais les positions relatives ne sont pas supportées pour l'instant.

6.2.2 Du métamodèle linguistique de période d'accessibilité au métamodèle d'événements composites répétitifs

Dans cette section, nous discutons la transformation de modèles entre le métamodèle linguistique de période d'accessibilité et le métamodèle Ptom. Tout d'abord, nous présentons le métamodèle de période d'accessibilité en section 6.2.2.1, puis la transformation elle-même en section 6.2.2.2.

6.2.2.1 Le métamodèle linguistique de période d'accessibilité (APM)

La figure 6.2.1 fournit le métamodèle de période d'accessibilité (APM) dont les instances sont le résultat de l'annotation réalisée par le module TAL de MoDyCo. Pour nous permettre de transformer les données du TAL en instances de PTOM, nous avons transposé le modèle d'annotation de TEISSÈDRE [Teissèdre 10] en métamodèle (i.e. APM).

L'élément central de ce métamodèle est la classe `AccessPeriod` qui porte des propriétés temporelles. Celles-ci s'expriment sous forme de `CalendarExpression` qui, regroupées en `CalendarExpressionInterval`, permettent de définir des intervalles périodiques.

Une `CalendarExpression` représente une expression, comme : « *tous les mardis à 10h* » (`dayOfWeekByLabel=Tuesday` et `hour=10`). Si à ce `CalendarExpression` on en associe un second tel que « *tous les mardis à 12h* », il est alors possible de constituer un `CalendarExpressionInterval` qui définit qu'un lieu sera ouvert « *tous les mardis de 10h à 12h* ». L'expression « *tous les mardis à 10h* » est le `start` de l'intervalle tandis que « *tous les mardis à 12h* » est le `end`. Ce `CalendarExpressionInterval` constitue la `base` (référence sortante) d'une `AccessPeriod`. Il est possible d'ajouter des `base`s complémentaires pour par exemple spécifier que ce lieu est également ouvert « *tous les jeudis de 14h à 18h* ».

De plus, l'utilisation d'un ou plusieurs `CalendarExpressionInterval` *via* la référence `specification` permet de restreindre la portée de la répétition en sélectionnant des mois utiles comme de « *mars à juin* ». L'expression finale est donc : « *ouvert tous les mardis de 10h à 12h et tous les jeudis de 14h à 18h de mars à juin* ». `CalendarExpressionInterval` est équivalent à `PeriodicTimeSpan` dans PTOM.

Une `AccessPeriod` possède un statut (`status`) pour indiquer la sémantique de la propriété temporelle, e.g. : « *ouvert* » (`open`), « *fermé* » (`closed`). Il est également possible d'ajouter des exceptions *via* la référence `exception` à une `AccessPeriod`.

6.2.2.2 Transformation de modèles

L'objectif de la transformation est de traduire les annotations réalisées par le module TAL de MoDyCo en instances de PTOM et donc de définir un convertisseur entre les métamodèles APM et PTOM.

Principales étapes de la transformation de modèles APM vers PTOM

1. Discriminer les expressions pour déterminer les `PeriodicRule` à générer. Dans l'exemple « *Lundi à samedi de 10h30 à 12h30 et de 14h à 18h30.* » (cf. figure F.2.1) : la phase d'annotation a produit plusieurs `specification` comme le montre la figure F.2.2 (lignes 5 et 8), i.e. « *de 10h30 à 12h30* » et « *de 14h à 18h30* ». Ces deux `specification` sont de même granularité (horaire : `Time`). PTOM interdit de spécifier plusieurs `FreqWithCalendarRef` ou `PeriodicTimeInterval` avec la même granularité au sein d'une même règle (`PeriodicRule`). Ainsi autant de règles que nécessaire sont à créer, deux pour ce cas. Les propriétés concernant la `base` seront

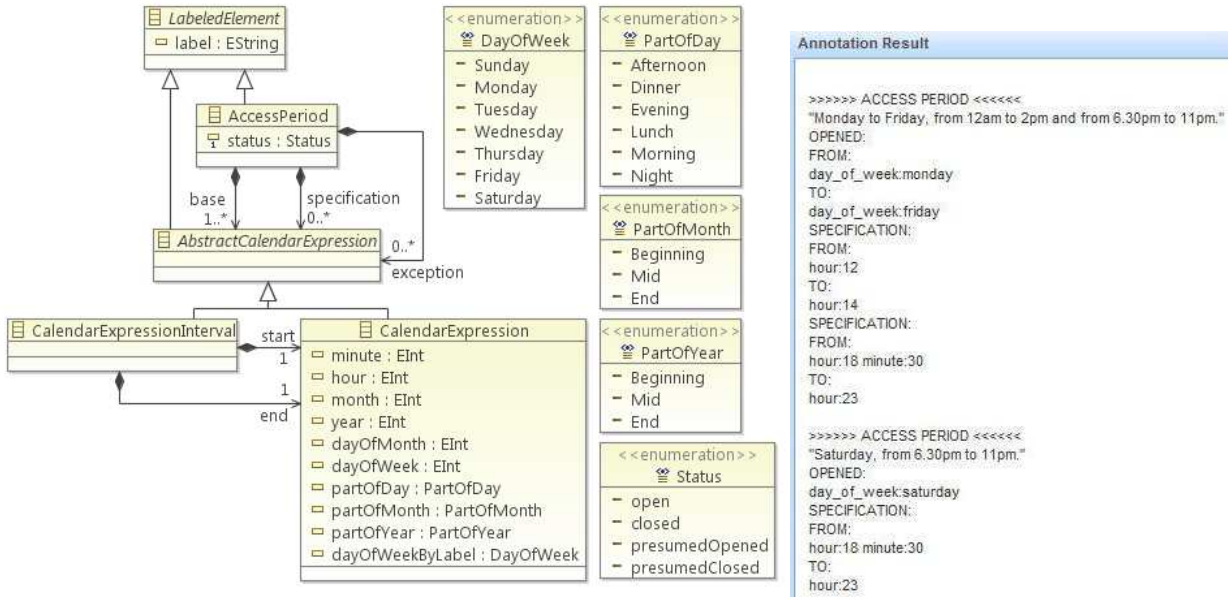


FIGURE 6.2.1: Métamodèle de période d'accessibilité (APM) accompagné d'un exemple d'annotations

dupliquées dans celles-ci. La granularité d'une période d'accessibilité est calculée par une méthode `getGranularity`.

2. Vérifier chaque règle de périodicité de type : `FreqWithCalendarRef` et `PeriodicTimeInterval`. Plus précisément, les `PeriodicInstant` qui leur sont associés sont validés de manière structurelle et sémantique. Dans le cas de `PeriodicTimeInterval`, la vérification concerne également la cohérence des deux `PeriodicInstant` qui le composent (b). Ainsi la validation consiste à vérifier :

   a) qu'un instant ou intervalle calendaire possède des propriétés : invariant `notEmpty` (cf. figure 6.2.2) ;

   b) la cohérence des granularités entre les expressions calendaires de début et de fin pour un intervalle : invariant `granularityConsistency` ;

   c) la cohérence des valeurs de rang, e.g. : « *chaque* 8^e *jour de chaque mois* », « *chaque* 8^e *jour de chaque semaine* » est faux car $8 > nbr\_max$ de jours dans une semaine (cf. chapitre 5) ;

3. Créer les `PeriodicInstant` de début et de fin (si nécessaire).
4. Ordonner les `AbstractTemporalExpression` d'un `PeriodicInstant` suivant l'ordre des unités calendaires (de la seconde à l'année). Nous utilisons l'ordre des unités calendaires de l'énumération `CalendarUnit` (cf. section 3.4.2) de PTOM.
5. S'assurer qu'il n'y ait ni sur- ni sous-spécification pour chaque `AbstractTemporalExpression` (cf. section 3.4.2.3).

6. Évaluer si un saut (*jump*) (cf. section 3.4.2) est nécessaire et l'ajouter : (#e1) *« du 1er octobre au 31 mars »* (cf. figure B.1.3), (#e2) *« de 11h à 2h du matin »* (cf. figure B.1.5). Nous faisons l'hypothèse que l'annotation TAL est correcte et que l'ordre des expressions calendaires doit être maintenu. Ainsi, (#e1) devient *« du 1er octobre au 31 mars de l'année suivante »* et (#e2) devient *« de 11h à 2h du jour suivant »*.
7. Créer le `FreqWithCalendarRef` ou `PeriodicTimeInterval` si nécessaire.
8. Créer l'événement.

Les sorties de la transformation sont des expressions en intension qui sont des instances de `FreqWithCalendarRef` ou de `PeriodicTimeInterval`.

Même si des exceptions sont levées au cours de l'exécution, celle-ci va à son terme sauf si bien sûr les expressions calendaires sont inexistantes, ce qui peut se produire lorsque l'annotation n'a pas fonctionné. Si des exceptions sont levées lors de la transformation (e.g. : lorsque des invariants sont violés), alors elles sont retranscrites à l'utilisateur sous forme de diagnostic textuel pour l'informer des problèmes rencontrés. À partir de ces informations, celui-ci peut compléter les modèles générés si nécessaire.

Nous donnons en annexe des résultats d'exécution de la transformation pour les exemples 3 et 81 du corpus RMM2 : figures F.1.3 et F.1.4 pour le 3 et F.2.3 et F.2.4 pour le 81.

- Pour l'exemple 3, deux événements ont été générés du fait que la phase d'annotation a également généré deux `AccessPeriod` distinctes.
- Pour l'exemple 81, deux `PeriodicRule` ont été ajoutées à la propriété temporelle de l'événement car la phase d'annotation a abouti à la définition de deux `specification`.

Nous fournissons également en annexe les contreparties textuelles pour ces deux exemples (cf. figures F.1.5 et F.2.5).

La section 7.5.4 décrit l'expérimentation cette transformation sur le corpus des expressions temporelles fournies par Relaxnews.

```
aspect class CalendarExpression {
  inv notEmpty is do
    !self.year.isVoid() or [..] or self.partOfDay.name!="none"
  end
}
```

FIGURE 6.2.2: Invariant (en KERMETA) sur une `CalendarExpression` afin de vérifier qu'elle n'est pas vide

6.2.3 Grammaire d'horaires d'accessibilité

6.2.3.1 Objectifs

Nous nous intéressons non seulement aux corpus de Relaxnews mais aussi à d'autres provenant de plusieurs dépôts de Données Ouvertes (Open Data). Lors de l'étude de ces corpus, nous avons examiné les différents types d'expressions temporelles utilisées. Nous avons remarqué la présence non négligeable d'expressions ne décrivant que des horaires d'ouvertures périodiques sans même spécifier les jours d'ouverture.

Ces horaires représentent une restriction des périodes d'accessibilité. Par exemple : « *de 9h30 à 12h30* », « *22:30 - 06:00* », « *20h* », « *19:00 | 20:00 | 21:00* ». Afin d'identifier ces expressions et de proposer une approche simple pour les analyser afin d'en déduire des événements, nous avons défini une grammaire Xtext spécifique pour les horaires d'accessibilité. Le résultat des analyses des textes permet de produire des modèles instanciant le Métamodèle de Période d'Accessibilité (APM). Ces modèles conformes à APM sont ensuite transformés en modèles Ptom.

Cette grammaire est nommée Stnl (Simplified Temporal Natural Language). Avant de la présenter, nous motivons son usage et caractérisons les types d'expressions qu'elle doit permettre d'analyser en section 6.2.3.2. Ensuite nous présentons les corpus étudiés en section 6.2.3.3, puis nous expérimentons Stnl sur ces corpus en section 6.2.3.4).

6.2.3.2 Expressions à analyser

Nous nous sommes restreints à quatre classes d'expressions :

1. instant périodique (InstP) composé d'un nombre d'heures et de minutes, e.g. : « *20h* », « *20h50* », « *20:50* » ;
2. instants périodiques multiples (InstPM) composé de plusieurs instants périodiques séparés par une chaîne de caractères, e.g. : « *19:00 | 20:00 | 21:00* » ;
3. intervalle périodique (ItrvP) composé d'un instant périodique de départ et d'un instant périodique de fin, e.g. : « de *9h30 à 12h30* » ;
4. intervalles périodiques multiples (ItrvPM) composé de plusieurs intervalles périodiques séparés par une chaîne de caractères, e.g. : « *10h à 12h et 14h à 18h* ».

Ces expressions sont analysées individuellement. L'objectif n'est pas de détecter un instant périodique dans une phrase comme un outil de TAL pourrait le réaliser, mais d'analyser un texte composé uniquement d'instants ou d'intervalles périodiques pour en déterminer des propriétés, e.g. : nombre d'heures, de minutes. Ainsi notre approche nécessite un pseudo formatage ou bien un pré-traitement éliminant les éléments non interprétables ne se rapportant pas à de l'information temporelle, e.g. : « *En semaine de 10h à 12h et 14h à 18h* ».

Cette restriction permet de conserver toute de la sémantique du texte original après analyse. Le résultat, sous forme de modèle EMF, est interprétable par une machine et strictement conforme aux exigences de l'utilisateur qui a spécifié l'expression concernée.

Cette restriction évite le rejet de portion de texte telle que « *samedi et dimanche* » dans la phrase « *Ouverture des Ateliers de 11h à 19h samedi et dimanche* ». En effet, si les jours de la semaine sont rejetés, alors le résultat de l'analyse sera partiel ce qui n'est pas satisfaisant. Par conséquent nous considérons que cette expression est à traiter avec d'autres techniques plus adaptées.

Dans la suite, nous donnons des statistiques sur les expressions rencontrées pour quatre corpus représentatifs et variés en terme de types d'expressions.

6.2.3.3 Corpus étudiés

Nous avons utilisé STNL sur quatre corpus d'expressions temporelles toutes en intension décrivant des périodes d'accessibilité avec des niveaux différents de formatage :

1. (c1) Le premier corpus est issu des données ouvertes de la Mairie de Paris [1] proposant la géolocalisation et les horaires d'ouverture de 327 kiosques à journaux parisiens. Les données concernant ces horaires sont isolées dans des champs, de plus elles sont toutes formatées de la même manière et sont uniquement des instants périodiques (InstP) tels que : « 08h30 » (InstP). Le formatage choisi par les concepteurs de ces données ouvertes est un fichier CSV (Comma Separated Values) dans lequel les horaires sont répartis sur trois colonnes, mais seules les deux premières sont exploitées avec des valeurs quantitatives (`OUV.` et `FERM.`), alors que la troisième (`OUV. DIM`) donne une information qualitative permettant de savoir si le kiosque est ouvert le dimanche toute la journée (« *OUI* ») ou uniquement le matin (« *MAT* »). Des exemples sont fournis dans le tableau 6.1.

ADRESSE	ARRDT	OUV.	FERM.	OUV. DIM
PCE DU CHATELET	75001	08H30	20H00	OUI
2 BLD DU PALAIS	75001	08H00	19H30	MAT
104 RUE DE RIVOLI	75001	08H00	20H00	-

TABLEAU 6.1: Extrait du corpus des horaires des « kiosques presse » parisiens fourni par Paris Data

2. (c2) Le deuxième corpus provient également de données ouvertes de la Mairie de Paris [2] donnant les horaires de 94 marchés découverts ou couverts parisiens. Les horaires sont isolés dans des champs et sont tous formatés de la même manière, e.g. : soit « *07h00 à 14h30* » (ItrvP), soit « *09h00 à 13h00 et 16h00 à 19h30* » (ItrvPM). Avant d'être mises à disposition *via* le dépôt de données ouvertes, ces

1. `http://opendata.paris.fr/opendata/jsp/site/Portal.jsp?document_id=101&portlet_id=102`
2. `http://opendata.paris.fr/opendata/jsp/site/Portal.jsp?document_id=134&portlet_id=102`

informations ont été vérifiées et formatées pour être interprétables à la fois par l'homme et la machine, par conséquent leur analyse en est facilitée. Le format CSV a également été choisi, les horaires sont répartis sur huit colonnes : sept pour les jours de la semaine et une pour les jours fériés. Ceci permet facilement de déduire la période de répétition des horaires affectés à un jour particulier, i.e. une semaine pour les sept premières colonnes. Des exemples sont fournis dans le tableau 6.2.

Type	Marché	Localisation	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche	Jour férié
Alimentaire découvert	AGUESSEAU	Place de la Madeleine, Bd Malherbes	-	07h00 à 14h30	-	-	07h00 à 14h30	-	-	-
Alimentaire couvert	BEAUVAU	Place d'Aligre	-	09h00 à 13h00 et 16h00 à 19h30	09h00 à 13h00 et 16h00 à 19h30	09h00 à 13h00 et 16h00 à 19h30	09h00 à 13h00 et 16h00 à 19h30	09h00 à 13h00 et 15h30 à 19h30	09h00 à 13h00	-
Timbres	CARRE MARIGNY	Avenue de Marigny et avenue Gabriel	-	-	09h00 à 19h00	-	09h00 à 19h00	-	09h00 à 19h00	09h00 à 19h00

TABLEAU 6.2: Extrait du corpus des horaires des marchés parisiens fourni par Paris Data

3. (c3) Le troisième corpus est issu des données ouvertes du Ministère de la Culture et de la Communication [3] hébergé par la plate-forme française d'ouverture des données publiques nommée Etalab [4]. Ce corpus liste « des événements culturels et des organismes producteurs d'événements en France et également des grandes manifestations en France et à l'étranger ». Ce corpus est composé de 35252 offres de lieux. Parmi les champs disponibles, on trouve le `Libellé` de l'organisme, l'`Adresse` et plus particulièrement les dates et horaires auxquels se déroulent ces événements représenté par le champ `les horaires`, une date de début de validité `date début`, une date de fin de validité `date fin`. Les données sont isolées à l'intérieur de champs, mais les horaires ne sont pas toujours définis. Le formatage est hétérogène

3. `http://www.data.gouv.fr/donnees/view/Agenda---Offres-culture-2011-30382214`
4. `http://www.etalab.gouv.fr/`

du fait de l'utilisation d'un texte libre sans contrôle. En effet, les informations sont saisies soit par l'organisme gérant l'offre ou bien par les services du Ministère de la Culture. La complexité va d'un simple horaire en intension « *20h* » à une expression possédant une étendue temporelle avec des horaires et jours périodiques et des exceptions : « *Du 1/5 au 30/9 : tous les jours sauf samedi et lundi de 10h à 12h et de 14h à 19h - Fermeture annuelle : Du 1/10 au 30/9 - Visite sur rendez-vous : Oui* ». Des exemples sont fournis dans le tableau 6.3.

libelleOrg	libelleOffre	libelleLieu	adresse CommuneOrg	horaire	periode DateDebut	periode DateFin
ENTRE COUR ET JARDIN	Rendez-vous aux jardins	-	Sézanne	14h -18h	04/06/2011	05/06/2011
TRESOR DE LA VILLE DE RIBEAU-VILLE	Collection	MUSEE MUNICIPAL	Ribeauvillé	Du 1/5 au 30/9 : tous les jours sauf samedi et lundi de 10h à 12h et de 14h à 19h Fermeture annuelle : Du 1/10 au 30/9 Visite sur rendez-vous : Oui	-	-
TRANS EUROPE THEATRE	Personne ne m'aurait cru, alors je me suis tu	-	Lésigny	19h	18/01/2011	30/01/2011

Tableau 6.3: Extrait du corpus de l'Agenda des Musées fourni par le Ministère de la Culture et de la Communication

4. (c4) Le quatrième corpus est fourni par l'agence de presse Relaxnews qui se compose de 513 expressions extraites manuellement de dépêches en langage naturel gérées par Relaxnews. Elles ont été choisies pour leur pertinence par l'agence elle-même. Elles représentent des périodes d'accessibilité à des lieux publics en lien avec des activités de loisirs. Des exemples sont fournis dans le tableau 6.4.

id	expression temporelle en langage naturelle
1	10h à 19h du lundi au samedi
2	7 jours sur 7, de 12h à 1h du dimanche au jeudi soir, vendredi et samedi jusqu'à 1h30 Petit-déjeuner de 8h30 à 10h30
5	De 10 h à 18 h (19h le dimanche)

Tableau 6.4: Extrait du corpus Relaxnews RelaxMultiMedias 2

6.2.3.4 Expérimentation de STNL

L'expérimentation de STNL a été réalisée sur les expressions des quatre corpus, les résultats sont donnés dans le tableau 6.5. La grammaire de STNL est fournie en annexe E.3.1. Les modèles issus de l'analyse avec STNL sont ensuite transformés en instances d'APM, puis en instances de PTOM pour devenir des événements complétant notre base d'événements.

Résultats \ Corpus	Kiosques (c1)	Marchés (c2)	Musées (c3)	Relaxnews RMM2 (c4)
Total corpus	327	94*8=752	35252	513
Expressions spécifiées	327 (100%)	252 (33,5%)	30805 (87,4%)	513 (100%)
Expressions manquantes	0	500 (66,5%)	4447 (12,6%)	0
Parmi les expressions spécifiées dans les corpus				
Expressions interprétées	327 (100%)	252 (100%)	18424 (59,8%)	12 (2,3%)
Non interprétées	0	0	12381 (40,2%)	501 (97,7%)
Exception (présence du terme « sauf »)	0	0	635 (2,1%)	59 (11,5%)
Parmi les expressions interprétées par STNL				
Instant Périodique (InstP)	327 (100%)	0	11306 (61,4%)	0
Instant Périodique Multiple (InstPM)	0	0	373 (2%)	0
Intervalle Périodique (ItrvP)	0	233 (92,5%)	6561 (35,6%)	12 (100%)
Intervalle Périodique Multiple (ItrvPM)	0	19 (7,5%)	184 (1%)	0
Marqueur hebdomadaire	0	252 (100%)	441 (1,4%)	-

TABLEAU 6.5: Résultats des analyses avec STNL sur quatre corpus d'expressions de périodes d'accessibilité

Les résultats montrent qu'avec STNL, nous sommes capables d'interpréter la totalité de corpus (c1 et c2) lorsque ceux-ci possèdent un formatage quasi strict. Lorsque les expressions sont plus hétérogènes et possèdent des expressions utilisant des phrases en langage naturel, les résultats sont moins bons. Malgré l'hétérogénéité du corpus des « musées », STNL a interprété près de 60% des expressions exprimées. Pour le corpus 4, la faiblesse des résultats était attendue, les expressions sont issues de textes de dépêches en langage naturel.

Les marqueurs hebdomadaires sont des informations complémentaires que l'on trouve dans deux des trois corpus de données ouvertes (c2 et c3). Ces informations renseignent sur les jours d'occurrence des événements à l'aide d'un booléen, e.g. : « *chaque lundi* », « *chaque mardi* », etc. Pour le corpus des « marchés », cette information fait partie *de facto* de l'information à prendre en considération pour construire la règle périodique car les horaires sont spécifiés dans des colonnes qui ont pour titre le libellé d'un jour. En revanche pour les musées, si des expressions sont conformes à STNL, cela implique que l'événement a lieu tous les jours. Ainsi une incohérence peut être détectée si une des colonnes dédiées aux marqueurs hebdomadaires est à « faux ». En effet, cela signifie

que l'expression aurait dû spécifier des jours particuliers d'occurrence. Ainsi pour les musées, grâce à STNL, nous avons constaté que 1,4% des expressions interprétées étaient incohérentes car elles possèdent des marqueurs hebdomadaires non complets.

STLN est une approche pertinente pour réaliser une étude préliminaire cherchant à identifier des classes d'horaires d'accessibilité (InstP, InstPM, ItrvP et ItrvPM). Cela s'avère efficace pour des données qui ont été pré-traitées, extraites d'un texte brut, comme ceci est généralement le cas pour des corpus de données ouvertes (Open Data) dont l'objectif est de favoriser la réutilisation des données.

6.2.4 Flux RSS

De nombreux sites web (sites liés aux loisirs, billetterie...) mettent à disposition des flux RSS (Rich Site Summary) pour communiquer et annoncer des spectacles, des horaires d'ouverture d'une exposition, etc. Nous nous sommes intéressés aux flux RSS de la « FNAC Spectacle[5] » et de « l'Équipe ».

Le flux RSS de la « FNAC Spectacle » annonce des spectacles et leurs horaires. Parfois ces horaires sont en fait des expressions en intension décrivant des périodes d'accessibilité.

Afin de proposer aux journalistes ce type de sources de données, nous avons écrit une transformation de modèles pour convertir des flux RSS en événements conformes à PTOM. Cette transformation propose une première étape dans la rédaction de l'événement, nous utilisons plus particulièrement les descriptions de l'`item` de RSS, elles contiennent les informations pouvant conduire à la création de nouveaux événements. Dans un second temps le journaliste doit compléter les événements générés.

À noter que pour cet exemple de source de données, nous n'avons pas analysé le contenu des différents champs, des informations pertinentes pourraient en être extraites.

6.3 Interopérabilité avec les normes et standards

Cette section traite de l'interopérabilité entre PTOM et des normes et standards tels que iCalendar et EventsML G2. Notre objectif est de définir les règles de correspondance pour importer et exporter des instances de PTOM dans les formats précités. Nous nous focalisons sur le format iCalendar qui est très répandu et le standard EventsML G2 qui est le format utilisé par les agences de presse pour échanger entre elles.

iCalendar

Comme nous l'avons présenté dans l'état de l'art (cf. chapitre 2), iCalendar possède à la fois des propriétés structurelles (entre événements) et temporelles. Dans la suite nous donnons les correspondances pour passer de PTOM à iCalendar suivant ces deux points de vue.

5. `www.fnacspectacles.com`

Relations entre événements

Le tableau 6.6 donne les correspondances entre PTOM et iCalendar concernant les relations entre événements.

PTOM	iCalendar	Commentaires
`T_Event`	`VEVENT`	
assoc. `structural`	`RELATED-TO;RELTYPE=PARENT/CHILD/SIBLING`	relations disponibles entre `VEVENT`
assoc. `temporal`	-	non supporté par iCalendar
assoc. `semantic`	-	non supporté par iCalendar

TABLEAU 6.6: Correspondances entre PTOM et iCalendar : relations structurelles

Expression temporelle

Le tableau 6.7 donne les correspondances principales entre PTOM et iCalendar concernant les expressions temporelles définissant les dates d'occurrence des événements.

La traduction d'un `T_Event` vers un `VEVENT` n'est pas toujours bijective. En effet, si le `T_Event` possède plusieurs horaires d'ouverture à différentes périodes de l'année, e.g. : « *de 7h à 21h30 du 1*er *mars au 31 mai et de 7h à minuit du 1*er *juin 31 août...* » (cf. figure B.1.2), il est alors nécessaire de créer plusieurs `VEVENT`. Cela se traduit en PTOM par l'ajout de quatre règles périodiques contenues par le même `T_Event`, alors qu'avec iCalendar, il est nécessaire de créer quatre événements distincts en spécifiant une relation de type *frère* (`sibling`) entre eux. Cette dispersion de l'information en iCalendar peut favoriser des effets de bord sur l'interprétation de la sémantique des règles de périodicité.

PTOM	iCalendar
T_Event	VEVENT
PeriodicRule	RRULE
descripteurs du début du PeriodicTimeInterval	DTSTART
descripteurs de la fin du PeriodicTimeInterval	DTEND
attribut times	FREQ
descripteurs du début et de fin d'un PeriodicTimeSpan	BYMONTH
	BYWEEKNO
	BYYEARDAY
	BYMONTHDAY
	BYDAY
	BYHOUR
	BYMINUTE
	BYSECOND
TemporalException (TM_GeometricPrimitive)	EXDATE
TemporalException (PeriodicRule)	(1)
RelativePosition	(2)

(1) non supporté depuis la dernière version d'iCalendar
(2) non supporté par iCalendar

TABLEAU 6.7: Correspondances entre PTOM et iCalendar : propriétés temporelles périodique

EventsML G2

EventsML G2 reprend toute la partie temporelle d'iCalendar à la fois pour les dates en extension et l'expression de règles de périodicité. Les noms des attributs peuvent varier, mais les fonctionnalités restent les mêmes [6]. En revanche EventsML G2 fournit des relations supplémentaires entre événements par rapport à iCalendar. Ainsi nous traitons ci-dessous uniquement le cas des relations mutuelles entre événements.

Relations entre événements

Les relations entre événements sont gérées par les structures se trouvant dans EventsML G2 sans faire référence à iCalendar. Le tableau 6.8 présente les correspondances entre

6. http://www.iptc.org/site/News_Exchange_Formats/EventsML-G2/

Ptom et EventsML G2.

Ptom	EventsML G2	Commentaires
`T_Event`	`Event`	
assoc. `structural`	relations `broader/narrower`	relations disponibles entre `ConceptItem`
assoc. `temporal`	-	non pris en charge
assoc. `semantic`	« item relation [7] », `seeAlso`, `dependsOn`, etc	relations disponibles entre `ConceptItem`

Tableau 6.8: Correspondances entre Ptom et iCalendar : relations structurelles

6.4 Interrogation des modèles conformes à PTOM

6.4.1 Objectifs

L'objectif du système d'interrogation en intension (Ptom-s) est dans un premier temps une preuve de concept pour retrouver des événements conformes à Ptom qui sont stockés dans une base de données. Puis dans un deuxième temps, nous souhaitons interroger les corpus qui ont été traités avec les outils du TAL et nos grammaires (cf. section 6.2.3), ceci à l'aide de requêtes temporelles en intension qui répondent aux exigences d'un sous-ensemble des relations Allen*, e.g. : *equals**, *overlaps**, etc (cf. chapitre 4). Notre approche est centrée sur l'utilisation maximale de la forme intensionnelle de la temporalité. Ainsi, dans un troisième temps, nous souhaitons comparer le temps d'exécution entre une requête sur une forme intensionnelle et celle en extension.

Nous nous sommes volontairement limités et nous n'avons pas engagé de réflexion concernant un système d'interrogation qui utiliserait des déductions ou un système d'inférence. En effet, nous considérons que ceci fera partie d'un travail futur, pour lequel nous supposons que les relations qualitatives entre intervalles non convexes Allen* seraient une aide.

Ainsi dans cette section, nous présentons tout d'abord le cas d'utilisation du système d'interrogation et les requêtes qu'il permet d'exécuter (section 6.4.2). Puis, nous montrons l'interface web permettant de paramétrer une recherche (section 6.4.3). Nous proposons un comparatif entre deux techniques d'interrogation qui nous a permis de faire le choix de la technologie employée (section 6.4.4). Enfin, nous décrivons la mise en œuvre de Ptom-s (section 6.4.5) en nous appuyant sur des expressions provenant du corpus Relaxnews.

6.4.2 Cas d'utilisation et requêtes

Ce système d'interrogation s'applique principalement sur des corpus d'événements définissant des périodes d'accessibilité de lieux publics. L'objectif est de donner la possibilité à un utilisateur (journalistes, organisateurs de spectacle) d'exprimer une requête qui est composée, par exemple, d'une plage horaire et d'un intervalle de jours en intension pour lesquels il souhaite connaître les événements qui vont se dérouler, par exemple : « *Quels sont les événements qui se dérouleront tous les jours du lundi au vendredi de 10h à 19h ?* » (req1).

D'autres cas d'utilisation, qui ne seront pas traités dans la suite, pourraient prendre la forme des questions : « *Quels sont les événements qui se dérouleront de 10h à 19h le 15 mars 2012 ?* » ou bien *« tous les lundis de décembre 2012, de 10h à 19h ?* ». (req1) est à considérer comme une requête élémentaire qui servira à répondre aux deux dernières questions.

Requête sous forme de patron de règle de périodicité

(req1) est en fait un patron de règle de périodicité (`PeriodicRule`) qui est à retrouver. L'expression « *de 10h à 19h* » constitue un `PeriodicTimeInterval` alors que « *tous les jours du lundi au vendredi* » est représenté par un `PeriodicTimeSpan`. Ainsi, l'utilisateur va spécifier une règle de périodicité comme requête de recherche.

Requête élémentaire étendue aux relations qualitatives de synchronisation

Nous considérons que l'utilisateur souhaite avoir comme résultat à la fois les événements qui se déroulent effectivement « *de 10h à 19h* », mais aussi ceux qui ont une intersection avec cette plage horaire, e.g. : « *de 11h à 18h* », « *de 14h à 20h* », etc. Ces relations sont nommées par Gosselin [Becher 06] : relations de « synchronisation » et se composent de 11 des 13 relations d'Allen (i.e. : sauf *precedes* et *Preceded_by*). Nous nous limitons aux relations souhaitées par les utilisateurs : *equals*, *during*, *contains*, *overlaps* et *Overlapped_by*. Il est bien entendu possible de réaliser la même chose avec les six autres relations. Dans notre cas, nous transposons les relations de synchronisation d'$ALLEN$ vers $ALLEN^*$. Ainsi nous retenons dans $ALLEN^*$ cinq types de relations :

$$\{equals^*, during^*, Contains^*, overlaps^*, Overlapped\_by^*\}$$

Le tableau 6.9 donne la traduction des relations Allen* en opérateurs de comparaison entre les valeurs des descripteurs périodiques. Les comparaisons s'opèrent sur tous les descripteurs, mais seuls ceux du `PeriodicTimeInterval` peuvent varier. En effet, c'est le `PeriodicTimeInterval` qui détermine la relation Allen* qui peut exister entre deux intervalles non convexes. Pour traiter le problème, nous utilisons les relations d'Allen en sélectionnant et en mettant en correspondance un intervalle de chaque expression.

concept PTOM	descripteur périodique	equals*	during*	Contains*	overlaps*	Overlapped_by*
intervalle sur les horaires (time interval) considéré comme	début : heures et minutes	r=c	r>c	r<c	r>c	r<c
	fin : heures et minutes	r=c	r<c	r>c	r>c	r<c
`PeriodicTimeInterval` `PeriodicTimeSpan` sur les jours	début : jour de la semaine	r<=c	r<=c	r<=c	r<=c	r<=c
	fin : jour de la semaine	r>=c	r>=c	r>=c	r>=c	r>=c
`PeriodicTimeSpan` sur les mois	début : mois	r<=c	r<=c	r<=c	r<=c	r<=c
	fin : mois	r>=c	r>=c	r>=c	r>=c	r>=c

TABLEAU 6.9: Traduction des relations ALLEN* en opérateurs de comparaison
r : valeur de l'expression recherchée
c : valeur de l'expression du corpus

6.4.3 Interface de recherche de PTOM-S

Pour que l'utilisateur puisse définir la requête de recherche (i.e. : `PeriodicRule`), nous proposons un formulaire web [8] qui est composé de 8 champs (cf. figure 6.4.1). Les champs sont numériques et organisés par unité (horaire, jour de la semaine et mois de l'année).

La première ligne des champs de saisie concerne les informations horaires, la deuxième permet de paramétrer les journées de la semaine concernées et la troisième les valeurs de recherche sur les mois.

La première ligne renseignée constitue le `PeriodicTimeInterval` de la `PeriodicRule`, les suivantes sont des `PeriodicTimeSpan`.

À noter que pour les jours et les mois, il serait possible d'utiliser leurs noms avec un développement ultérieur. Dans le formulaire, la valeur « `1` » signifie « `lundi` » pour les jours et « `janvier` » pour les mois.

La partie inférieure de l'interface présente les résultats suivant la relation ALLEN* recherchée entre les intervalles de la requête et ceux des propriétés temporelles contenues dans la base.

8. `http://relaxmultimedia2.univ-lr.fr/ptom/index.php?r=panel/query`

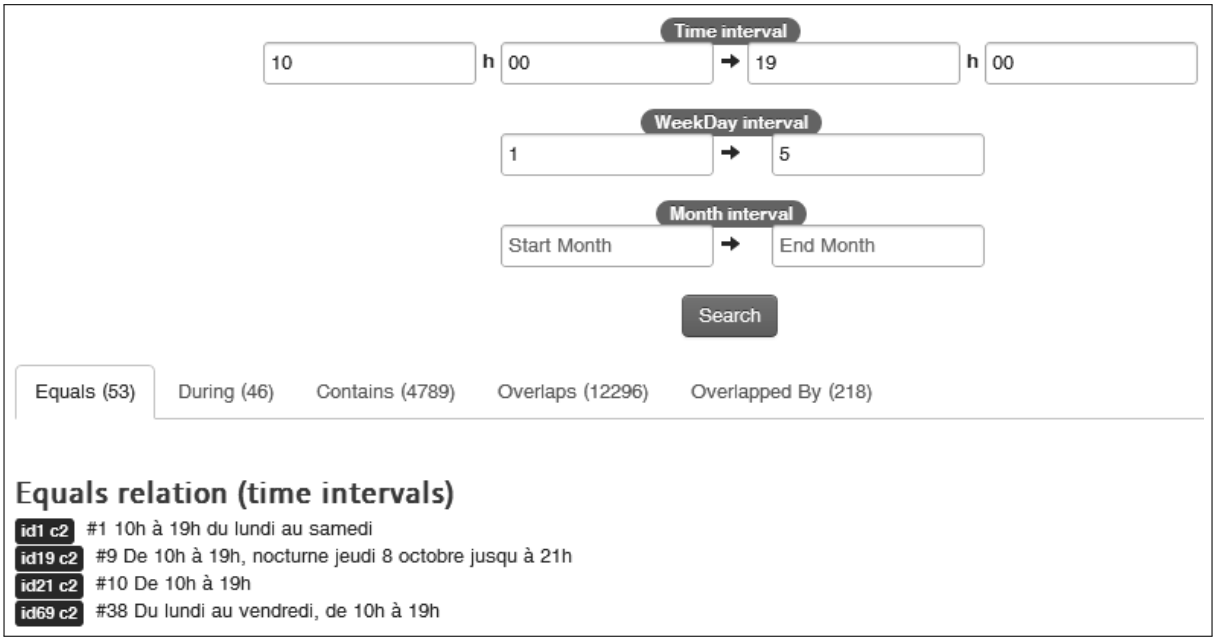


FIGURE 6.4.1: Interface de recherche et résultats de PTOM-S : « *de 10h à 19h du lundi au vendredi* »

6.4.4 Technologies d'interrogation : comparaison entre une technologie IDM et un SGBD

Même si nous avons l'intuition qu'une interrogation *via* un SGBD sera plus rapide qu'avec une technologie de l'IDM. Il nous semble pertinent de réaliser ce comparatif pour obtenir une estimation du gain et valider nos choix.

Pour choisir la technologie adéquate pour exécuter les requêtes, nous nous sommes appuyés sur des critères de performance et d'interopérabilité. Pour l'utilisateur du système d'interrogation, il est indifférent d'utiliser l'une ou l'autre et le seul critère retenu est le temps de réponse détaillé ci-dessous.

Nous comparons deux approches pour interroger des données contenues initialement dans des modèles, ici conformes à PTOM, à savoir : EMF et une base de données *ad hoc* (SGBD MySql). Les expérimentations réalisées consistent à retrouver toutes les expressions temporelles du corpus Relaxnews telles que : « *l'événement a lieu tous les jours du lundi au vendredi de 10h à 19h* ».

Approche EMF

Nous avons conçu une application en JAVA-EMF qui parcourt l'ensemble des événements du corpus, sur chacun d'entre eux nous inspectons les propriétés temporelles qui possèdent une `PeriodicRule`, puis un `PeriodicTimeInterval` dont le début est égal

à « *10h00* » et la fin à « *19h00* », enfin nous nous intéressons au `PeriodicTimeSpan`. S'il existe, alors il doit commencer par « *chaque lundi* » et se terminer au moins après « c*haque vendredi* ».

Approche base de données

Nous avons défini un schéma de base de données permettant de stocker des intervalles et des instants périodiques, ce schéma sera discuté en section 6.4.5. Une requête similaire à celle présentée en figure 6.4.2 a été utilisée pour répondre aux critères de sélection énoncés ci-dessus.

Résultats du comparatif

Le tableau 6.10 donne les temps d'exécution [9] pour les deux technologies présentées ci-dessus. Selon ces résultats, on peut en conclure que l'interrogation en utilisant les outils de l'IDM (i.e. : EMF) est plus lente que l'usage des bases de données avec un facteur de 36. Cela peut s'expliquer par les nombreux parcours de listes qui ne sont pas très optimisés avec EMF.

Les résultats montrent qu'au-delà des rapports entre les temps de réponse, l'exécution de l'application EMF est beaucoup trop longue, car elle dépasse la *seconde*, ce qui ne semble pas acceptable pour un usage dans un système d'information comme un site web.

À noter que la solution SGBD s'intègre facilement dans des systèmes d'information et elle représente un moyen très commun pour stocker et interroger des données.

Approche	Temps moyen d'exécution (10) de la requête en *s* réalisé sur un corpus de 1010 événements (100% des événements ont été retrouvés)	Intégration dans un système d'information existant	Interopérabilité avec des données externes
EMF	1,264 ($36 \times SGBD$)	faible	faible
SGBD	0,035	bonne	moyenne

Tableau 6.10: Comparatif des temps d'interrogation pour le corpus Relaxnews suivant la technologie employée

Pour conclure, nous avons fait le choix d'utiliser un SGBD qui est à la fois plus rapide et plus facilement déployable dans un système d'information existant.

9. Les exécutions ont été réalisées 10 fois pour chaque approche sur un PC « Dell Precision M2400 », processeur « Intel Core2 Duo T9800@2,93GHz » et 4Go de mémoire vive.

6.4.5 Interrogation en intension vs en extension

Objectifs et motivations

Nous allons, dans cette section, estimer le gain lié à l'utilisation d'expressions en intension en place de celles en extension. Nous avons mené une étude comparative en terme d'interrogation s'appuyant sur le corpus Relaxnews. Nous l'avons choisi, car parmi ceux à notre disposition, c'est celui qui met en œuvre les expressions en intension les plus complexes. Il possède notamment des expressions avec des horaires et des jours d'occurrences pour former des expressions telles que : « *de 10h à 19h du lundi au vendredi* ».

Pour rappel, ce corpus est composé de 513 expressions qui, après traitement, engendrent 1010 événements. Le nombre d'événements est supérieur au nombre d'expressions, car comme nous l'avons dit en section 6.2.2.2, des expressions peuvent engendrer plusieurs événements. Tous ces événements se définissent à l'aide d'intervalles périodiques ou d'instants périodiques respectivement au nombre de 1876 et 196 (soit 9,5% des expressions).

En moyenne les événements de ce corpus produisent 163 occurrences par an. Ainsi, nous proposons de générer toutes les occurrences en extension pour une année pour ensuite comparer les temps d'interrogation entre le mode intensionnel et extensionnel.

Nous mettons en œuvre cette étude comparative à l'aide du SGBD MySql. Nous choisissons une base de données pour utiliser notre solution au sein d'un système d'information d'une entreprise et également en raison des résultats obtenus en section 6.4.4.

Protocole de l'étude

Nous avons défini un schéma de base de données pouvant enregistrer à la fois des expressions temporelles en intension (cf. annexe G.1) et en extension (cf. annexe G.2). Puis nous avons généré les scripts SQL d'insertion à partir des modèles du corpus. Ainsi, nous avons créé 1010 événements correspondant à la forme intensionnelle et 164630 occurrences en extension. La proportion entre le nombre d'intervalles et d'instants a été conservée. Afin d'obtenir les temps de traitement, nous avons exécuté une requête sélectionnant l'ensemble des identifiants des événements « *qui débutent à 10h et se terminent à 19h et qui ont lieu du lundi au vendredi.* »

Mise en œuvre et résultats

Base de données d'expressions en intension

Le schéma de la base de données pour la partie en intension (annexe G.1) définit des événements, des règles périodiques, des instants et des intervalles périodiques. Chaque colonne correspond à un `AbsolutePeriodicDescriptor` de PTOM. Pour simplifier l'interrogation, nous avons spécifié une vue, nommée « *canonical_rule* », qui contient les intervalles périodiques des règles de périodicité.

Le tableau 6.11 montre trois nuplets de cette vue. L'équivalent des nuplets est également donné sous forme de texte en langage naturel directement issu du corpus Relaxnews. Selon la requête que l'on souhaite poser (req1) seules les expressions (1) et (3) seront retrouvées. En effet, les événements de l'expression (2) n'ont lieu que « *le lundi* » alors que nous souhaitons retrouver les événements qui ont lieu *« tous les jours du lundi au vendredi* ». Les valeurs `NULL` indiquent que le descripteur n'a pas été spécifié. Si une valeur est à `0`, cela signifie que le rang n'a pas été spécifié pour ce descripteur, e.g. : « *chaque jour* ».

champs de la vue	expression (1)	expression (2)	expression (3)
id_prule	1	16	21
start_second	NULL	NULL	NULL
start_minute	NULL	NULL	NULL
start_hour	10	10	10
start_day	1	1	0
start_rankday	NULL	NULL	NULL
start_week	0	0	NULL
start_month	NULL	NULL	NULL
start_rankmonth	NULL	NULL	NULL
start_year	NULL	NULL	NULL
start_century	NULL	NULL	NULL
start_code	00110000	00110000	00100000
end_second	NULL	NULL	NULL
end_minute	NULL	NULL	NULL
end_hour	19	19	19
end_day	6	1	0
end_rankday	NULL	NULL	NULL
end_week	0	0	NULL
end_month	NULL	NULL	NULL
end_rankmonth	NULL	NULL	NULL
end_year	NULL	NULL	NULL
end_century	NULL	NULL	NULL
end_code	00110000	00110000	00100000

Expressions issues du corpus RMM2 Relaxnews :
(1) « *10h à 19h du lundi au samedi.* »
(2) « *De 10h à 19h le lundi.* »
(3) « *De 10h à 19h.* »
(schéma en annexe G.1)

TABLEAU 6.11: Exemples de nuplets de la vue « *canonical_rule* »

La vue « *canonical_rule* » donne des informations complémentaires comme `start_code` et `end_code`. Ce code est un nombre en base 2 déterminant si un descripteur a été spécifié pour sa propre granularité (`heure`) ou pour une granularité supérieure. Celui-ci sera utilisé dans la requête en intension que nous explicitons ci-après (cf. figure 6.4.2).

Chaque chiffre du code correspond à une unité calendaire allant (de gauche à droite) de la seconde au siècle. Par exemple pour l'instant de début de l'expression (1) le code est : `00110000`, le 3^{e} chiffre est à `1` ce qui signifie que `start_hour(=19)` a été défini à la granularité `heure` et le 4^{e} chiffre est à `1`, ainsi `start_day(=1)` a été créé à la granularité `jour`. Si l'expression avait été : « *à 10h chaque lundi* », le code aurait été : `00010000`, seul le 4^{e} chiffre est à `1`. Ce qui diffère entre les cas, concerne l'organisation des descripteurs dans la règle de périodicité. En effet, pour « *à 10h chaque lundi* », l'ensemble des descripteurs appartiennent à la même `AbsoluteTemporalExpression` alors que pour l'expression (1) « *à 10h* » est une `AbsoluteTemporalExpression` attachée au `PeriodicTimeInterval` et « *du lundi* » est une `AbsoluteTemporalExpression` attachée à un `PeriodicTimeSpan`.

Requêtes

La requête en intension présentée sur la figure 6.4.2 est valable pour la relation Allen* *equals** comme en témoignent les sélections, e.g. : « `start_hour=10` », « `end_hour=19` », etc. Pour les quatre relations il suffit de modifier les opérateurs de comparaison que nous avons donnés dans le tableau 6.9.

Le fragment de requête SQL `substring(start_code,-5,1)=1)` est un exemple d'usage du code présenté ci-dessus pour `start_code`. Il signifie qu'on souhaite vérifier que le chiffre correspondant à la granularité `jour` est à `1`. Si tel est le cas et que, dans le même temps `start_day` est supérieur à `0`, alors l'expression originale contenait un jour de la semaine et si `start_day` est égal à `1`, alors ce jour est « `lundi` ».

```
SELECT DISTINCT r.id_event
--interrogation de la vue représentant la règle sous forme canonique et de la relation prule
FROM canonical_rule c, prule r
WHERE c.id_prule = r.id_prule
--horaire
  AND (
    (c.start_hour*60+c.start_minute) =600  --10*60 (10h00)
    OR (c.start_hour =10  AND c.start_minute IS NULL)
  ) AND (
    (c.end_hour*60+c.end_minute) =1140  --19*60 (19h00)
    OR (c.end_hour =19  AND c.end_minute IS NULL)
    OR c.end_code=0
  )
--jour de la semaine
  AND (
    (c.start_day <=1  AND substring(start_code,-5,1)=1) --lundi
    OR c.start_day=0
  ) AND (
    (c.end_day >=5  AND substring(end_code,-5,1)=1) --vendredi
    OR c.end_day=0 OR c.end_code=0
  )
--semaine, mois, année, siècle
  AND (
    (c.start_week<=1 AND substring(start_code,-4,1)=1)
    OR (
      (c.start_week=0 OR c.start_week IS NULL)
      AND (
        (c.start_month <=1  AND substring(start_code,-3,1)=1)
        OR (
          (c.start_month=0 OR c.start_month IS NULL)
          AND (
            substring(start_code,-2,1)=1
            OR (
              (c.start_year=0 OR c.start_year IS NULL)
              AND (
                substring(start_code,-1,1)=1
                OR c.start_century=0 OR c.start_century IS NULL
  )))))))
  AND (
    (c.end_week>=4 AND substring(end_code,-4,1)=1)
    OR (
      (c.end_week=0 OR c.end_week IS NULL)
      AND (
        (c.end_month >=12  AND substring(end_code,-3,1)=1)
        OR (
          (c.end_month=0 OR c.end_month IS NULL)
          AND (
            substring(end_code,-2,1)=1
            OR (
              (c.end_year=0 OR c.end_year IS NULL)
              AND (
                substring(end_code,-1,1)=1
                OR c.end_century=0 OR c.end_century IS NULL
  )))))))
  )))))));
```

FIGURE 6.4.2: Requête en intension pour la relation *equals** avec l'expression recherchée : « *de 10h00 à 19h00 du lundi au vendredi* »

La figure 6.4.3 fournit la requête en extension qui a été utilisée dans cette expérimentation. Cette requête retourne tous les événements qui ont lieu « *de 10h à 19h* » et dont

le jour d'occurrence est compris entre « *le lundi et le vendredi inclus* ».

```
SELECT DISTINCT id_eventptom
FROM eventcto
WHERE
  HOUR(start)= 10  AND MINUTE(start)= 0  AND WEEKDAY(start) BETWEEN  1  AND  5
  AND (
    (end IS NULL) OR
    (HOUR(end)= 19  AND MINUTE(end)= 0
    AND WEEKDAY(end) BETWEEN  1  AND  5  AND WEEKDAY(start)=WEEKDAY(end))
  );
```

FIGURE 6.4.3: Requête en extension recherchant des périodes : « *de 10h00 à 19h00 pour les jours allant de lundi (1) à vendredi (5)* »

Comparatif des temps de réponse : intension / extension

Le temps de réponse[10] de la requête en intension est de $35,1ms$ alors que celui de la requête en extension est de $117,8ms$. Le gain est réel car le temps d'exécution est 3,36 fois plus rapide en faveur des requêtes en intension. Il est à noter que les requêtes en extension sont réalisées sur un corpus de dates générées pour une seule année civile, alors que les événements peuvent, bien entendu, se reproduire sur plusieurs années. Par conséquent, ce rapport de temps d'exécution est amené à croître sans discontinuer.

Grâce à cette étude, nous confirmons notre hypothèse de base et constatons qu'au-delà du gain d'expressivité sémantique évident, on peut sur des arguments uniquement liés au temps d'exécution préférer les expressions temporelles en intension au lieu de stocker l'ensemble des dates d'occurrence des événements.

La section 7.6 présente une expérimentation de PTOM-S pour en préciser le mode de mise en œuvre.

6.5 Visualisation des événements à travers une application web générée

Après avoir étudié la modélisation, nous présentons dans ce chapitre une application qui exploite PTOM pour visualiser des événements *via* une interface web. L'objectif ici est de nous appuyer sur une bibliothèque de composants web et de fournir au concepteur de l'application des fonctionnalités pour faciliter le travail de conception et de mise en œuvre.

Plus particulièrement nous proposons, un mécanisme de mise à jour de modèles de composants graphiques et une génération complète des applications web visées. Dans

10. Les exécutions ont été réalisées 10 fois pour chaque approche sur un PC « Dell Precision M2400 », processeur « Intel Core2 Duo T9800@2,93GHz » et 4Go de mémoire vive.

ce chapitre, nous proposons une démarche IDM pour répondre à ces objectifs en nous appuyant sur les besoins et données du projet RelaxMultiMedias 2.

6.5.1 Motivations

Les applications web deviennent de plus en plus élaborées grâce à l'usage de bibliothèques de composants offrant des fonctionnalités avancées pour la visualisation de données. Avec ce type de bibliothèque, une partie importante du travail de développement porte sur l'intégration de données c'est-à-dire la sélection et la structuration des données fournies à ces composants. Les données visualisées dans ces applications peuvent provenir de différentes sources : bases de données relationnelles, bases de connaissance (points d'interrogation SPARQL), flots XML, JSON, etc.

Les travaux autour de l'Ingénierie du Web Dirigée par les Modèles (Model-Driven Web Engineering en anglais, MDWE) montre que la conception d'applications web doit être réalisée à un niveau d'abstraction suffisamment élevé pour manipuler des composants web sans avoir à gérer des concepts bas niveau, comme des balises « Paragraphs » ou « Anchors ». L'IDM se prête bien à cette approche orientée-composant, car les paramètres des composants peuvent être modélisés par un ou plusieurs métamodèles qui représentent des métamodèles de configuration.

Nous proposons de modéliser les différentes préoccupations liées à une application web dédiée à la visualisation d'information. Les modèles et métamodèles issus de cette réflexion de séparation des fonctions seront les variables d'entrée des générateurs de code source pour finalement obtenir de manière automatisée une application web prête à l'emploi.

Lorsqu'une nouvelle version d'une bibliothèque est publiée, de nouveaux composants peuvent apparaître ou certains peuvent être mis à jour (ex. ajout d'un nouveau paramètre de configuration). Si le développeur souhaite utiliser ces améliorations, il convient de l'aider dans le processus de mise à jour des modèles concernés. De plus, l'approche proposée permet au développeur de découvrir une bibliothèque et de disposer d'une abstraction de celle-ci. Cette contribution est à rapprocher des techniques liées à la rétro-ingénierie. Nous proposons en section 6.5.5.4 une technique de mise à jour du métamodèle de composant graphique pour faciliter la prise en charge des évolutions.

Il est rarement pertinent de visualiser l'ensemble des données disponibles. Un travail préalable de réorganisation et de sélection de données est nécessaire. Nous traiterons cette problématique en 6.5.6 où nous proposons un générateur de données au format JSON (JavaScript Object Notation) à partir de modèles EMF.

Les techniques de l'IDM assurent l'automatisation de la production de données et la génération de code source. Nous démontrons la mise en œuvre de notre approche en utilisant la bibliothèque Simile Exhibit dédiée à la publication des informations structurées à partir de serveurs web. Notre approche, nommée Modsea (MOdel Driven engineering for Simile Exhibit Application), est donc orientée-modèle comme le prouve l'utilisation de différentes facettes :

– métamodélisation pour abstraire les concepts utilisés ;
– transformations de modèles pour traduire les concepts d'un métamodèle générique vers un métamodèle spécifique ;
– composition de modèles [Clarke 01] pour construire le métamodèle de composants graphiques (`Widgets`) ;
– génération de code fondée sur des patrons prenant en entrée des modèles.

6.5.2 Approches connexes

Afin de construire des applications web de manière flexible, des approches d'ingénierie du web comme WebML [Brambilla 08], UWE [Koch 08] ou WebSA [Meliá 06] proposent une séparation des préoccupations généralement liées aux aspects de « présentation », de « contenu » et de « navigation ». L'aspect présentation correspond à la définition des composants utilisés par l'application et de leur positionnement graphique. L'aspect contenu concerne les données gérées par l'application, alors que la partie navigation définit les scénarios d'enchaînement des fonctionnalités ou des pages web. Les approches MDWE permettent de modéliser ces préoccupations séparément sous forme de métamodèles, puis de les instancier afin d'obtenir des modèles exploitables par une machine. À partir de ces modèles, des générateurs de code produisent partiellement ou complètement d'une application web.

L'émergence des technologies associées à l'IDM, plus particulièrement celles concernant la transformation de modèles et la génération de texte [Jouault 06, Eclipse 12, Muller 05, OMG 11a] facilitent cette mise en œuvre.

L'approche WebML permet de créer des applications web complexes à l'aide de modèles de présentation, de contenu et de navigation. Elle est mise en œuvre dans l'outil WebRatio qui possède des éditeurs graphiques permettant de spécifier ces modèles, puis de générer le code source de l'application web. La génération du code source est réalisée en XSLT, mais cette technologie ne permet pas facilement d'abstraire les règles de transformation contrairement aux langages dédiés à la transformation de modèles cités ci-dessus.

L'approche UWE utilise des transformations écrites en ATL. Ces transformations prennent en entrée un ensemble de modèles UML stéréotypés par le développeur.

Dans une approche MDWE, chaque préoccupation est spécifiée sous forme de modèles conçus pour être indépendants les uns des autres. Cependant, ces modèles conçus indépendamment doivent être connectés. Pour répondre à ce besoin, l'approche WebSA propose d'introduire des modèles d'intégration reliant les modèles entre eux. Cela évite de créer des dépendances entre modèles tout en facilitant la réutilisation des modèles pour créer des familles d'applications. D'autres approches comme « EMF on Rails » [López-Landa 12] utilisent une plate-forme existante à laquelle sont rajoutées des composants complémentaires définis et générés avec une approche IDM.

Le projet ANR YourCast [11] propose d'agréger dans une application web des flux d'informations (agenda, flux RSS, flot de messages Twitter). Ces informations sont à diffuser sur des écrans MultiMedias dans des lieux publics. YourCast est basé sur une approche de variabilité dans un contexte de ligne de produit (Software Product Line, SPL) [Clements 06, Chen 11], ceci à base de modèles de configuration (*Feature Model* et Feature Diagram [Schobbens 07]). Le principe consiste à proposer des modèles de configuration cohérents qui composeront l'application finale. Dans ce projet, un autre point important concerne la sélection de données et la mise à disposition de services pour intégrer des données hétérogènes. Pour notre part, nous souhaitons produire une application avec laquelle l'utilisateur interagit en mode bureau, par conséquent les problématiques d'interface graphique sont différentes.

Des travaux comme [Taddesse 10a, Taddesse 10b] traitent de la gestion sémantique de flux d'actualités en termes d'identification d'éléments reliés ou encore de mesures de similarités sémantique. En ce qui nous concerne, les ressources à visualiser proviennent d'une seule source et les données à traiter sont donc connues et présentes dans les bases de données que nous gérons ; les agrégations de contenus étant réalisées en amont lors de la création des événements par les journalistes.

6.5.3 Les bibliothèques de composants graphiques web

Il existe de nombreuses bibliothèques de composants web. Nous nous sommes intéressés plus particulièrement à celles exposant les caractéristiques de leurs composants graphiques. En effet, comme il a été précisé dans les motivations, nous souhaitons extraire des informations des bibliothèques pour favoriser la mise à jour des applications que nous allons produire. OpenLaszlo [12], Ext JS [13], Simile Exhibit [14] sont des candidates répondant à nos besoins.

OpenLaszlo utilise sur des fichiers XML qui définissent à la fois la configuration des composants, la disposition des éléments (*layout*) et la liaison avec la source de données (*data binding*). Ext JS est implémenté en JavaScript, les propriétés de configuration des composants sont exposées dans des annotations. Ces annotations et, par conséquent, la configuration est également exploitable *via* une extension de GWT [15] à travers GWT Ext GWT [16]. Que ce soit pour OpenLazlo ou Ext JS, les composants fournis sont riches, mais non spécifiquement adaptés à la présentation de données telles que des cartes ou chronologies. En revanche, Simile Exhibit offre ce type de composants, ce qui justifie notre choix de l'utiliser pour notre approche.

À noter que le W3C a proposé une recommandation traitant des composants gra-

11. `http://yourcast.unice.fr/`
12. http ://www.openlaszlo.org/
13. http ://www.sencha.com/products/extjs/
14. http ://www.simile-widgets.org/exhibit/
15. http ://code.google.com/webtoolkit/
16. http ://www.sencha.com/products/extgwt/

phiques d'Application Web [W3C 11] dont une partie est consacrée à la configuration des composants. Au meilleur de notre connaissance, nous n'avons pas trouvé de bibliothèque suffisamment aboutie implémentant cette norme pour que nous puissions l'utiliser.

Simile Exhibit

Simile Exhibit est issu du projet Simile (Semantic Interoperability of Metadata and Information in unLike Environments) [MIT 11]. C'est une bibliothèque écrite en JavaScript pour la conception d'applications web dédiées à la visualisation de données hétérogènes (*mashup*) et le développement d'interfaces internet riches (*Rich Internet Applications*, RIA). Cette bibliothèque propose plus d'une dizaine de composants graphiques (*widgets*) à savoir : des cartes (*Maps*), des chronologies multi-échelles (nommées *Timelines* dans la suite), des graphiques (*Timeplots*), des filtres par mots-clés (*Facets*), etc.

Ces composants utilisent des sources de données JSON qu'ils peuvent partager afin de synchroniser leurs contenus et offrir différents points de vue à l'utilisateur. Le système de synchronisation de la source de donnée est géré directement par Simile Exhibit sans que le développeur n'ait à la programmer. Ainsi notre travail consiste à concevoir et à mettre en œuvre des métamodèles de configuration pour générer des applications Simile Exhibit.

6.5.4 Les métamodèles pour générer une application Simile Exhibit

En amont de la génération des applications Simile Exhibit, nous proposons un ensemble de métamodèles traitant de différentes préoccupations : contenu visualisé, composants employés et disposition graphique utilisée avec leurs inter-relations (cf. figure 6.5.1). Nous ne nous sommes pas intéressés à l'aspect navigation car l'interface souhaitée est constituée d'une seule page.

Comme cela existe dans d'autres approches [Koch 08, Rossi 08] nous avons introduit un métamodèle de composants (`Widgets`) et un métamodèle de disposition (`Layout`). L'architecture de notre approche est illustrée par la figure 6.5.1. En ce qui concerne les données sources et les données à afficher, nous considérons deux métamodèles : le métamodèle d'événements (Ptom) et celui de contenu (`Content`). `Content` sélectionne parmi toutes les classes et propriétés présentes dans le métamodèle Ptom celles qui sont retenues pour être utilisées dans l'application web. Le passage de Ptom à `Content` a été mis en œuvre à l'aide d'une transformation de modèles.

Pour la liaison entre les composants et les données, nous utilisons un métamodèle de correspondance nommé `Widget_Content`. Les liens entre les composants et les définitions de disposition sont stockés comme instances du métamodèle `Widget_Layout`. Les métamodèles `Layout`, `Widget_Content` et `Widget_Layout` sont détaillés dans l'annexe H.2. Cette architecture rend facilement réutilisable chacune des instances des métamodèles pour produire des familles de configuration sans nécessiter la recréation totale des modèles.

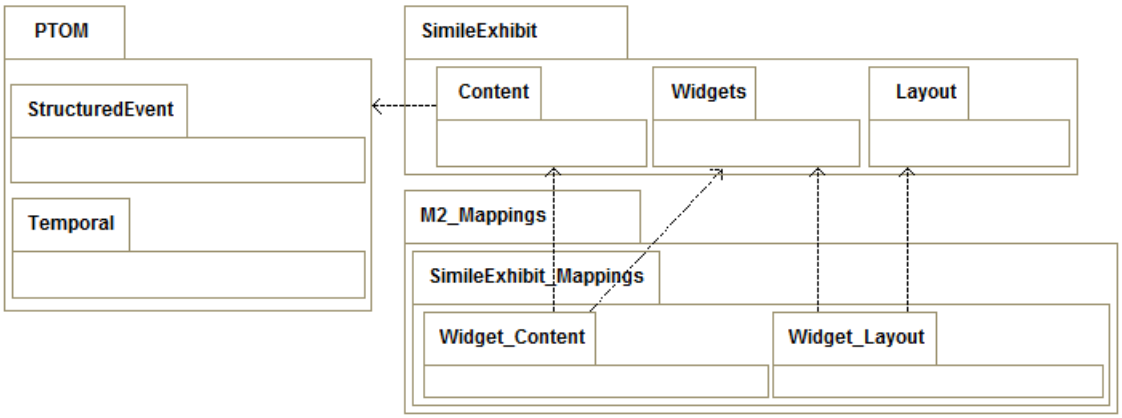


FIGURE 6.5.1: Métamodèles pour la publication d'événements

6.5.5 Métamodèle de composants graphiques : « Widgets »

Le métamodèle `Widgets` permet de configurer les composants utilisés dans l'application web. Le métamodèle de composants est dual, une partie est consacrée aux propriétés communes (métamodèle `CWidget`) et l'autre partie concerne les attributs spécifiques (métamodèles `SWidget`). La partie spécifique (`SWidget`) est représentée par un ensemble de métamodèles de composants définissant chacun les propriétés spécifiques du composant concerné.

6.5.5.1 Les éléments communs : « CWidget »

Le métamodèle `CWidget` (cf. figure 6.5.2) contient les propriétés communes aux widgets. Nous distinguons deux parties : celle de gauche pour la hiérarchie de `Widget`, qui rassemble les classes des composants graphiques que l'utilisateur va pouvoir configurer ; celle de droite est composée de classes relatives aux composants du système Simile Exhibit (hiérarchie de `HiddenComponent`).

La classe `Widget` est la racine de l'arborescence des composants graphiques, celle-ci se spécialise en différents types de composants comme `Timeline`, `Filter`, etc. Chaque type de composant possède une classe abstraite qui sera spécialisée afin de modéliser tous les composants graphiques de la bibliothèque.

Les composants se décomposent en deux catégories liées à leur fonction (cf. figure 6.5.2]) :

- les `Facets` qui filtrent les données par mots-clés, dates, ...,
- les `Views` qui affichent les données sous forme de `Thumbnail`, `Table`, `Maps`, `Timelines` et `Timeplots`.

Une `Facet` possède une source de données sur laquelle un filtre va être appliqué, par exemple un nom d'événement. Une `Facet` peut être également un `TextSearch` ou bien un `Calendar` avec lequel il sera possible de définir une étendue temporelle pour restreindre les données suivant leurs dates d'occurrence.

Une `View` possède un `Lens` définissant une description pour un objet. Le contenu du `Lens` s'écrit en HTML et peut être affiché sous forme de bulle d'information dans le cas de `Map` et `Timeline`.

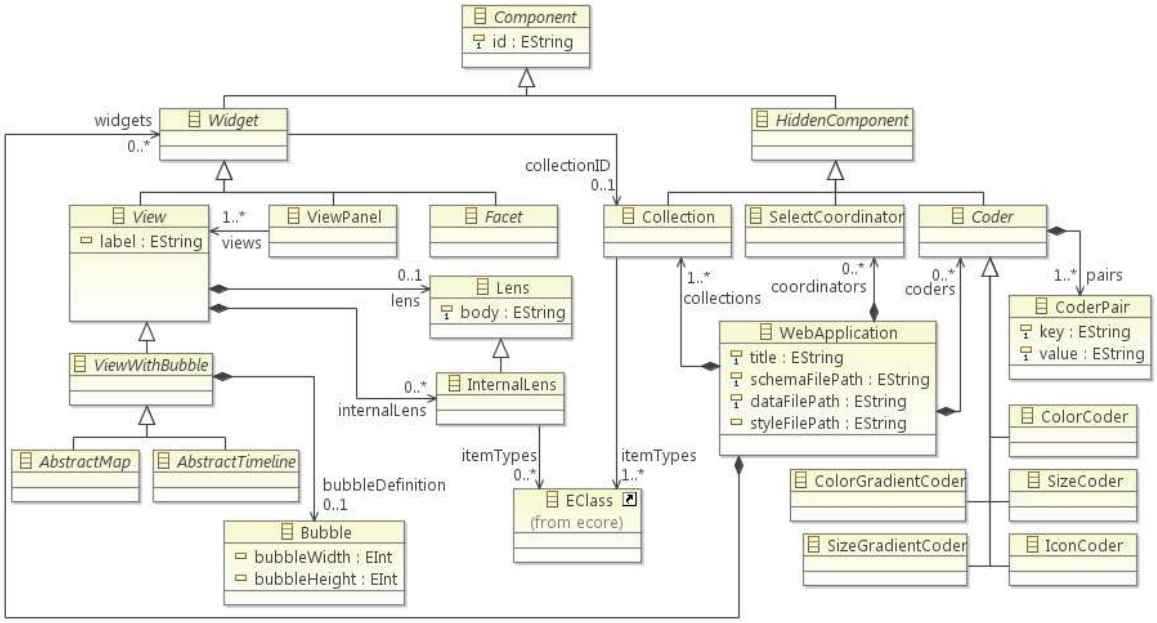


FIGURE 6.5.2: Métamodèle des éléments communs à tous les composants Simile Exhibit : `CWidget`

6.5.5.2 Les éléments spécifiques : « SWidgets »

Chaque composant graphique possède son métamodèle (un `SWidget`). Nous présentons en figure 6.5.3 le `SWidget` pour le composant de type chronologie : `TimelineView`.

Une `TimelineView` est un type de composant graphique dédié à l'affichage de chronologie alors, celle-ci spécialise classe `AbstractTimelineView`.

La classe `TimelineView` contient les attributs spécifiques comme sa hauteur, des unités, etc. À noter, les références vers des classes du `CWidget` comme `ColorCoder` et `IconCoder` qui permettent d'associer respectivement des couleurs et des icônes selon des valeurs d'attributs.

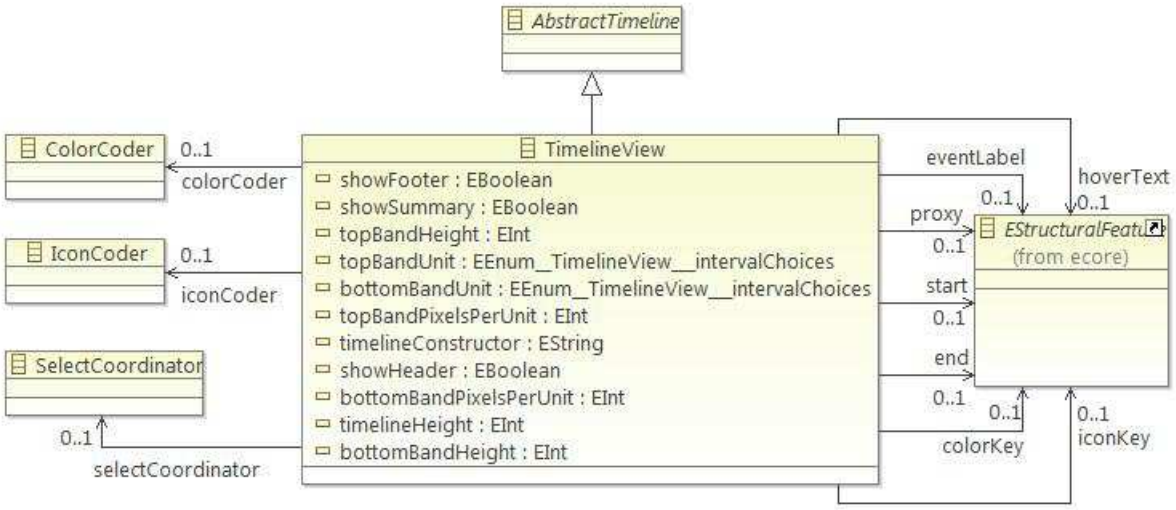


FIGURE 6.5.3: `SWidget` pour le composant de type chronologie : `TimelineView`

6.5.5.3 Création des métamodèles SWidget par extraction

Comme nous l'avons énoncé dans nos motivations (section 6.5.1), lorsqu'une nouvelle version de bibliothèque est publiée, des composants existants peuvent être mis à jour ou détruits alors que de nouveaux peuvent être créés. Si le développeur souhaite utiliser ces évolutions, il convient de mettre à jour `Widgets`. Si les nouvelles versions sont fréquentes, il est difficile de réaliser des vérifications continues et de modifier manuellement le métamodèle `Widgets`. Ainsi une extraction automatique des propriétés de configuration est à proposer pour maintenir le métamodèle `Widgets` à jour.

Cette section a pour objectif de décrire notre approche de génération des `SWidgets` comme celui de `TimelineView`, tout d'abord en procédant à une extraction d'information puis par l'utilisation de la promotion de modèles comme le montre la figure 6.5.4.

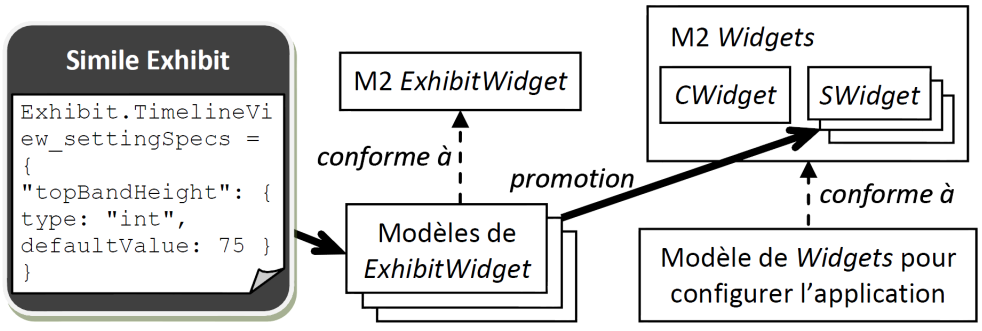


FIGURE 6.5.4: De la spécification de la bibliothèque Exhibit au modèle `Widgets` de configuration de l'application web

Extraction des attributs de présentation à partir de la description JSON des widgets de Simile Exhibit

Le développement de bibliothèques comme Simile Exhibit ou Ext JS est réalisée en JavaScript. Pour ces bibliothèques, un composant peut être défini par une série de propriétés à savoir : dans un objet JSON pour Simile Exhibit (cf. figure 6.5.5) ou bien encore dans des annotations dans le cas d'Ext JS (cf. figure 6.5.6). Ces structures et informations permettent à un programme d'analyser les composants et d'extraire suffisamment d'informations pour en déduire un modèle de configuration pour chacun des composants (`SWidgets`). Ainsi nous proposons de faciliter l'évolution du métamodèle `Widgets` en générant ses parties spécifiques `SWidget` par l'analyse automatique des composants. Nous présentons un second exemple avec Ext JS pour démontrer le caractère générique de notre approche qui n'est pas limité à la seule bibliothèque Simile Exhibit. Nous ne détaillons cependant que l'application à Simile Exhibit dans la suite.

```
Exhibit.TimelineView._settingSpecs = {
  "topBandHeight": { type: "int", defaultValue: 75 },
  "topBandUnit": { type: "enum", choices: Exhibit.TimelineView._intervalChoices },
  "colorCoder": { type: "text", defaultValue: null },
  "selectCoordinator": { type: "text", defaultValue: null },
};
Exhibit.TimelineView._intervalChoices = [
  "millisecond", "second", "minute", "hour", "day", "week",
  "month", "year", "decade", "century", "millennium"
];
Exhibit.TimelineView._accessorSpecs = [
{
  accessorName: "getProxy",
  attributeName: "proxy" },
{
  accessorName: "getDuration",
  bindings: [
    { attributeName: "start", type: "date", bindingName: "start" },
    { attributeName: "end", type: "date", bindingName: "end", optional: true }
  ]
},{
  accessorName: "getColorKey",
  attributeName: "colorKey",
  type: "text"
}];
```

FIGURE 6.5.5: Extrait des descriptions JSON pour le composant graphique `TimelineView` : `settingSpecs` et `accessorSpecs`

```
@cfg {Boolean} rootVisible false to hide the root node (defaults to true)
@cfg {Object} dragConfig Custom config to pass to the {@link Ext.tree.TreeDragZone} instance

@cfg {String} hlColor The color of the node highlight (defaults to C3DAF9)
```

FIGURE 6.5.6: Extrait de la configuration pour un objet `TreePanel` d'Ext JS

Nous devons tout d'abord connaître et spécifier l'information qui doit être extraite des descriptions JSON. Ainsi nous catégorisons les différentes propriétés selon leur usage. À noter que les descriptions définissent des attributs (e.g. : `topBandHeight`) accompagnés d'un type (e.g. : `int`) et le cas échéant d'une valeur par défaut (e.g. : `75`). Pour abstraire cette catégorisation et la structuration des propriétés, nous avons défini un métamodèle nommé `ExhibitWidget` (cf. 6.5.7) qui *de facto* abstrait les parties configurables (variables) des composants de la bibliothèque Simile Exhibit. Ce métamodèle exprime les différents types de composants graphiques, i.e. `WidgetType` (des instances de `WidgetType` seront `Timeline`, `Filter`, etc) et les différents types de propriétés de configuration que peut posséder un tel composant :

- Les paramètres de forme, d'affichage, de dimension du widget comme les attributs (e.g. : `topBandHeight`, `topBandUnit`...). Nous les nommons `WidgetAttribute`.
- Un composant graphique possède des références vers des composants internes à Simile Exhibit (e.g. : `colorCoder`), que nous appelons `InternalComponentReference` (comme vu précédemment avec `TimelineView`).
- Un composant graphique contient des références vers des sources de données externes (e.g. : `start`, `end`). `start` et `end` sont en fait des références dérivées vers des attributs du métamodèle `Content` (`ContentFeature`). Ces références se nomment `ExternalDataReference`.

Au sein du code JavaScript, ces propriétés sont réparties dans deux définitions JSON (cf. figure 6.5.5) :

- `settingSpecs` pour les attributs de `Widget` (`WidgetAttribute`) et les références internes (`InternalComponentReference`). On pourra noter l'utilisation d'une énumération avec `Exhibit.TimelineView._intervalChoices` qui remplace un type de donnée en fournissant un ensemble de valeurs acceptables.
- `accessorSpecs` pour les références vers des propriétés du modèle de `Content` (`External-DataReference`).

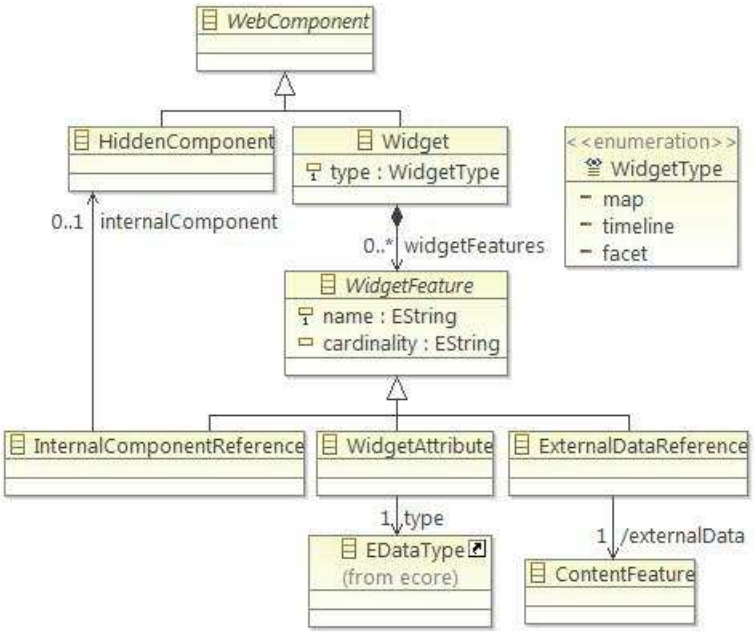


FIGURE 6.5.7: Métamodèle `ExhibitWidget` qui abstrait les propriétés des composants graphiques de Simile Exhibit

L'analyse des définitions JSON est réalisée avec une bibliothèque JAVA (JSON [17]). Le résultat de cette opération est un ensemble de modèles conformes au métamodèle `ExhibitWidget`. À partir de ces modèles une transformation de modèles de type « promotion [18] » est appliquée pour obtenir des métamodèles `SWidget`.

Génération des métamodèles SWidget

À partir des propriétés analysées et catégorisées, chaque `SWidget` est généré par un mécanisme de promotion de modèles, dernière étape du processus de la figure 6.5.4. Nous utilisons la promotion de modèle car nous sommes ici capables de traduire la totalité des concepts du métamodèle de configuration de composants en concepts Ecore. Chaque `Widget` est représenté par une `EClass` Ecore, le tableau 6.12 donne les correspondances entre les métamodèles `ExhibitWidget` et `SWidget`.

17. http ://www.json.org
18. Un modèle est transformé pour devenir un métamodèle.

ExhibitWidget	SWidget	Exemple	Commentaires
Widget	ecore::EClass	TimelineView	l'instance de Widget devient la classe racine du métamodèle SWidget
HiddenComponent	ecore::EClass	ColorCoder	chaque instance de HiddenComponent devient une classe similaire avec son homonyme du CWidget, elles seront par la suite composées
WidgetAttribute	ecore::EAttribute	topBandHeight, topBandUnit	chaque instance de WidgetAttribute devient un attribut de la nouvelle EClass racine
InternalData-Reference	ecore::EReference	colorCoder	chaque instance d'InternalDataReference devient une référence de la classe racine vers une classe du CWidget
ExternalData-Reference	ecore::EReference	start, end, proxy	chaque instance d'InternalDataReference devient une référence dérivée de la classe racine vers un attribut du métamodèle Content

TABLEAU 6.12: Correspondances entre les métamodèles ExhibitWidget et SWidget

Le métamodèle TimelineView de la figure 6.5.3 est un résultat de cette transformation par promotion.

Les types, les cardinalités, les valeurs par défaut et les énumérations sont également pris en compte. À noter que les types de données utilisés par Simile Exhibit sont compatibles avec ceux d'Ecore. Les métamodèles SWidgets seront ensuite intégrés au métamodèle Widgets par compositions de modèles successives (cf. 6.5.5.4).

La validité de cette approche est dépendante de l'implémentation JavaScript des widgets Exhibit, qui doit continuer à suivre les mêmes règles, i.e. : la répartition des attributs dans des descriptions JSON : settingSpecs et accessorSpecs. Si des nouvelles règles devaient apparaître, il serait nécessaire de modifier les règles d'extraction des propriétés de configuration.

6.5.5.4 Construction du métamodèle Widgets par composition de modèles successives

Comme annoncé dans les motivations, afin de rendre MODSEA adaptable aux futurs ajouts et améliorations de Simile Exhibit, la construction du métamodèle `Widgets` est réalisée avec une approche générative : la composition de modèle. Le métamodèle `Widgets` est fondé initialement sur les éléments communs du métamodèle `CWidget` (cf. 6.5.2 : `Widget`, `View`, `Lens`) et de onze métamodèles `SWidgets` issus respectivement de onze composants Exhibit. Les métamodèles de `SWidget` sont intégrés successivement au métamodèle `Widgets`.

Cette intégration est réalisée à l'aide du moteur de composition de modèle KOMPOSE [Fleurey 07] présenté en section 3.2.4. Pour mémoire, KOMPOSE utilise une composition de modèle par fusion qui identifie les concepts communs aux modèles à composer. Cette identification utilise une approche fondée sur les signatures [Reddy 05]. Dans notre cas nous souhaitons fusionner des modèles. Ainsi les signatures utilisées seront celles des classes ECORE ce qui équivaut à leur nom qualifié (`packageName::ClassName`). Pour l'utilisateur final aucune opération de mise en correspondance n'est à réaliser, elle est assurée en totalité par KOMPOSE. En amont lors de la création des `SWidgets`, pour rendre plus aisée l'opération d'identification, nous avons veillé à ajouter et à spécialiser la classe abstraite correspondant au type de composant tel que `AbstractTimeline` (cf. annexe H.1).

La figure 6.5.8 montre un extrait du résultat du processus de composition de modèles pour le composant `TimelineView`. Ainsi, le métamodèle `SWidget` dédié à `TimelineView` a été intégré dans `Widgets` en utilisant les points de jonctions représentés par les classes Ecore : `AbstractTimeline`, `IconCoder` and `ColorCoder` (éléments sur fond noir). Ce sont les signatures de ces trois classes qui permettent l'identification des points communs entre le métamodèle global `Widgets` et le métamodèle à intégrer `SWidget`. La classe `TimelineView`, la relation d'héritage d'`AbstractTimeline` et les références `iconCoder` et `colorCoder` (éléments en gras) ont été ajoutées par KOMPOSE.

Lorsqu'une nouvelle version de Simile Exhibit est disponible, le processus de génération de `Widgets` peut être lancé afin de maintenir la cohérence. Aucune modification complémentaire n'est nécessaire.

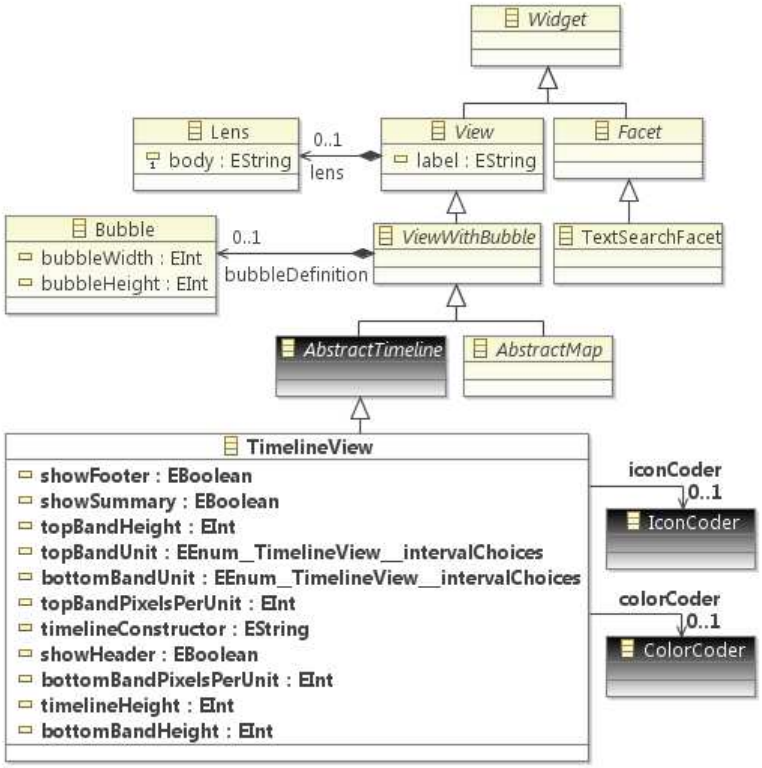


FIGURE 6.5.8: Résultat de la composition de modèles pour le composant `Timeline`

6.5.6 Directives de filtrage sur des propriétés pour transformer des modèles EMF en JSON

Problématique

De nombreuses applications, web notamment, traitent des données au format JSON (JavaScript Object Notation [Crockford 06]). JSON permet de structurer des données de manière moins verbeuse que le format XML.

La bibliothèque Simile Exhibit qui est utilisée dans MODSEA nécessite en entrée des données au format JSON. Les données que nous manipulons se présentent intialement sous forme de modèles EMF et sont des instances de PTOM. Ainsi, nous devons proposer une solution pour générer du JSON à partir de modèles EMF.

La génération du JSON à mettre en œuvre dans notre cas possède quelques particularités qui ne sont pas permises par une solution applicative comme EMFJs [19]. En effet, Simile Exhibit affiche des données mais il est rarement pertinent de visualiser l'ensemble des données disponibles. Des références Ecore de type composition sont souvent utilisées pour structurer le métamodèle, mais elles n'ont pas d'intérêt à être visualisées. Un tra-

19. `http://marketplace.eclipse.org/content/EMFjs`

vail de réorganisation et de sélection de données est alors nécessaire. Pour répondre à ce besoin, nous proposons de générer un flux JSON à partir d'un modèle EMF. Cette génération est paramétrable par le développeur de l'application, elle utilise les techniques de l'IDM. Seules les données nécessaires et suffisantes à l'application Simile Exhibit seront ainsi générées.

Mise en œuvre

Ce convertisseur nommé EMF2JSON a été développé en Kermeta. Ce programme est indépendant du métamodèle associé au modèle traité, ainsi la conversion est générique pour tout modèle EMF. Le résultat de la conversion est un flux texte en JSON (cf. figure H.4.2) utilisable dans les applications web.

Afin de donner de la souplesse pour générer le JSON souhaité, nous avons défini quatre directives sous forme d'annotations utilisées par le programme Kermeta pour générer les données. Ces directives s'appliquent aux classes et propriétés du métamodèle contenant les données, i.e. `Content`. Dans les exemples qui suivent nous prenons l'exemple d'un métamodèle `Content` utilisé lors de RMM2 (cf. figure 6.5.9).

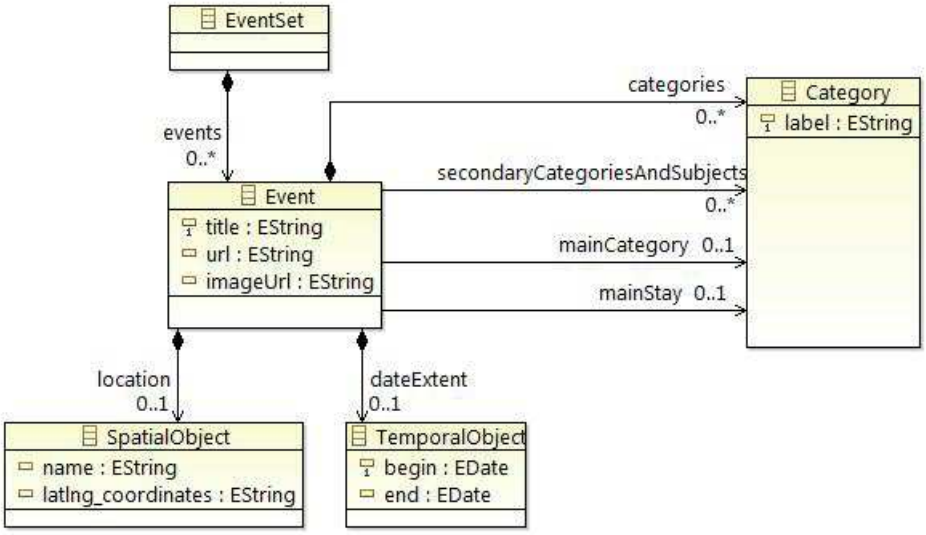


Figure 6.5.9: Métamodèle `Content` utilisé pour RMM2 sur lequel les directives s'appliquent

Ainsi, les quatre annotations (cf. 6.5.10) sont :

1. `isItemType` : désigne le type de données racine, qui dans le cas du métamodèle `Content` utilisé, est la classe `Event`.
2. `excluded` : permet d'exclure tous les attributs et propriétés qui ne sont pas à générer en JSON. Dans l'exemple ci-dessous cela permet de retirer une référence qui est

utilisée pour contenir des catégories (`categories`) alors que ces mêmes catégories seront déjà générées *via* des références ayant une sémantique plus forte comme `mainStay` ou bien `mainCategory`. Par conséquent dans le JSON généré, `categories` est vide.

3. `isLabel` : duplique un attribut en lui donnant comme nom `label`. Cette annotation est utile notamment pour des composants affichant des données nécessitant un attribut nommé `label` comme Simile Exhibit. L'attribut est dupliqué car l'original peut être utilisé par ailleurs.
4. `valuesInContainer.label` : déplace la valeur de l'attribut `label` dans le conteneur de l'objet concerné. Cela est utile pour limiter le nombre d'objets générés, e.g. : `mainStay`, `mainCategory` et `secondaryCategoriesAndSubjects` (cf. détails en annexe H.4).

```
/** EMF2JSON Directives */

@EMF2json " isItemType "

aspect class Event {
  @EMF2json " excluded "
  attribute categories : Category [0..*]
  @EMF2json " isLabel "
  attribute title : String [1..1]
  @EMF2json " valuesInContainer.label "
  reference mainStay : Category [0..1]
  @EMF2json " valuesInContainer.label "
  reference mainCategory : Category [0..1]
  @EMF2json " valuesInContainer.label "
  reference secondaryCategoriesAndSubjects : Category [0..*]
}
```

FIGURE 6.5.10: Annotations en KERMETA pour personnaliser la génération du JSON avec EMF2JSON

6.5.7 Génération de l'application web

Le concepteur d'un application choisit et configure les composants graphiques avec les différents modèles fournis par MODSEA : une instance du métamodèle `Widgets` et des instances des autres métamodèles `Layout`, `Widget_Content` et `Widget_Layout`. L'annexe H.3 donne un exemple de configuration pour une application réalisée dans le cadre de RMM2.

Le code HTML de l'application web est généré grâce à l'application de patrons de code source écrit en KET. Cette application est réalisée par un programme KERMETA.

Ces patrons prennent en entrée les différentes instances des métamodèles énoncées ci-dessus. Ils sont spécifiques à chaque composant graphique de Simile Exhibit et sont eux-mêmes générés conformément à leur `SWidget` respectif. Ainsi, à partir d'un seul patron générique, il est possible de créer automatiquement l'ensemble des patrons nécessaires à la génération du code HTML des composants graphiques.

Ci-dessous un exemple de patron servant à créer une `TimelineView`. Comme nous venons de le dire, ce patron a lui-même été généré à partir d'un patron KET et du `SWidget` dédié à la `TimelineView`.

```
<div ex:role="view" ex:viewClass="Exhibit.TimelineView" ...
      <% if (not self.start.isVoid()).andThen{ e | not self.start.name.isVoid()} then
%> ex:start="<%= self.getJavaScriptElement(self.start) %>" <% end %>
... >
```

Le résultat de la génération est une application web prête à l'emploi comme le montre la figure 6.5.11. Les utilisateurs peuvent trier et filtrer les événements suivant des catégories à l'aide de facettes (colonne de gauche), afficher une vue chronologique (cadre du haut) et une carte (cadre du bas) indiquant leur positionnement spatio-temporel. Grâce à cette approche modulaire, nous avons généré un ensemble d'applications en réutilisant les modèles et donc proposer à nos partenaires de RMM2 les différentes combinaisons conformes à leurs besoins.

FIGURE 6.5.11: Résultat d'une génération avec MODSEA

6.6 Conclusion

Dans ce chapitre nous avons décrit les différents modes d'acquisition de modèles à l'aide soit d'interfaces graphiques (web) soit de grammaire. Nous montrons ensuite

l'usage de transformations de modèles pour rendre interopérable PTOM avec des standards ou bien des modèles *ad hoc* (e.g. : APM).

La grammaire STNL que nous présentons est un moyen rapide et pratique pour analyser des horaires d'ouverture en langage naturel lorsque les textes sont suffisamment formatés. STNL pourra être étendue pour prendre en compte les jours de la semaine et ainsi pouvoir analyser des expressions du type « *de 10h à 19h du lundi au vendredi* » afin d'augmenter le nombre d'expressions interprétées. Notre approche est utile et efficace pour les « horaires d'accessibilité », restriction des « périodes d'accessibilité », car la sémantique et l'interprétation d'expressions conformes à STNL sont déterministes ce qui est un gage de qualité.

Nous proposons, avec PTOM-S, une preuve de concept pour un système d'interrogation afin de retrouver des événements. L'interrogation est réalisée sur les propriétés temporelles en intension de type « période d'accessibilité » suivant un sous-ensemble des relations ALLEN*. Ainsi, PTOM-S permet de retrouver les événements ayant lieu « *entre 10h30 et 12h tous les jours* » (*during**) ou encore « *tous les jours du lundi au vendredi de 10h à 19h* » (*equals**). À cette occasion nous montrons que l'interrogation des expressions temporelles que nous traitons est plus efficace en intension qu'en extension.

Nous avons également décrit l'approche MODSEA qui vise à fournir un cadre de développement au concepteur d'applications web dédiées à la publication d'événements à caractère spatio-temporel. Un outil fondé sur des modèles est proposé afin de configurer et générer des familles d'applications de visualisation basées sur la bibliothèque Simile Exhibit. MODSEA a permis aux agences de presse de concevoir rapidement des variantes d'applications à la fois du côté Relaxnews et AFP.

Nous nous sommes également intéressés au cycle de vie des composants de l'application en proposant la génération automatique de métamodèles de configuration de composants graphiques lors de la mise à disposition systématique des nouvelles versions de bibliothèques et ce à partir de leur description JavaScript et JSON.

Pour faire évoluer MODSEA, l'approche évoquée en section 6.5.2 consistant à utiliser des *Feature Model* serait intéressante pour modéliser nos configurations (choix des modèles utilisés pour la génération HTML) qui sont pour le moment définies dans un format textuel *ad hoc* (cf. figure H.3.5). Notre approche, qui consiste à extraire des informations d'une bibliothèque, pourrait s'appliquer à d'autres outils tels que Ext JS, dans la mesure où celui-ci décrit également les attributs de ses composants dans des annotations (cf. Figure 6.5.6).

Chapitre 7

Applications et cas d'utilisation

Ce chapitre présente un ensemble de cas d'utilisation pour les métamodèles et les méthodologies que nous avons présentés auparavant. Il s'agit pour nous de montrer la variété des domaines applicatifs ciblés et l'étendue des fonctionnalités potentielles avec leurs différents niveaux de complexité.

7.1 Introduction

Nous aborderons en premier lieu un exemple académique utilisé par Terenziani pour présenter sa proposition d'un langage de contraintes sur les occurrences d'événements périodiques. Parmi les cinq cas exemplaires qu'il présente, nous retenons le quatrième qui met en jeu à la fois les aspects structurels et temporels. Les autres cas sont élémentaires et moins représentatifs de nos préoccupations. Nous donnons notre modélisation avec Ptom qui peut être comparée avec celle exprimée selon les concepts introduits par Terenziani.

À la frontière des domaines universitaire et applicatif, nous nous intéresserons ensuite à deux types d'études.

Les études du premier type sont menées en géographie appliquée, avec des objectifs liés à la sociologie. Cette étude concerne les « Fêtes annuelles de Gand ». L'auteur s'intéresse à repérer et analyser des archétypes comportementaux récurrents chez les visiteurs selon les heures, les jours et les années. Nos modèles permettent de capturer et de spécifier dans des bases de connaissance les résultats des recherches et de les mettre à disposition, prêtes à être exploitées par d'autres chercheurs.

Le deuxième type d'étude concerne, un laboratoire de géographie, qui analyse l'impact des conditions environnementales, de la législation en cours sur l'écosystème en général et plus particulièrement la pêche à pied professionnelle sur la côte du Finistère Sud. Ces études utilisent des systèmes multi-agents qui simulent les différents facteurs en interaction et observent les évolutions à court et moyen terme. Les systèmes multi-agents exploitent des bases de connaissance qu'ils mettent à jour en continu. Ptom offre les moyens de modéliser, stocker et interroger les connaissances temporelles.

Nous continuons par l'exposé des points méthodologiques essentiels du projet applicatif qui a servi de base au financement de la thèse, à savoir le projet ANR RMM2, que

nous avons mené avec nos partenaires industriels (Relaxnews, AFP) et nos collègues universitaires (MoDyCo).

C'est l'occasion d'exposer l'architecture globale de la plate-forme qui met en œuvre nos propositions et d'insister sur les contributions fonctionnelles pratiques de nos travaux du point de vue utilisateur, intermédiaire ou final.

Nous terminons par des expérimentations du système d'interrogation PTOM-S afin de montrer les résultats des requêtes suivant plusieurs données d'entrée.

7.2 Exemple issu des travaux de Terenziani

Comme nous avons déjà pu l'indiquer aux chapitres 2 et 4, TERENZIANI a réalisé de nombreux travaux et études sur la modélisation d'événements répétitifs dont la construction des règles de périodicité fondées sur des éléments calendaires. Dans cette section, nous avons pour objectif de mettre en œuvre nos formalismes, PTOM et ALLEN*, pour les appliquer à un des exemples de TERENZIANI. Nous avons choisi l'exemple numéro 4 de l'article [Terenziani 00]. Les autres exemples se traduisent de façon élémentaire en PTOM.

7.2.1 Présentation de l'exemple

Cet exemple (figure 7.2.1) met en jeu à la fois la récurrence en exprimant une fréquence de répétition et des relations qualitatives entre événements. Selon les formalismes de TERENZIANI, cet exemple exprime une contrainte entre événements « LOC-QUAL » (cf. section 2.3.4.1) avec des récurrences d'événements dites « *pseudo* périodiques ».

> « *Entre le 29/09/1997 et le 19/12/1997, deux fois chaque mardi la classe I_A aura deux cours de Gymnastique précédent (strictement ou non) un cours d'Histoire.* »

FIGURE 7.2.1: Exemple numéro 4 traduit en français de l'article [Terenziani 00]

Dans cet exemple, il est important de distinguer les événements. Ainsi, nous en identifions deux : « *cours de Gymnastique* » et « *cours d'Histoire* ». La relation qualitative « *précédent (strictement ou non)* » indique que les deux « *cours de Gymnastique* » ont lieu avant un « *cours d'Histoire* ». Le cas où le « *cours d'Histoire* » suit immédiatement le second « *cours de Gymnastique* » est admis. Ce patron se répète « *deux fois tous les mardis entre le 29/09/1997 et le 19/12/1997* ».

7.2.2 Mise en œuvre avec PTOM et Allen*

Décomposition structurelle et récurrence

La figure 7.2.2 montre la décomposition en événements de l'exemple précédent réalisée avec PTOM. Ainsi, nous avons d'un côté les cours de Gymnastique et ceux d'Histoire. Nous proposons de regrouper les deux cours de Gymnastique qui se répètent avant chaque cours d'Histoire dans un événement nommé `SéancesDeGym`. Nous associons à `CoursDeGym` une propriété temporelle indiquant qu'il y a deux `CoursDeGym` par `SéancesDeGym`, i.e. : « *2 fois* ». La séquence `SéancesDeGym - CoursHistoire` se répète deux fois tous les mardis, ainsi nous ajoutons à la fois à `SéancesDeGym` et `CoursHistoire` une propriété temporelle indiquant qu'ils se répètent *« 2 fois tous les mardis du 29/09/1997 et le 19/12/1997* ». La relation `structural` entre `SéancesDeGym` et `CoursDeGym` indique que `SéancesDeGym` est composé de `CoursDeGym` qui se répète 2 fois par `SéancesDeGym`.

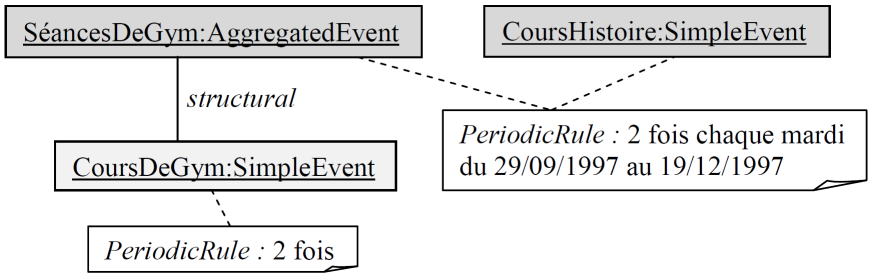


FIGURE 7.2.2: Décomposition en sous-événement et spécification des propriétés temporelles

Relation temporelle

Après avoir créé les sous-événements et spécifié les propriétés temporelles, il est maintenant nécessaire d'ajouter la relation temporelle qualitative liant `SéancesDeGym` et `CoursHistoire`. La figure 7.2.3 donne la traduction en ALLEN* de la relation qualitative utilisée dans l'exemple : « *(strictement ou non) avant* ». Ainsi, nous utilisons *interleaves** pour traduire : « *strictement avant* » et les relations *meets** et *Met_by** pour : « *non strictement avant* ». L'utilisation de l'opération `firstOcc` permet de calculer la première occurrence de l'événement concerné.

```
Topology:
(meets*(CoursHistoire, SéancesDeGym)and
Met_by*(SéancesDeGym, CoursHistoire)
)or(
interleaves*(CoursHistoire, SéancesDeGym) and
interleaves*(SéancesDeGym, CoursHistoire)and
firstOcc(SéancesDeGym) precedes firstOcc(CoursHistoire))
```

FIGURE 7.2.3: Relation qualitative entre événements

7.2.3 Apports de PTOM

Alors que TERENZIANI propose un langage, PTOM offre en plus le paradigme qui permet d'ajouter des contraintes (OCL ou autres), de faciliter son extension.

7.3 Les Fêtes du marché de Gand

Nous considérons ici une étude particulière menée par VERSICHELE [Versichele 12] qui traite de la fréquentation des Fêtes de Gand (Belgique) à des fins de gestion prévisionnelle des ressources : biens de consommation, moyens de contrôle et d'assistance, ou de maîtrise de la pollution. Les Fêtes de Gand constituent un événement qui se déroule annuellement après la mi-juillet depuis un siècle et demi. Cette information est globale. Plus précisément, les Fêtes de Gand ont lieu sur une période de dix jours durant la seconde quinzaine de juillet. Les dates exactes varient d'une année à l'autre. Certains festivals co-localisés sont identifiés, tels que pour l'édition 2012, celui des marionnettistes, le festival international de danse ou le festival du cirque pour la jeunesse, etc. Les programmes quotidiens, constituent une information encore plus fine. Tous ces éléments sont des instances d'événements susceptibles de porter effectivement des informations temporelles plus ou moins complexes.

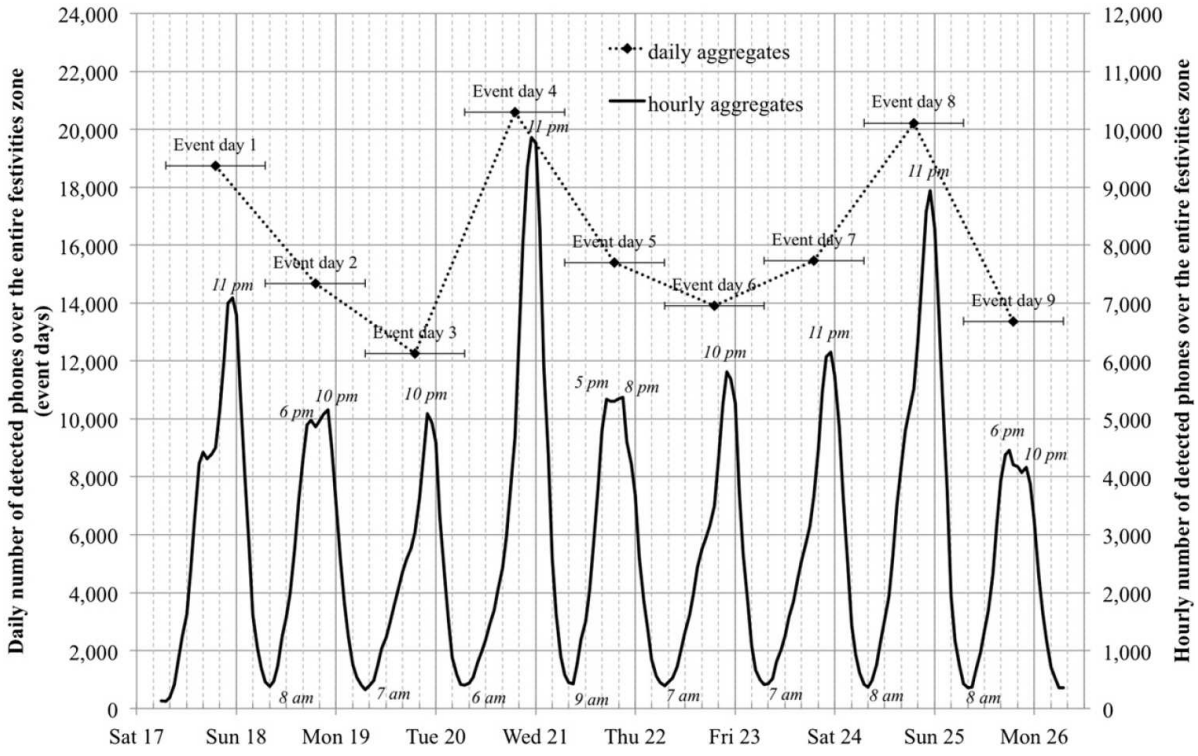
7.3.1 Présentation de l'étude

L'étude menée par M. VERSICHELE [Versichele 12] (cf. figure 7.3.1) analyse les fréquentations absolues et relatives dans différents lieux et à différentes périodes avec des granularités diverses (heure, jour). Ses résultats font apparaître des périodicités et des relations entre occurrences d'événements éventuellement interprétées en termes de causalité.

> « *De manière générale, les pics de fréquentation ont lieu entre 22h et 24h sauf pour les dimanches et le jour de la fête nationale (21 juillet). Ces jours là (dimanches et Fête Nationale), le pic de fréquentation est moins prononcé et s'étale entre 17h et 20h. A contrario, les faibles fréquentations ont lieu systématiquement chaque jour de 2h à 9h.* »

Ainsi, les études menées sur l'ensemble des dates concrètes de présence de chaque visiteur (estimée par un moyen ingénieux impliquant une technologie Bluetooth) apportent un ensemble d'informations de différents types que l'on retrouve dans PTOM :

- structurel : décomposition événements et sous événements selon la catégorie et le lieu des spectacles ;
- temporel : classes contiguës de fréquentation homogène avec leur patron d'occurrences périodiques ;
- relationnel : par exemple, l'interdépendance entre les Fêtes de Gand et la Fête Nationale en matière de fréquentation.



(source [Versichele 12])

FIGURE 7.3.1: Affluence des Fêtes de Gand

Pour rendre ces informations accessibles de façon systématique en les intégrant dans des bases de connaissance, il convient de les modéliser sous les trois aspects précédents.

Plus précisément, les bases de connaissance ainsi constituées aident l'utilisateur (ici le géographe) à rechercher des événements aux propriétés voisines de celles mises en évidence afin de conforter ses hypothèses. En retour, les événements sur lesquels le géographe travaille et qu'il vient de contribuer à catégoriser, viendront peupler la base de connaissance et participer ainsi à la suite du processus.

Il est donc primordial de disposer d'un modèle et de langages, afin de décrire des propriétés périodiques et des catégories d'événements pour formuler des requêtes, interroger une base de connaissance et procéder à enregistrer de nouveaux résultats. Clairement, une suite de dates concrètes (la suite en extension des dates calendaires d'occurrence des événements) masque totalement la sémantique exprimée par des descriptions en intention avec des assertions du type : « *Lors des Fêtes de Gand, les pics de fréquentation ont lieu chaque jour à 23h sauf les dimanches et pendant la Fête Nationale* ».

7.3.2 Utilisation potentielle des relations « temporal » et « structural » de PTOM

La relation « temporal »

On distingue deux types d'associations : entre l'événement générique « *LesFêtesDeGand* » et l'ensemble des éditions annuelles de ces fêtes existe un lien temporel (`temporal`). Lorsque l'on affirme que « *LesFêtesDeGand* » ont lieu annuellement, on indique que l'on va trouver des occurrences : « *FêtesDeGand_1985* », « *FêtesDeGand_2006* »,..., « *FêtesDeGand_2012* » dont les propriétés temporelles seront en accord avec la contrainte énoncée au niveau supérieur sur l'événement « *LesFêtesDeGand* » (une occurrence unique chaque année en juillet, etc).

La relation « structural »

Lorsqu'on reconnaît plusieurs types de festivals (Marionnettes, Danse, Cirque), on indique un lien structurel (`structural`) avec l'instance *mère*. Si classe *mère* et classes *filles* sont porteuses d'informations temporelles, il doit y avoir cohérence : les occurrences des *filles* devant avoir lieu durant la période d'occurrence de la *mère*.

Décomposition horaire

Pour les besoins de l'étude de fréquentation (cf. figure 7.3.1), chaque journée a été décomposée en périodes (tranches) d'une heure, chaque période constituant de fait un événement. Il y a donc une succession de liens de type `temporal` entre un sous-festival donné dont les occurrences ont lieu chaque jour durant la période spécifiée (par exemple : « *du 14 au 23 juillet 2012 pour les marionnettes* »). Chaque jour se décompose lui-même (`temporal`) en 24 événements ayant respectivement lieu périodiquement toutes les heures. À chaque niveau de la décomposition hiérarchique correspond un événement ayant ses

propres propriétés temporelles. La granularité de ces informations gagne en finesse et perd en synthèse selon que l'on parcourt la hiérarchie dans un sens ou dans l'autre.

7.3.3 Apports de PTOM

Représenter de façon numérique et directe cette sémantique est précisément ce que nos modèles (i.e. PTOM) se donnent comme objectifs. Ces cas d'utilisation traitent d'événements complexes, structurés de différentes manières et qui ont en général une durée non nulle. Ces caractéristiques sont plus larges que celles régissant les événements dans les systèmes à états. La complexité réside en particulier dans le fait que les événements peuvent être considérés à plusieurs niveaux. Un événement peut lui-même se décomposer structurellement en sous-événements de différents types. Selon un point de vue temporel, on peut reconnaître des événements définis à différents degrés de finesse.

7.4 Bénéfices d'un modèle temporel en intension pour la simulation d'activités humaines

Ce cas d'utilisation est né d'une collaboration avec le laboratoire GÉOMER (UMR 6554 LETG). L'objectif est de coupler le métamodèle temporel PTOM avec un Système Multi-Agent (SMA) simulant des activités humaines et notamment celles liées à la pêche à pied dans les zones maritimes.

Pour les géomaticiens de Géomer, un des intérêts d'utiliser le modèle que nous proposons est d'éviter, comme pour RelaxMultiMedias 2, de stocker l'ensemble des périodes d'autorisation dans une base de données et ce sur des emprises temporelles de plusieurs années. PTOM est utilisé ici comme un modèle pivot permettant la conception d'applications interopérables dans les domaines de raisonnement analytique.

La suite de cette section présente tout d'abord en 7.4.1 les motivations et l'exemple d'activité humaine qui sera mis en œuvre tout au long de ce cas d'application. Puis nous décrivons les contraintes temporelles pour cette activité en 7.4.2. Après avoir présenté le SMA sous-jacent en 7.4.3, nous décrivons notre approche pour le coupler avec PTOM en 7.4.4. Puis en 7.4.5 nous commentons les apports de PTOM afin de modéliser des contraintes temporelles pour des activités humaines régies par une règlementation.

7.4.1 Motivations

Dans un processus de prise de décision mettant en jeu l'espace et le temps, les volumes de données à traiter sont souvent importants. Manipuler des données temporelles sous forme de dates ancrées dans le calendrier (dates concrètes) a pour effet de masquer une grande partie de leur sémantique qui pourrait être mise à profit pour améliorer le moteur d'analyse d'un tel processus.

Le développement de modèles dynamiques dans le domaine des activités humaines ou bien des interactions nature-société [Le Tixerant 10] nécessite une analyse multi-échelles des interactions en termes spatiaux et temporels [Peuquet 94]. Le point de vue précédent s'applique dans le cas d'utilisateurs ou composants logiciels qui requièrent des services dont les réponses dépendent du contexte de leurs sollicitations, et plus particulièrement dans notre cas, de la localisation et de l'heure. Ces services dialoguent avec un modèle d'activité humaine afin d'intégrer les paramètres spatiaux et temporels de la sollicitation, avant de retourner une réponse cohérente à l'utilisateur ou au composant logiciel.

Le domaine d'application traité se réfère à une plate-forme de simulation utilisant des SMA géolocalisés [Weiss 00, Shoham 09]. L'exemple d'activité humaine que nous avons choisi, qui sera décrit ci-après, traite de la pêche à pied professionnelle.

Notre approche reste proche du langage naturel pour améliorer les interactions avec les utilisateurs finaux tout en étant non ambigüe grâce à la grammaire formelle de PTOM.

La pêche à pied professionnelle à la telline (*Donax Trunculus*)

Notre cas d'utilisation concerne l'activité de pêche à pied professionnelle à la telline (*Donax Trunculus*) dans la Baie de Douarnenez [1] (France). A Douarnenez, la pêche à la telline est apparue dans les années 70 et s'est avérée très rentable, ce qui a conduit à l'épuisement rapide du stock de telline [Guillou 82]. Afin de réguler l'activité et qu'elle puisse perdurer, l'administration a rédigé des règles statuant sur l'accès aux zones de pêche notamment en limitant leur durée et en imposant de déclarer les volumes de récoltes. Dans ce cas d'utilisation, nous souhaitons montrer l'utilisation de PTOM et sa syntaxe textuelle pour modéliser la partie de la règlementation relative au calendrier de pêche. Plus précisément, PTOM sera couplé avec un SMA afin de simuler le processus cette activité.

7.4.2 Modélisation temporelle d'activités humaines

7.4.2.1 Principes généraux

La modélisation temporelle consiste à construire un Calendrier de Pratique Potentielle (CPP) qui résulte de la compilation de diverses contraintes (cf. figure 7.4.1) :

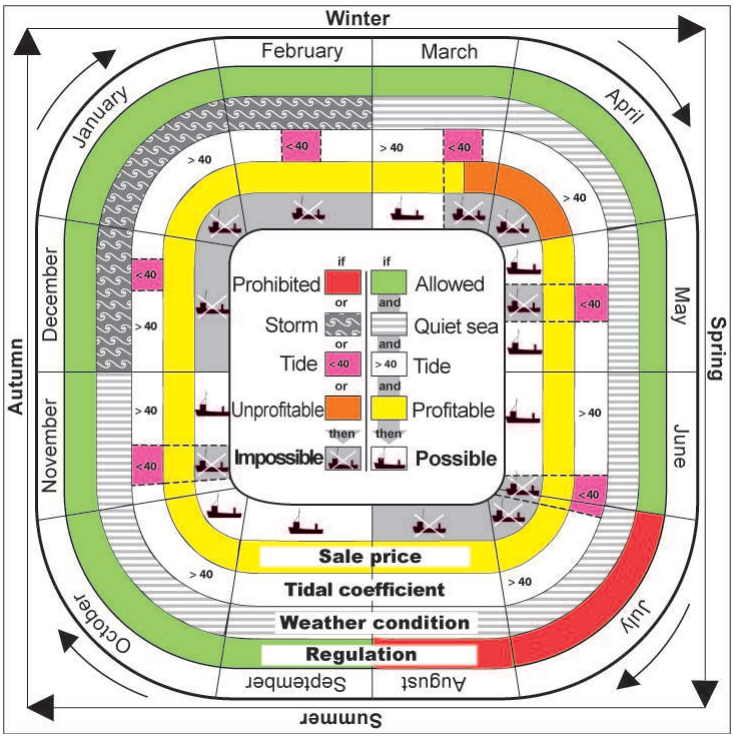
1. pour chaque zone de pêche règlementée, la loi précise si la pêche est « autorisée », « restreinte » ou « interdite ».
2. l'état de la mer et la marée, ainsi que la température, contraintes d'accessibilité aux zones de pêche (rochers, etc).
3. La qualité bactériologique de l'eau a également une incidence sur les droits de pêche.

1. `http://www.geonames.org/maps/google_48.167_-4.417.html`

Toutes ces contraintes sont liées à plusieurs facteurs périodiques. En ce qui concerne la zone règlementée de pêche à pied de Douarnenez-Camaret, en plus d'autres contraintes (conditions climatiques, coefficient de marée ou bien le prix de vente), il est stipulé que :

> « *La pêche à pied est interdite la nuit de 21 heures à 6 heures du matin entre le 1er juillet et le 31 août et tous les samedis de l'année. En dehors de ces périodes, la pêche est autorisée à partir 3h avant la basse mer jusqu'à 3 heures après la même basse mer (selon l'almanach des marées de Douarnenez). Sinon, elle est limitée.* »

Cette règle est représentée sur le CPP de la figure 7.4.1, par la première contrainte (ceinture extérieure) qui est nommée « *Regulation* ».



(source [Faucher 10e])

FIGURE 7.4.1: Calendrier de Pratique Potentielle de la pêche à pied de Donax trunculus

La figure 7.4.1 utilise une représentation dite de roue temporelle [Peuquet 02, Moellering 76, Edsall 05] pour représenter la périodicité (cycle) des propriétés tels que la règlementation et les coefficients de marée. La période d'une marée est d'environ 12h25 (demi-cycle lunaire, durée entre deux basses mer) alors que pour d'autres contraintes le cycle est annuel. La roue temporelle correspond à une zone géographique homogène pour les autres paramètres. Elle représente un ensemble de conditions moyennes calculées sur plusieurs années consécutives et fournit ainsi un modèle moyen pour les paramètres qui ont un

impact sur la pêche à pied à la telline. Des conditions locales ajoutent éventuellement quelques corrections à l'effet moyen.

La spécification et la définition du comportement des agents nécessite que la connaissance temporelle et les bases de données soient interrogées lors de l'exécution afin de mettre à jour l'environnement modélisé, l'état des agents et leurs interactions. Il s'agit d'une question complexe, étant donné que toutes les règles temporelles élémentaires doivent être codées ainsi que leurs exceptions, y compris les positions relatives entre événements (météo, océanographie, décisions administratives, etc.).

La plus-value apportée par PTOM est de permettre des traitements à partir de la sémantique temporelle des événements plutôt que de traiter directement des données temporelles (dates) d'occurrences.

7.4.2.2 Modélisation avec PTOM

Le cas particulier de la marée

Les marées constituent un phénomène naturel, qui génère de nombreuses dates d'occurrences géolocalisées [Andrienko 03, Roth 09] notamment du fait que les marées n'ont pas lieu au même moment y compris pour des localités faisant parties d'une même région. Les marées sont *pseudo* périodiques, en effet, le phénomène se répète environ toutes les 12h25 mais ceci n'est qu'une moyenne. Des formules de calcul existent qui mettent en jeu des phénomènes physiques, astronomiques, ce qui rend difficile sa définition complète avec PTOM.

Pour des questions de fiabilité de résultats de simulation, il ne semble pas envisageable que le simulateur utilise un modèle temporel des marées approximatif (réalisable avec PTOM), ni de mettre en œuvre un modèle aussi complexe voire plus complexe que celui traité pour calculer uniquement un des paramètres d'entrée d'une simulation. Les données de marée sont utilisées *a posteriori* par la simulation, par conséquent les dates et heures réelles peuvent être connues car les phénomènes sont passés. Ainsi dans ce type de cas, il semble raisonnable de stocker en base de données tous les horaires des marées nécessaires pour le simulateur et de partager ces données avec d'autres activités humaines qui pourraient être simulées au sein de la même plate-forme.

Autrement dit, les horaires des marées sont stockés dans la base de données comme des dates concrètes. En revanche, le calcul donnant le statut d'autorisation est lui réalisé en intension. La sémantique est ainsi conservée.

Pour récupérer l'horaire de marée, il est donc nécessaire de spécifier une emprise temporelle dans la requête sur la base de données.

Décomposition en événements

Nous décomposons la partie de la règlementation relative au temps pour la spécifier en PTOM. Ainsi, nous créons trois événements : le premier concerne la « *Nuit* », le deuxième la « *Marée Basse* » et le troisième « *l'autorisation de la pêche* ».

1. Comme la règlementation le définit, la « *Nuit* » est un événement qui a lieu « *tous les jours de 21h à 6h du jour suivant* », le *jump* est à utiliser.
2. La « *Marée Basse* » va chercher, comme il a été dit précédemment, dans une base de données les horaires de marée pour une étendue temporelle qui est restreinte pour chaque jour demandé au complément de la « *Nuit* ».
3. « *L'autorisation de pêche* » est restreinte entre 3h avant et 3h après la « *Marée Basse* », nous utilisons ici les relations d'Allen entre instants $Before$ et $After$ avec un écart de « 3h », qui permettent de traduire la fenêtre d'autorisation centrée sur la basse mer. La pêche est interdite la nuit, par conséquent nous ajoutons une exception sur l'événement « *Nuit* ».

```
//Donax trunculus (telline)
//night definition following the regulation
The event "night" occurs periodically
  according to the rule(s) below
  - rule: (identified by night_)
    from each 21th hour of each day
    to each 06th hour of each [n+1] day
end of the event

//low tide in Douarnenez
The event "low tide" occurs periodically
  according to the rule(s) below
  - rule: (identified by low_tide_Douarnenez)
    1 times during one 12hours 25minutes period
    //The first occurrence of low tide
  and starts on "2010-07-15T14:23:00"
end of the event

//seashell of Donax trunculus digging
The event "Telline seashell digging" occurs periodically
  according to the rule(s) below
  //3 hours before the begin of the "low tide"
  //3 hours after the end of the "low tide"
  //low tide is defined as a periodical instant
  //then the begin and end of the "low tide" are the same instant
  - rule:
    from with a gap of 3 hours before low_tide_Douarnenez
    to with a gap of 3 hours after low_tide_Douarnenez
  //Exceptions, except the night and each Saturday
  except from equals night_ to equals night_ and except each Saturday
  and except from each July to each August

end of the event
```

Figure 7.4.2: Syntaxe textuelle Ptom de la règlementation de la pêche à pied professionnelle à la telline

7.4.3 Simuler une activité humaine sous contraintes spatio-temporelles avec la plate-forme DAHu

La simulation d'activité comme celle de la pêche à pied à la telline est réalisée à l'aide de la plate-forme DAHu (Dynamique des Activités Humaines, [Tillier 10, Tissot 04]).

Ce SMA est en charge de simuler les activités anthropiques et leur interaction avec l'environnement. Une interface graphique spécifique permet de sélectionner une espèce, une emprise spatiale et temporelle pour une simulation en particulier. La figure 7.4.3 montre un résultat de simulation sous forme de carte interactive consultable à différentes échelles.

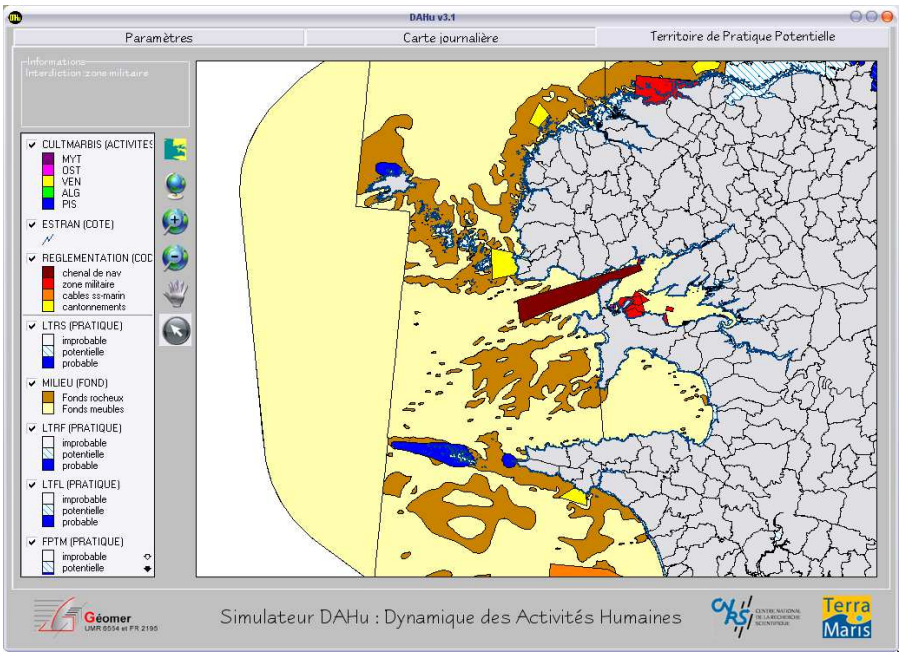


FIGURE 7.4.3: Sortie de simulation avec DAHu

7.4.4 Modéliser et interroger des informations temporelles lors de la simulation d'activités humaines

Cette section décrit la mise en œuvre d'un service basé sur des informations spatio-temporelles (dans la suite LTBS, Located Time Based Service) pour la plate-forme DAHu, qui est utilisée pour simuler l'activité de la pêche à pied à la telline.

Nous avons conçu un LTBS comme un composant externe et indépendant qui peut être partagée par plusieurs applications distantes, à condition qu'ils se réfèrent au même modèle temporel, i.e. PTOM. Les clients peuvent appeler le LTBS avec des paramètres, les résultats peuvent être de différentes natures. Les principaux cas d'utilisation du LTBS sont les suivants :

- modéliser les règles temporelles pour des interactions nature-société pour un endroit donné.
- répondre aux requêtes sur le modèle temporel et vérifier que des instants ou périodes sont compatibles avec les règles temporelles modélisées précédemment.
- fournir un Calendrier de Pratique Potentielle (de manière optionnelle, calculer les

dates ancrées dans le calendrier qui spécifie quand la pratique est autorisée ou interdite).

Dans la plate-forme SMA DAHu, les agents logiciels sont supposés être liés à des lieux particuliers. Leur état évolue tout au long du processus de simulation, et ils peuvent à tout moment récupérer des éléments d'information en fonction du temps et du lieu. Reprenons l'exemple de la pêche à pied et considérons un agent qui représente une zone de pêche règlementée. Il est lié à un objet dans le modèle temporel qui intègre son comportement dynamique en conformité avec l'ensemble des règles administratives. Lorsque le SMA interroge le statut de la zone de pêche au cours d'une période donnée (cf. figure 7.4.4), il envoie une requête (avec cette période comme paramètre d'entrée) au LTBS, qui renvoie « autorisé », « interdit » ou « limité ».

Lors de sa création, la plate-forme DAHu utilisait des propriétés temporelles exprimées en extension. Ainsi, lors d'une simulation, DAHu interagissait avec la base de données contenant tous les instants concrets indiquant que l'activité correspondante avait eu lieu. Désormais, la base de données enregistre uniquement les expressions périodiques de manière intensionnelle. DAHu interroge cette base de données *via* le LTBS.

Un autre cas d'utilisation serait d'identifier tous les Territoires de Pratique Potentielle et de les afficher sous forme cartographique, ceci en fournissant un intervalle de temps pendant lequel la pratique est autorisée, par exemple, « *à partir de 11h18 à 17h18* », en supposant que la marée est basse à 14h18 dans la zone correspondante. Ces sorties sont susceptibles d'être soit exploitées par d'autres calculs de simulation, ou visualisées sur une carte sur un ordinateur ou bien sur un smartphone afin qu'un utilisateur final les analyse notamment dans le cadre d'un processus de décision.

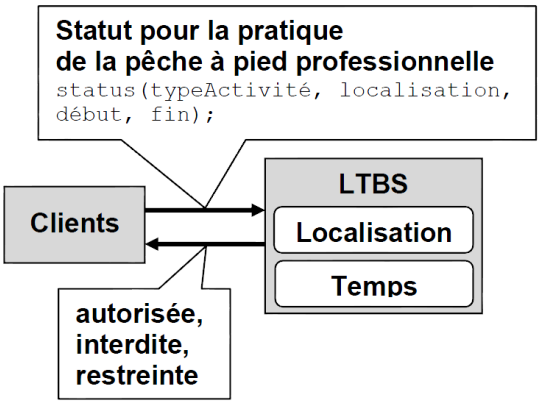


FIGURE 7.4.4: Architecture d'un LTBS

7.4.5 Apports de PTOM

Les différentes règles qui régissent les activités humaines se révèlent complexes et ambigües. De plus, une série de dates ancrées dans le calendrier ne peut pas exprimer une connaissance sous-jacente qui permettrait d'établir avec certitude la périodicité

d'un phénomène ou bien établir des dépendances entre phénomènes. Nous avons pris ici l'exemple d'une autorisation d'activité qui dépend des horaires de marée et de la nuit. Ainsi l'utilisation d'un modèle conceptuel comme PTOM s'est révélé très utile pour capturer et enregistrer la richesse sémantique de la règlementation. Nous avons montré le bénéfice de l'utilisation d'expressions temporelles définies en intension pour à la fois modéliser la périodicité et les dépendances entre événements. Nous avons eu recours à la notion de `RelativePosition` présente dans PTOM. Grâce à cette approche, les données sont plus concises, ce qui est plus efficace en termes de persistance, d'interrogation et de calcul pour des composants logiciels comme un SMA.

Dans cet exemple de la pêche à pied de coquillage, tous les composants de l'infrastructure interagissent *via* les modèles conformes à PTOM qui sont pivots, i.e. : le SMA DAHu, le moteur de requête, le LTBS, le générateur de langage naturel et l'interface graphique pour la visualisation de la carte. La sortie du processus de simulation est un ensemble de cartes qui peuvent être utilisées par des experts afin de prédire la dynamique de l'interaction nature-société dans le cadre d'un écosystème donné qui possède une règlementation.

Dans le futur, un des objectifs serait de fournir à un utilisateur nomade des informations sur la règlementation liée à une espèce suivant un lieu et une étendue temporelle [Faucher 10e] et d'y intégrer des Calendriers de Pratique Potentielle.

7.5 RelaxMultiMedias 2 : Plate-forme de gestion de ressources journalistiques

Comme nous l'avons déjà indiqué, cette thèse s'est déroulée en grande partie dans le contexte du projet RelaxMultiMedias 2. Dans cette section, nous allons décrire les cas d'utilisations des travaux liés à ce projet et de la plate-forme de production journalistique qui en a résulté [2].

7.5.1 Objectifs et processus général

Pour rappel, RMM2 a pour objectif de fournir aux journalistes une aide à la création d'événements pour peupler une base de données qui sert à la fois à produire du contenu pour les médias et à constituer des agendas de couverture d'événements pour les journalistes. De manière générale, nous sommes intéressés à la partie du processus qui consiste à acquérir et modéliser les événements puis à les exploiter par un système d'interrogation, éventuellement couplé à une application de visualisation.

Le processus mis en œuvre dans RMM2 est résumé par la figure 7.5.1 et enchaîne trois étapes qui visent à acquérir et modéliser puis à vérifier et valider, et enfin à exploiter des événements et données temporelles provenant de textes en langage naturel.

2. `http://www.afprelaxnews.com`

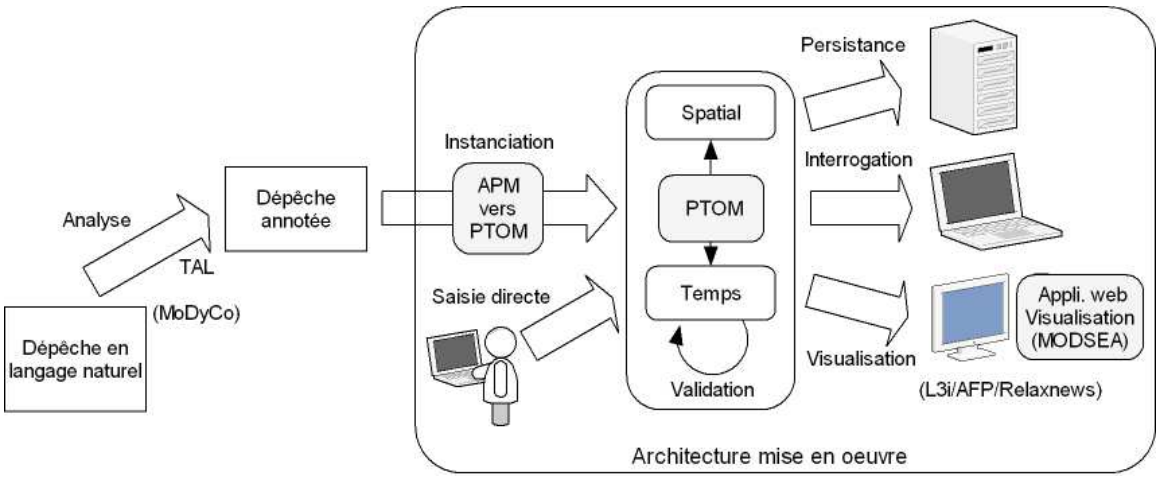


FIGURE 7.5.1: Architecture générale

L'objectif principal nous concernant est de traiter des textes en langage naturel pourvus d'annotations pour les transformer en textes exprimés en langage contrôlé. Nous modélisons les événements à l'aide du métamodèle PTOM, son instanciation est réalisée *via* des transformations de modèles qui permettent ainsi de convertir des données provenant du TAL [Faucher 10c] ou de sources externes comme des flux RSS. Cette conversion comble l'écart sémantique entre le texte initial des dépêches annotées et les informations garanties cohérentes, enrichies de métadonnées et de propriétés temporelles interprétables par une machine. L'outil d'annotation a été développé avec les techniques TAL du laboratoire MoDyCo [Battistelli 08, Teissèdre 10, Battistelli 11].

Après la phase annotation, il est nécessaire de proposer à un utilisateur final un moyen de visualiser sous forme textuelle, de diagnostiquer et de corriger les expressions interprétées. En effet, des expressions correctes pour le système d'annotation peuvent être incomplètes pour le métamodèle temporel, i.e. seule la date de début de l'événement est spécifiée sans la fin. Il peut aussi subsister des incohérences d'unités comme, par exemple, l'utilisation de la même unité à plusieurs reprises : « *chaque 3e jour de chaque mardi de chaque semaine* ». Ici, « *3e jour* » et « *mardi* » ont la même résolution temporelle. La vérification consiste à vérifier que les données acquises et modélisées soient structurellement et sémantiquement correctes. Pour cela nous mettons en œuvre les différentes approches (contraintes OCL et modèle de calendrier, pour la vérification sémantique) présentées dans les chapitres 3 et 5.

Pour finir, nous exploitons les événements et leurs propriétés temporelles grâce à un système d'interrogation et une application de visualisation d'événements est proposée pour afficher de manière conviviale à la fois le caractère temporel et spatial des événements sur différents supports numériques (cf. chapitre 6).

7.5.2 Illustration du processus général avec l'exemple de l'événement « Festival des Francofolies »

Pour illustrer le processus général de traitement, prenons le cas d'un texte (déjà décrit en section 1.1) publié sous forme de dépêche dans un journal régional au sujet du festival musical des Francofolies (cf. figure 7.5.2). Ce texte constitue l'information brute entrant dans le système. Le premier traitement consiste à reconnaître des entités nommées, i.e. : les éléments d'information pertinents pour la création d'événements.

La Rochelle : les Francofolies dévoilent leurs têtes d'affiche 2011.
« Sud Ouest » révèle le programme en exclusivité des soirées Saint-Jean-d'Acre du 12 au 16 juillet.
Une offre de billets à prix serrés seront en vente dès le mardi 1er février.
Programme en exclusivité :
Mardi 12 juillet : Christophe Maé, Zaz, Ours
Jeudi 14 juillet : Lilly Wood and the Prick, Cocoon, The Do

Légende : What, Where, Who, When.

FIGURE 7.5.2: Texte en langage naturel annonçant les Francofolies de La Rochelle 2011

Le système d'extraction de connaissance produit automatiquement, à partir du texte initial, un texte annoté, ce qui rend ces informations exploitables pour la création assistée d'événements. Comme il a été mentionné précédemment, cette première analyse doit impérativement être validée et complétée par un expert humain (journaliste), de façon à identifier les liens entre les différents éléments (lieux, acteurs, dates), pour typer précisément les informations et lever les ambiguïtés. Les annotations pour la dépêche prise en exemple sont données dans le tableau 7.1.

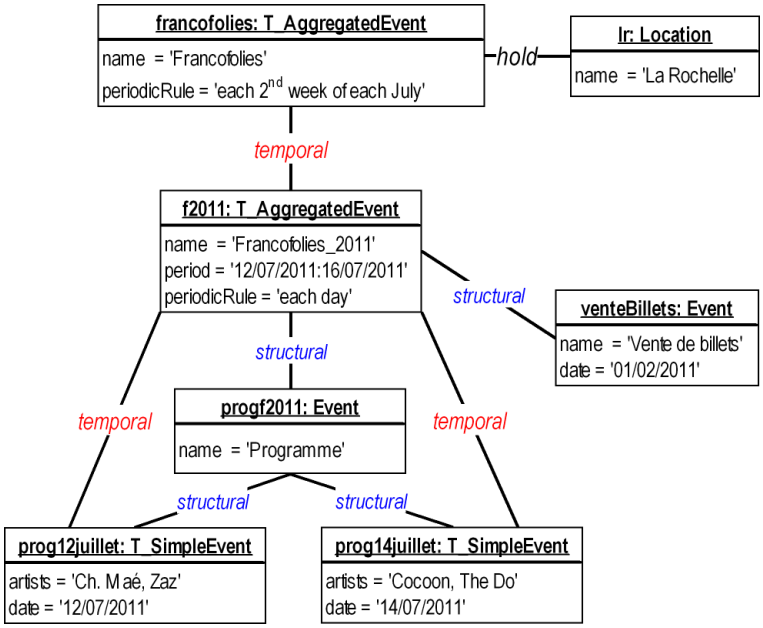
`Where:city = La Rochelle`	`What = vente de billets`
`What = les Francofolies`	`When:date-day = mardi 1er février`
`When:date-year = 2011`	`What = Programme`
`Where:scene = Saint-Jean-d'Acre`	`When:date-hour = Mardi 12 juillet 18h`
`When:duration-start-day = 12 juillet`	`Who:singer = Ch. Maé, Zaz`
`When:duration-end-day = 16 juillet`	`When :date:hour = Jeudi 14 juillet 19h`
	`Who:songGroup = Cocoon, The Do`

TABLEAU 7.1: Résultats de l'annotation de la dépêche annonçant les Francofolies 2011

Les annotations possèdent un type, ex. `What = les Francofolies` et parfois un attribut, ex. `Where:city = La Rochelle`. Ces informations sont ensuite complétées pour notamment mettre en relation les *What*. Par exemple, `Programme` est défini comme un

élément constitutif de « *les Francofolies* ». Lorsque la phase d'annotation est finalisée par le journaliste, une transformation de modèles est appliquée afin de peupler le modèle d'événements et de générer des instances [Faucher 10c].

La figure 7.5.3 est un exemple de résultat de cette transformation et constitue un modèle instanciant le métamodèle d'événements présenté en section 3.4.1. L'élément typé *Where* devient `Location`, les *What* deviennent des `Event`, `T_SimpleEvent` ou `T_AggregatedEvent` suivant leurs fonction et position dans la hiérarchie des événements précédemment décidée par le journaliste. Enfin les *When* et *Who* sont transformés en attributs. Dans la dépêche traitée comme exemple, certains concerts ne sont pas pris en compte car les artistes n'ont pas encore notifié leur présence. Les événements seront complétés au fur et à mesure, soit *via* une nouvelle dépêche annonçant les compléments de programmation, soit directement par les journalistes.



(Pour simplifier la figure, l'attribut `artists` a été ajouté aux instances `T_SimpleEvent`, alors qu'en toute rigueur, cette information relève des métadonnées de l'événement considéré)

FIGURE 7.5.3: Diagramme d'instances du métamodèle d'événement pour l'exemple des Francofolies

En ce qui concerne la visualisation des événements tels que « *Francofolies 2011* », nous utilisons notre approche générative nommée MODSEA (cf. section 6.5) produisant une application web pour trier et filtrer les événements suivant des catégories à l'aide de facettes, d'une chronologique et d'une carte indiquant leur positionnement spatio-temporel [Faucher 10a].

Ajoutons que notre approche MODSEA a été utilisée par l'AFP et Relaxnews pour construire une application de visualisation à partir d'un flux événementiel (figure 7.5.4).

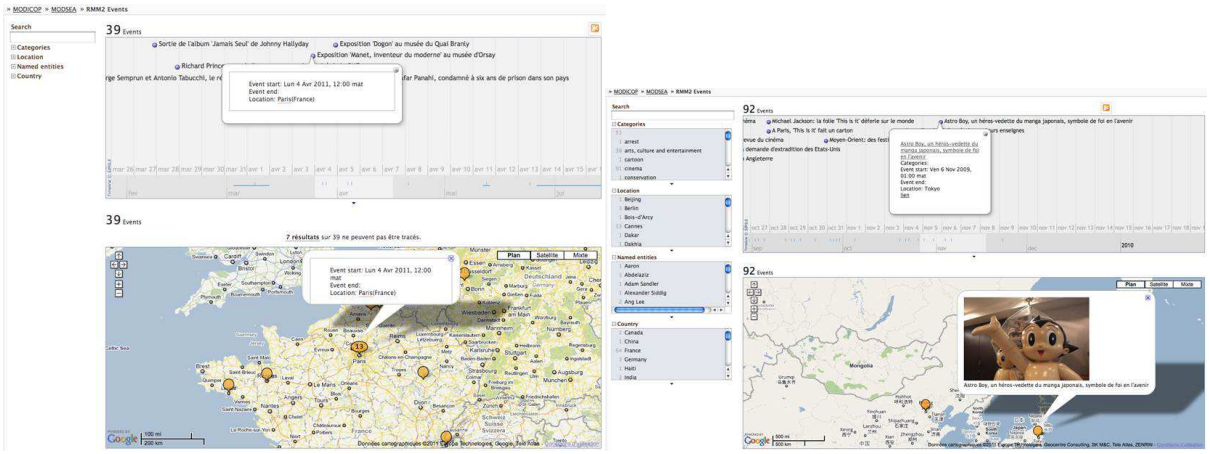


FIGURE 7.5.4: Exemples de réalisation de l'AFP à l'aide de MODSEA

7.5.3 Métamodèles de référence

La mise en œuvre, dans une démarche générique, des objectifs décrits précédemment et l'automatisation des processus supposent la spécification et l'usage d'un nombre conséquent de métamodèles interdépendants. La figure 7.5.5 détaille cette architecture.

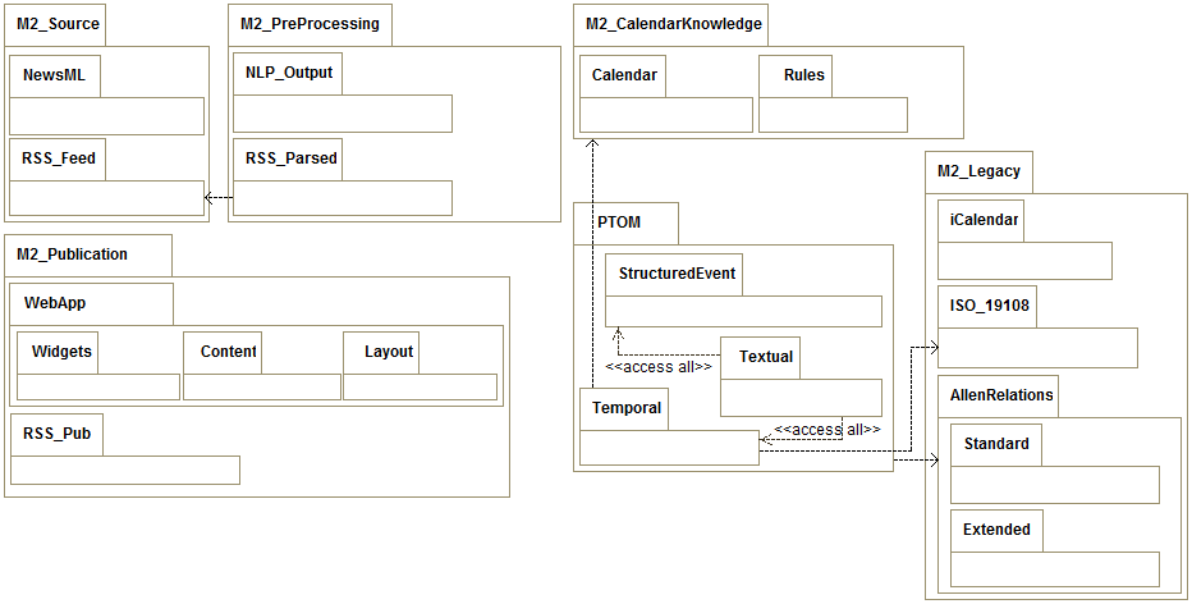


FIGURE 7.5.5: Les différents paquetages et métamodèles de la plate-forme RMM2

Chaque paquetage correspond globalement à une activité du processus métier décrit sur la figure 7.5.1 : `M2_Source` et `M2_PreProcessing` pour l'extraction et l'annotation,

PTOM pour les métamodèles pivots gérant les événements et leurs propriétés temporelles dont la sémantique est issue de `M2_CalendarKnowledge` et de `M2_Legacy`, enfin `M2_Publication` dédié à l'export des informations et à l'adaptation aux systèmes cibles de visualisation.

PTOM est le métamodèle pivot vis-à-vis des autres métamodèles. Il exprime la structure des événements créés et contient les éléments suivants :

- `StructuredEvent` décrit la structure composite des événements et leurs relations mutuelles (cf. section 3.4.1).
- `Temporal` décrit les dates d'occurrence des événements et, dans le cas d'un événement répétitif, les règles régissant cette (*pseudo*) périodicité (cf. section 3.4.2). Une formulation équivalente de ces règles est offerte *via* une syntaxe concrète (textuelle, paquetage `Textual`).

La partie `Temporal` de PTOM s'appuie sur le métamodèle `M2_Legacy` qui contient la description de la spécification ISO 19108 (schéma temporel pour l'information géographique). Celle-ci inclut les relations d'ALLEN [Allen 83] et l'extension ALLEN* (cf. chapitre 4), que nous proposons, qui s'applique aux intervalles non convexes et permet de traiter les événements répétitifs. `Temporal` s'appuie sur le métamodèle `Calendar` qui décrit la connaissance calendaire, i.e. : les entités présentes dans un calendrier (jour, mois,...) et leurs règles d'occurrence. Cette représentation du calendrier permet au processus de validation sémantique des événements de détecter des incohérences telles que : « *le 2e lundi de chaque semaine* » (cf. chapitre 5 et [Faucher 11]). En amont, le métamodèle `M2_PreProcessing` représente l'information extraite des sources d'information, notamment l'information annotée des dépêches après traitement automatique de la langue (`NLP_Output`) et celle extraite des flux RSS (`RSS_Parsed`). Cette information est ensuite exploitée pour instancier le métamodèle PTOM.

La dernière étape, dans la chaîne de création des événements, utilise le métamodèle `M2_Publication` décrivant un modèle général d'interface graphique en séparant les sources de données à visualiser (`Content`), les composants graphiques (`Widgets`) utilisés et la disposition de ces composants (`Layout`). Pour la visualisation des événements, nous avons utilisé la bibliothèque de composants Simile Exhibit.

Cette modélisation des entités intervenant dans la chaîne de production a été réalisée dans un souci d'intégration de nouvelles sources de données et pour faciliter l'exportation vers de nouvelles plates-formes de diffusion. L'IDM nous a permis d'exploiter les métamodèles à la fois pour la production de données et pour générer des interfaces graphiques dédiées à la visualisation des événements stockés en base.

7.5.4 Du texte en langage naturel au texte contrôlé

L'expérimentation qui suit avait pour objectif de tester avec des exemples réels à la fois l'expressivité du métamodèle temporel et le processus de validation des modèles. Cette expérimentation exploite le corpus déjà décrit dans le chapitre 6. Pour rappel celui-ci a été établi par l'agence de presse Relaxnews afin de couvrir leurs besoins. Il est composé

d'un nombre conséquent d'expressions temporelles (513) qui ont été extraites de dépêches d'information. Ces expressions définissent des périodes d'accessibilité, par exemple « *le musée est ouvert tous les jours de 10h à 19h sauf le lundi* ». Pour mettre en œuvre cette expérimentation, un processus complétant le processus général vu précédemment a été défini, il se compose de quatre étapes.

Processus

1. La première étape consiste à annoter le texte en langage naturel qui a été extrait des dépêches. Cette phase d'annotation est réalisée par les algorithmes TAL développés par Charles Teissèdre dans le cadre de sa thèse (2009-2012). Le résultat est un fichier XMI, qui est une instance du métamodèle de période d'accessibilité discuté en section 6.2.2. À noter que le système d'annotation TAL considère que les textes sont uniquement exprimés en intension, ainsi les sorties de ce système sont toutes sous forme intensionnelle
2. Puis à l'aide de la transformation de modèles décrite en section 6.2.2.2, nous instancions la partie temporelle du métamodèle Ptom. Le résultat est un ensemble d'événements munis de leurs propriétés temporelles. Pendant cette phase, les instances de Ptom sont vérifiées à l'aide de contraintes. Un diagnostic est produit pour informer l'utilisateur des éventuelles erreurs sur les expressions temporelles. Si l'utilisateur le souhaite, l'évaluation des contraintes peut être ré-exécutée pour générer un nouveau diagnostic.
3. Un texte en langage contrôlé (conforme à la grammaire de Ptom) est généré afin de proposer à l'utilisateur une vue textuelle proche du langage naturel
4. Enfin, l'utilisateur peut valider si le texte fourni en tant que source (langage naturel) a été interprété correctement, sinon il peut le modifier.

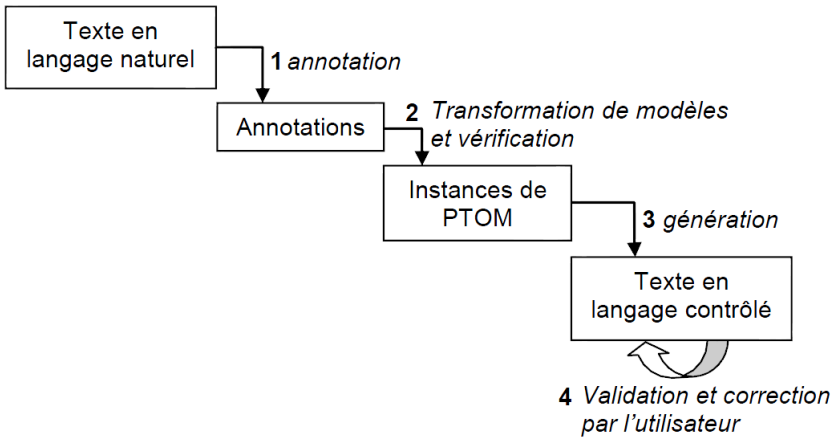


Figure 7.5.6: Processus mise en œuvre pour l'expérimentation RelaxMultiMedias 2

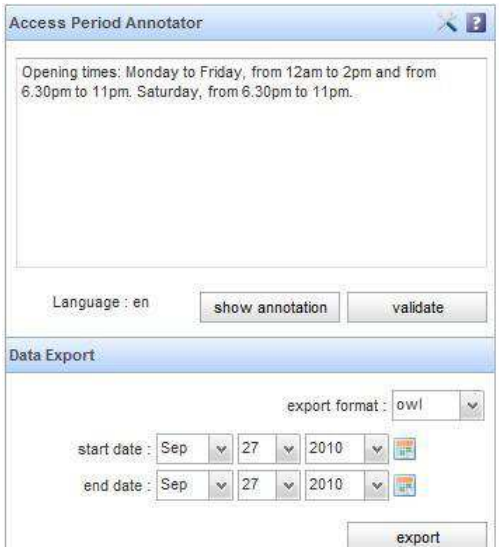
Mise en œuvre avec l'application TKA

Le métamodèle PTOM et sa mise en œuvre JAVA ont été intégrés à l'application web TKA (Temporal Knowledge Acquisition, [Faucher 10c, Mondeca 10]) qui permet de saisir et de structurer des expressions temporelles spécifiant des périodes d'accessibilité à des lieux publics. TKA a été développée dans le cadre de RMM2 et par la société Mondeca.

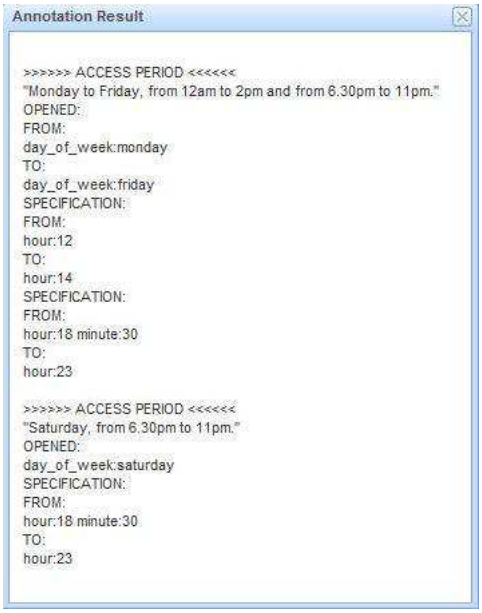
Cette application (cf. figure 7.5.7) dispose d'une zone de saisie de texte en langage naturel (étape a), l'utilisateur lance le processus d'annotation TAL (étape b). Les annotations obtenues sont stockées comme des instances du métamodèle APM (cf. chapitre 6 et type de métamodèle NLP_OUTPUT) (étape c), celles-ci sont transformées et validées en instances de PTOM. Notre contribution dans TKA est la mise en œuvre de la transformation de modèles pour passer de l'étape c à l'étape d. De plus nous proposons à l'utilisateur de générer le texte contrôlé qui paraphrase le modèle généré (étape d).

À partir de ce texte, l'utilisateur peut modifier et corriger les expressions temporelles, y ajouter des événements, etc. Le modèle XMI sera mis à jour à partir de ce texte corrigé. Le processus de validation décrit en section 5.4 peut être appliqué. Pour rappel, la validation s'applique sur chacune des règles de périodicité qui sont de type `FreqWithCalendarRef` et `PeriodicTimeInterval`.

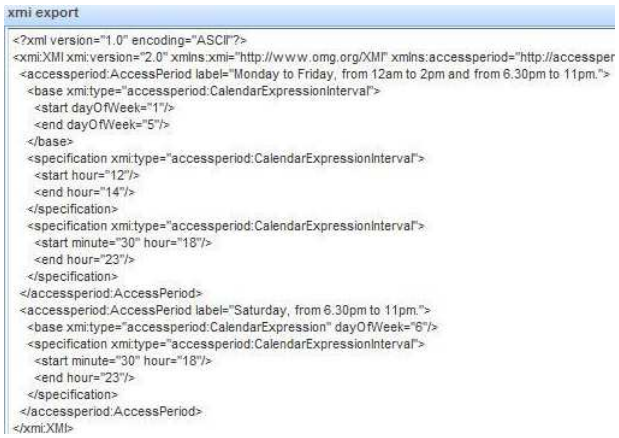
Plus précisément, les `PeriodicInstant` qui leur sont associés sont validés de manière structurelle et sémantique. Dans le cas de `PeriodicTimeInterval`, la validation concerne également la cohérence des deux `PeriodicInstant` qui le composent. Comme nous l'avons vu auparavant au chapitre 6, les sorties de la transformation sont des expressions en intension qui sont des instances soit de `FreqWithCalendarRef` soit de `PeriodicTimeInterval`, ainsi la validation permet d'obtenir un diagnostic complet pour les exemples du corpus.



(a) Saisie



(b) Annotations



(c) Sortie XMI conforme à APM

```
The event occurs periodically according to the rule(s) below
    - rule: from each 12th hour
            to each 14th hour
    using a time span as from each Monday
                         to each Friday
    - rule: from each 30th minute of each 18th hour
            to each 23th hour
    using a time span as from each Monday
                         to each Friday
    end of the periodic temporal occurrence description
end of the event

The event occurs periodically according to the rule(s) below
    - rule: from each 30th minute of each 18th hour
            to each 23th hour
    using a time span as from each Saturday
                         to each Saturday
    end of the periodic temporal occurrence description
end of the event
```

(d) Texte contrôlé en Ptom

Figure 7.5.7: Processus de création d'un texte contrôlé à partir d'un texte en langage naturel avec l'application TKA

Mise en œuvre de manière programmatique

Pour évaluer la chaîne sur l'ensemble du corpus, nous avons écrit un programme en Kermeta qui exécute, sur l'ensemble des modèles, du corpus la transformation de modèles entre APM et Ptom. Durant celle-ci nous enregistrons dans un fichier les éventuelles levées d'exception, ce qui nous fournit un diagnostic. Puis, les modèles de Ptom sont transformés en texte contrôlé.

Les résultats de cette expérimentation sont disponibles sur le web [3]. Nous fournissons :
- les expressions d'origine en langage naturel ;
- les fichiers XMI correspondant au résultat de la phase d'annotation ;
- les fichiers XMI après transformation, ils contiennent les événements instances du métamodèle PTOM ;
- enfin le diagnostic de la phase de vérification.

Les principales problèmes rencontrés viennent du système d'annotation TAL. En effet, la limite de notre transformation de modèles est donnée par celle du système d'annotation. Ainsi nous sommes capables de transformer tous les éléments fournis par le TAL, mais nous ne sommes pas en mesure de compléter ce qui manque. Parmi les 513 expressions, une seule n'est absolument pas comprise par le système d'annotation ce qui nous empêche bien entendu d'en déduire un événement. 16 expressions sont quant à elles partiellement comprises, nous levons d'ailleurs une exception pour avertir l'utilisateur qu'il manque des informations.

Cette expérimentation a permis d'améliorer à la fois les performances des outils TAL et celles de notre transformation. En effet, nous avons notamment intégré une portion de code permettant d'ajouter automatiquement un *jump* si nécessaire. Grâce à nos levées d'exceptions, comme l'invariant détectant un problème d'incohérence de granularité, le système d'annotation a été amélioré par nos partenaires spécialistes du TAL pour éliminer la totalité des erreurs de ce type.

À noter que nous ne fournissons pas de comparatif entre le texte en langage naturel et les événements générés. Celui-ci n'a pas été réalisé et nous estimons que ce travail ne fait pas parti de cette thèse car ce comparatif est très dépendant de la phase d'annotation réalisée par MoDyCo.

7.6 Expérimentation du système d'interrogation PTOM-S

Dans cette section, nous décrivons une expérimentation du système d'interrogation PTOM-S (cf. chapitre 6) suivant plusieurs jeux de données saisies dans l'interface PTOM-S. Ceux-ci vont servir de paramètres aux requêtes (heure et jour de début et de fin, etc).

Cette expérimentation a été réalisée à partir d'expressions temporelles de trois corpus qui ont été analysées par le système TAL de MoDyCo-Mondeca ou par la grammaire STNL, à savoir : Open Data Etalab Musée (« c1 », analysé avec STNL), RMM2 Relaxnews (« c2 », TAL-MoDyCo-Mondeca) et RMM2 Relaxnews (« c3 », STNL).

Chaque jeu de données de PTOM-S représente une expression temporelle en intension, telle que : « *tous les jours de 10h à 19h du lundi au vendredi* ». Cinq requêtes vont exploiter chaque jeu de données pour rechercher les événements, celles-ci sont fondées

3. `http://relaxmultimedia2.univ-lr.fr/ap2tom/benchmark.html`

sur les relations étendues ALLEN*. Nous proposons ainsi de rechercher tous les événements ayant des horaires (heures de début et de fin) soit identiques (*equals**), inclus et contenants (*during** et *Contains**) ou simultanés (*overlaps** et *Overlapped_by**).

Dans les sections suivantes, nous fournissons les jeux de données (paramètres d'exécution) et les résultats des exécutions sont donnés en terme de nombre d'événements retrouvés. Nous les accompagnons d'exemples d'expressions temporelles retrouvées en langage naturel.

7.6.1 Exécution avec sélection sur les horaires et sans sélection sur les jours de la semaine

Dans cette section, nous fournissons les résultats numéraires et textuels pour les cinq relations ALLEN*.

Paramètres d'exécution des requêtes (formulaire web)

début				fin			
heure	minute	jour de la semaine	mois	heure	minute	jour de la semaine	mois
10	00	-	-	19	00	-	-

Les résultats attendus sont des événements :
- pour *equals** : qui commencent à 10h00 et se terminent à 19h00 indépendamment des jours qui pourraient être spécifiés.
- pour *during** : qui commencent avant 10h00 et se terminent après 19h00 indépendamment des jours qui pourraient être spécifiés.
- pour *Contains** : qui commencent après 10h00 et se terminent avant 19h00 indépendamment des jours qui pourraient être spécifiés.
- pour *overlaps** : qui commencent après 10h00 et se terminent après 19h00 indépendamment des jours qui pourraient être spécifiés.
- pour *Overlapped_by** : qui commencent avant 10h00 et se terminent avant 19h00 indépendamment des jours qui pourraient être spécifiés.

Résultats en terme de nombre d'expressions trouvées :

*equals** (85) - *during** (55) - *Contains** (5035) - *overlaps** (1702) - *Overlapped_by** (211)

Extrait de résultats en $equals^*$ sur les horaires

Identifiant	Expressions en intension retrouvées	Commentaires
id1 c2 #1	10h à 19h du lundi au samedi	
id14 c2 #7	De 10h à 19h le lundi et le mercredi. De 10h à 22h du jeudi au dimanche	seul le fragment « De 10h à 19h » a été exploité pour retrouver cet événement
id15 c2 #7	De 10h à 19h le lundi et le mercredi. De 10h à 22h du jeudi au dimanche	seul le fragment « De 10h à 19h » a été exploité pour retrouver cet événement
id17 c2 #8	De 10h à 19h le vendredi et le samedi & de 10h à 18h le dimanche	seul le fragment « De 10h à 19h » a été exploité pour retrouver cet événement
id18 c2 #8	De 10h à 19h le vendredi et le samedi & de 10h à 18h le dimanche	seul le fragment « De 10h à 19h » a été exploité pour retrouver cet événement
id19 c2 #9	De 10h à 19h, nocturne jeudi 8 octobre jusqu'à 21h	seul le fragment « De 10h à 19h » a été exploité pour retrouver cet événement
id21 c2 #10	De 10h à 19h	
id69 c2 #38	Du lundi au vendredi, de 10h à 19h	
id998 c2 #509	Vendredi de 11h à 21h, samedi et dimanche de 10h à 19h	seul le fragment « de 10h à 19h » a été exploité pour retrouver cet événement
id5002 c3 #10	De 10h à 19h	
id10121 c1 #171	10h-19h - ABBAYE D'ARTHOUS, Hastingues	
id10184 c1 #294	10h à 19h - ABBAYE DE VAUCELLES, Les Rues-des-Vignes	
id10188 c1 #320	10h - 19h - ABBAYE ROYALE DE FONTEVRAUD, CENTRE CULTUREL DE L'OUEST, Fontevraud-l'Abbaye	

TABLEAU 7.2: Extrait de résultats en $equals^*$ sur les horaires sans sélection sur les jours de la semaine

Extrait de résultats en *during** sur les horaires

Identifiant	Expressions en intension retrouvées
id186 c2 #94	Lundi à samedi de 9h30 à 19h30
id10477 c1 #1564	7h30-19h30 - ART PROCESS, Paris 11ème

Extrait de résultats en *Contains** sur les horaires

Identifiant	Expressions en intension retrouvées	Commentaires
id74 c2 #41	Du mardi au dimanche de 10h30 à 18h30	
id10323 c1 #639	9h à 12h et 14h à 17h - ARBORETUM DU MAS ROUSSILLON (CCEAME), Canet-en-Roussillon	Les deux fragments « 9h à 12h » et « 14h à 17h » ont été exploités pour retrouver cet événement

Extrait de résultats en *overlaps** sur les horaires

Identifiant	Expressions en intension retrouvées	Commentaires
id24 c2 #12	De 10h à 20h du lundi au samedi, nocturne jusqu'à 21h jeudi et vendredi	seul le fragment « De 10h à 20h » a été exploité pour retrouver cet événement
id10006 c1 #6	18 :00 - 23 :00 - ISTITUZIONE MUSEO CIVICO COMPLESSO MONUMENTALE S. MARIA DEI COMMENDATIS, Maddaloni	

Extrait de résultats en *Overlapped_by** sur les horaires

Identifiant	Expressions en intension retrouvées	Commentaires
id6 c2 #3	9h30 à 11h30 et de 12h30 à 18h30, du mardi au samedi	seul le fragment « 9h30 à 11h30 » a été exploité pour retrouver cet événement
id11893 c1 #4294	9h-17h - CHATEAU D'ESPEYRAN, Saint-Gilles	

Les résultats ci-dessus sont conformes aux événements attendus. À noter qu'aucun jour n'est spécifié dans les paramètres. Ainsi, dans ce cas, tous les événements coïncidants au niveau des horaires sont retrouvés quelque soit leur jour d'occurrence, i.e. : « *tous les jours* », « *tous les vendredis* », « *du lundi au samedi* », etc.

7.6.2 Exécution avec sélection sur les horaires et sélection sur les jours (du lundi au vendredi)

Paramètres d'exécution des requêtes (formulaire web)

début				fin			
heure	minute	jour de la semaine	mois	heure	minute	jour de la semaine	mois
10	00	1 (lundi)	-	19	00	5 (vendredi)	-

Résultats en terme de nombre d'expressions trouvées

*equals** (53) - *during** (46) - *Contains*∗ (4789) - *overlaps** (1606) - *Overlapped_by** (166)

Identifiant	Expressions en intension retrouvées	Commentaires
id1 c2 #1	10h à 19h du lundi au samedi	
id19 c2 #9	De 10h à 19h, nocturne jeudi 8 octobre jusqu'à 21h	seul le fragment « De 10h à 19h » a été exploité pour retrouver cet événement
id21 c2 #10	De 10h à 19h	cet événement a lieu tous les jours
id69 c2 #38	Du lundi au vendredi, de 10h à 19h	
id5002 c3 #10	De 10h à 19h	cet événement a lieu tous les jours
id10121 c1 #171	10h-19h - ABBAYE D ARTHOUS, Hastingues	cet événement a lieu tous les jours
id10184 c1 #294	10h à 19h - ABBAYE DE VAUCELLES, Les Rues-des-Vignes	cet événement a lieu tous les jours
id10188 c1 #320	10h - 19h - ABBAYE ROYALE DE FONTEVRAUD, CENTRE CULTUREL DE L'OUEST, Fontevraud-l'Abbaye	cet événement a lieu tous les jours

Tableau 7.3: Extrait de résultats en *equals** sur les horaires (du lundi au vendredi)

Dans ce cas, les événements ayant lieu « *tous les jours du lundi au vendredi de 10h00 à 19h00* » ont été retrouvés, en effet le paramétrage limite la requête à des événements ayant lieu « *du lundi au vendredi* ». Remarquons que les événements identifiés `id1 c2 #1` et `id21 c2 #10` font également partis des résultats. En effet, les événements ayant lieu « *tous les jours du lundi au samedi* » (`id1 c2 #1`) ou tout simplement « *tous les jours* » (`id21 c2 #10`) possèdent également des occurrences « *tous les jours du lundi au vendredi* ». On remarquera que l'événement identifié `id17 c2 #8` (cf. tableau 7.2) n'est pas présent dans ce résultat car il avait lieu de « *10h00 à 19h00* » uniquement « *tous les vendredi*s ».

7.6.3 Exécution avec sélection sur les horaires et sélection sur les jours (du lundi au samedi)

Paramètres d'exécution des requêtes (formulaire web)

début				fin			
heure	minute	jour de la semaine	mois	heure	minute	jour de la semaine	mois
10	00	1 (lundi)	-	19	00	6 (samedi)	-

Résultats en terme de nombre d'expressions trouvées :

*equals** (46) - *during** (43) - *Contains*∗ (4769) - *overlaps** (1597) - *Overlapped_by** (153)

Identifiant	Expressions en intension retrouvées	Commentaires
id1 c2 #1	10h à 19h du lundi au samedi	
id19 c2 #9	De 10h à 19h, nocturne jeudi 8 octobre jusqu'à 21h	seul le fragment « De 10h à 19h » a été exploité pour retrouver cet événement
id21 c2 #10	De 10h à 19h	cet événement a lieu tous les jours
id5002 c3 #10	De 10h à 19h	cet événement a lieu tous les jours
id10121 c1 #171	10h-19h - ABBAYE D ARTHOUS, Hastingues	cet événement a lieu tous les jours
id10184 c1 #294	10h à 19h - ABBAYE DE VAUCELLES, Les Rues-des-Vignes	cet événement a lieu tous les jours
id10188 c1 #320	10h - 19h - ABBAYE ROYALE DE FONTEVRAUD, CENTRE CULTUREL DE L'OUEST, Fontevraud-l'Abbaye	cet événement a lieu tous les jours

Tableau 7.4: Extrait de résultats en *equals** sur les horaires (du lundi au samedi)

Dans ce cas, les événements ayant lieu « *tous les jours du lundi au samedi de 10h00 à 19h00* » ont été retrouvés, en effet le paramétrage limite la requête à des événements ayant lieu « *du lundi au samedi* ». Remarquons que l'événement identifié `id69 c2 #38` (cf. tableau 7.3) ne fait plus parti des résultats car il a lieu « *tous les jours du lundi au vendredi de 10h00 à 19h00* », mais pas « *le samedi* ».

7.6.4 Exécution avec sélection sur les horaires et sélection sur les jours (du lundi au dimanche)

Paramètres d'exécution des requêtes (formulaire web)

début				fin			
heure	minute	jour de la semaine	mois	heure	minute	jour de la semaine	mois
10	00	1 (lundi)	-	19	00	7 (dimanche)	-

Résultats en terme de nombre d'expressions trouvées :

*equals** (45) - *during** (42) - *Contains** (4758) - *overlaps** (1594) - *Overlapped_by** (153)

Identifiant	Expressions en intension retrouvées	Commentaires
id19 c2 #9	De 10h à 19h, nocturne jeudi 8 octobre jusqu'à 21h	seul le fragment « De 10h à 19h » a été exploité pour retrouver cet événement
id21 c2 #10	De 10h à 19h	cet événement a lieu tous les jours
id5002 c3 #10	De 10h à 19h	cet événement a lieu tous les jours
id10121 c1 #171	10h-19h - ABBAYE D'ARTHOUS, Hastingues	cet événement a lieu tous les jours
id10184 c1 #294	10h à 19h - ABBAYE DE VAUCELLES, Les Rues-des-Vignes	cet événement a lieu tous les jours
id10188 c1 #320	10h - 19h - ABBAYE ROYALE DE FONTEVRAUD, CENTRE CULTUREL DE L'OUEST, Fontevraud-l'Abbaye	cet événement a lieu tous les jours

Tableau 7.5: Extrait de résultats en *equals** sur les horaires (du lundi au dimanche)

Dans ce cas, seuls les événements ayant lieu « *tous les jours de 10h00 à 19h00* » ont été trouvés, car le paramétrage limite la requête à des événements ayant lieu « *tous les jours* » autrement dit « *du lundi au dimanche* ».

7.7 Conclusion

Dans ce chapitre nous avons décrit plusieurs cas d'utilisation du métamodèle Ptom notamment avec sa mise en œuvre pour traiter des exemples relatifs aux travaux académiques de Terenziani et le cas expérimental des Festivités du marché de Gand. Ces deux cas d'étude utilisent conjointement les relations (`structural`), temporelles

(`temporal`) et qualitative (Allen*). Dans tous les cas, l'expressivité de Ptom s'est avérée adaptée à la modélisation des événements récurrents à la fois pour définir des propriétés temporelles mais également pour préciser des relations mutuelles entre événements.

Concernant l'expression du temps en intension, Ptom a été instancié avec divers types d'expressions : des instants et intervalles périodiques qui parfois possèdent des positions relatives comme pour la règlementation de la pêche à pied ou bien des périodes d'accès pour les corpus provenant d'agences de presse.

Que ce soient les 513 expressions du corpus Relaxnews ou les 35252 cas du Ministère de la Culture et de la Communication, l'ensemble de ces exemples ont permis de tester et de faire évoluer la transformation de modèles entre les métamodèles de période d'accessibilité et Ptom. Plus largement, nous avons montré l'utilisation de Ptom, des grammaires et de la transformation à travers la plate-forme de gestion de ressources journalistiques du projet RelaxMultiMedias 2.

L'ensemble de ces instances de Ptom ont été intégrées à la base de données d'événements pour être consultées *via* le système d'interrogation d'expressions temporelles en intension Ptom-s. Ce dernier permet de retrouver les événements avec différentes requêtes se conformant aux relations étendues Allen*.

Ptom a également été couplé à un SMA pour permettre la modélisation d'expressions temporelles liées à des textes règlementaires afin de déterminer le statut concernant l'autorisation de la pratique de la pêche à pied professionnelle.

Chapitre 8

Conclusion et perspectives

8.1 Enjeux et démarche générale

Le travail mené au long de la rédaction de cette thèse recouvre, à l'image du projet ANR qui l'a suscité, un périmètre scientifique et technologique très large, et s'inscrit dans les préoccupations de plusieurs communautés scientifiques. Le cycle de vie de l'activité de gestion de contenu journalistique dédié aux loisirs et à la culture concerne en amont, les communautés des linguistes et du traitement de la langue naturelle. Il convient ensuite de transmettre le relais aux concepteurs de systèmes d'information. Toutes les facettes de cette dernière discipline sont impliquées (modélisation, persistance, interrogation, visualisation). À chaque étape, il s'agit de faire coopérer des espaces de modélisations différents et naturellement de procéder à la vérification et à la validation des entrées et sorties. Pour atteindre ces objectifs, et compte tenu de la complexité des problèmes traités, l'IDM apparaît comme une nécessité afin d'organiser et d'automatiser pour réutiliser des solutions génériques garantes de qualité (intégrité, évolutivité).

Plus que des données, il s'agit bien ici de manipuler de la connaissance. Les structures d'événements temporels portent une forte sémantique qu'il s'agit de préserver et

d'exploiter. Nos travaux relèvent également en partie de l'intelligence artificielle et font appel au raisonnement formel.

Que ce soit en ce qui concerne la langue naturelle exprimant des assertions dont le sujet est le temps, les structures algébriques, géométriques ou topologiques portant sur les intervalles, les logiques temporelles, la spécification des calendriers et des référentiels temporels, il existe des normes, des publications, des langages, des outils, en résumé de nombreux travaux par rapport auxquels il convient de se situer, et dont nombre de résultats doivent être réutilisés ou étendus. Quelquefois il est nécessaire de créer de nouveaux concepts pour pallier des lacunes.

Il est clair que cette thèse ne peut mener à son terme les réflexions dans tous les domaines abordés. Nous avons travaillé dans l'esprit suivant :

- Centrer notre contribution sur un noyau complet et cohérent qui est un métamodèle pivot pour les événements temporels. Ce métamodèle représente la structure centrale vers laquelle les éléments pertinents des modèles métiers amont sont traduits ou, *a contrario*, à partir de laquelle sont extraits les éléments pertinents pour les utilisateurs. Les règles formelles liées à cette métamodélisation permettent de garantir la correction et la cohérence des informations peuplant les modèles correspondants.
- Nous nous sommes attachés à démontrer systématiquement, par des exemples concrets didactiques issus de la littérature, ou réels et traités dans le cadre du projet ANR RelaxMultiMedias 2, la faisabilité de nos propositions.
- Au-delà du noyau central que nous venons d'évoquer, notre travail ouvre la porte à de nombreux développements qu'ils soient théoriques, en particulier sur des questions de raisonnement temporel, ou algorithmiques lorsque l'on envisage des environnements d'interrogation des bases de connaissance temporelles constituées. En se fondant sur notre métamodélisation, il est maintenant possible de préciser, quel que soit le contexte applicatif, les questions à résoudre, les optimisations à apporter, les développements à spécifier. Le cadre de notre métamodèle permet aussi de juger de la pertinence et de l'efficacité des solutions proposées.

8.2 Principales contributions

De façon plus précise, nous évoquerons maintenant nos apports concernant successivement le métamodèle d'événements temporels, l'interfaçage avec les applications externes sources et cibles, les structures et propriétés mathématiques qui sous tendent notre vue des propriétés temporelles. Enfin nous discuterons la mise en œuvre et l'intérêt majeur de l'approche IDM que nous avons appliquée tout au long de ce travail.

8.2.1 Métamodèles fondamentaux

Notre contribution repose sur le métamodèle Ptom qui spécifie d'une part le concept d'événement composite et d'autre part permet les propriétés temporelles de ces évé-

nements. Elle inclut également un modèle de calendrier grégorien qui est un élément indispensable de l'ensemble pour permettre le raisonnement temporel. D'autres types de calendriers pourraient être produits selon la même démarche.

Événements

PTOM reprend des éléments de la norme EventsML G2, les enrichis *via* des spécialisations et des extensions utiles, mais surtout leur donne une représentation semi formelle selon le paradigme objet (UML) assortie de contraintes formelles OCL précisant la sémantique des éléments modélisés. Le paradigme objet est mieux adapté à la réutilisation et à l'extension.

D'un point de vue structurel, les événements sont décomposés en une structure hiérarchique permettant de gérer de façon cohérente plusieurs niveaux d'abstraction.

L'intégration explicite de métadonnées permet de prendre en compte les informations de type autre que temporel (spatial, thématique...). De la même façon, nous modélisons les relations mutuelles non temporelles au niveau du métamodèle d'événement. Les relations temporelles sont naturellement encapsulées dans le modèle de propriétés temporelles.

PTOM permet naturellement l'import d'événements initialement spécifiés en EventsML G2 ou en iCalendar. Pour l'import à partir d'autres sources, il convient de développer des modules spécifiques à chaque source, mais génériques vis-à-vis des données. Ce module prend en entrée le schéma des événements PTOM qui est fixé, et celui, variable, qui dépend de la source considérée.

Propriétés temporelles

Nous partons de deux constatations fondamentales et corrélées :

- L'une reconnaît que les événements périodiques portent une forte sémantique temporelle par rapport aux événements ponctuels, et qu'en cela, on doit s'intéresser en priorité à leur représentation fidèle et opérationnelle.
- L'autre consiste à préférer la représentation des ensembles de dates en intension plutôt qu'en extension. Deux raisons justifient ce choix : d'une part le caractère économique du codage d'une expression synthétique par rapport à celui d'une suite de dates concrètes dans le cas d'événements périodiques, d'autre part, et surtout, le fait que l'expression en intension explicite la sémantique de périodicité alors que l'extension la masque.

Naturellement, PTOM permet néanmoins le codage de tout type de règles y compris pour les événements non périodiques.

Nous avons étudié les publications des linguistes et de la communauté TAL pour spécifier des formes de représentation de la connaissance temporelle qui soient robustes et consensuelles (non liées à un métier spécifique). Les travaux du laboratoire MoDyCo et le rapport OGRE offrent une synthèse sur cette question. Les types de structures

PTOM qui représentent l'intension de règles d'occurrences périodiques reprennent donc les concepts reconnus pertinents par OGRE.

Dans une règle de périodicité PTOM, chaque élément constitutif signifiant est identifié et directement accessible dans le modèle, constituant ainsi un argument naturel d'interrogation et de raisonnement.

PTOM, de par la sémantique propre de sa modélisation UML, porte un ensemble de contraintes liées à la structure et aux multiplicités des éléments de modélisation (en accord avec le métamodèle d'UML). Des contraintes additionnelles sont exprimées en OCL. Celles-ci sont exprimées par un ensemble d'invariants et de préconditions vérifiées à l'exécution dans les modules de peuplement (de PTOM). Ainsi, la correction et la cohérence des instances saisies dans le modèle sont elles assurées.

Dans cet ordre d'idées nous travaillons sur une extension d'OCL qui prend en compte les primitives temporelles et les concepts fondamentaux que sont les éléments calendaires, les instants, les intervalles et leurs relations mutuelles.

PTOM réutilise les classes fondamentales définies dans la norme ISO 19108 qui référence la norme ISO 8601:2004 pour la mise en œuvre du type `DATE`, et inclut des classes outils permettant de contrôler une partie des propriétés géométriques et topologiques des concepts « instant » et « intervalle ».

Un apport significatif de PTOM réside dans la définition de deux types de décomposition d'événements complexes [Faucher 12a, Faucher 12b]. L'une structurelle (`structural`) reconnaît le fait qu'un événement est constitué de sous-événements (e.g. : « *Tour de France / étapes* »), l'autre temporelle (`temporal`) qui rattache un événement périodique à chacun des événements que constituent ses occurrences (« *Tour de France / {... Tour 2012, Tour 2013, Tour 2014, ...}* »). Ces deux points de vue sont spécifiés de façon indépendante, mais impliquent des contraintes conjointes. Une des richesses de PTOM est de permettre ce type de spécification et d'en gérer les propriétés.

La spécification objet de PTOM possède une contrepartie textuelle donnée sous la forme d'une grammaire formelle proche du langage naturel [Faucher 10b]. Cette grammaire, automatiquement synchronisée avec le métamodèle *via* les outils de l'IDM, est particulièrement utile pour faciliter – en reformulant ses termes – le dialogue avec l'utilisateur, lors des phases de vérification et de peuplement du modèle.

PTOM possède enfin un sous-ensemble spécialisé qui exprime les contraintes topologiques ALLEN* opérant sur les intervalles non convexes et exploite les résultats que nous avons établis à partir des travaux de LIGOZAT.

Calendrier grégorien

Dans le domaine applicatif où nous nous sommes placés, il n'est pas possible de spécifier des propriétés temporelles sans références calendaires. La norme ISO 19108 donne des éléments pour cela, mais ceux-ci apparaissent sous forme textuelle et s'avèrent insuffisants dans la perspective d'une utilisation pratique en l'état.

Nous donnons une spécification objet (modèles UML accompagnés de contraintes

OCL) du calendrier grégorien. Un point important est que cette spécification du calendrier exploite le métamodèle PTOM. Ceci est naturel, puisqu'il est indiscutable que les éléments constituant le calendrier sont des événements pour la plupart périodiques. « *Années* », « *mois* », « *jours* » sont des événements conformes à PTOM dont les relations mutuelles sont à la fois de type `structural` et `temporal`. Enfin, les extensions ALLEN* que nous avons définies sont tout à fait appropriées pour spécifier les relations topologiques entre ces événements calendaires.

Ainsi, la spécification du calendrier grégorien est de fait un exemple particulier de mise en œuvre de PTOM et des concepts d'ALLEN*.

En retour, la richesse sémantique portée par le calendrier grégorien, ainsi spécifiée, est intensivement utilisée par les outils logiciels attachés à PTOM, pour vérifier [Faucher 11] et, plus généralement, gérer la formulation des expressions temporelles attachées aux événements.

8.2.2 Gestion de l'interopérabilité

L'interopérabilité présente pour nous principalement quatre aspects, interfaçant respectivement PTOM avec les sources produisant du contenu, avec les solutions de gestion de la persistance des informations peuplant le modèle, avec les outils et environnements dédiés au raisonnement temporel et avec les utilisateurs finaux qui interrogent le système et visualisent des événements et leurs métadonnées (temporelles en ce qui nous concerne).

Interface avec les fournisseurs de contenu et les experts

Nous avons traité des sources de types divers, produisant du contenu plus ou moins structuré. Pour le texte brut, nous sous sommes appuyés sur les compétences TAL qui après un prétraitement, restituent un texte décoré d'annotations temporelles qui se prêtent à l'analyse automatique et à la traduction vers PTOM dès lors qu'un métamodèle d'annotation est disponible [Faucher 10c]. Le traducteur est fondé sur la spécification du métamodèle d'annotation et du métamodèle d'expression temporelle dans PTOM. Dans de nombreux cas, le système d'annotation produit par le TAL est incomplet ou ambigu. Il est alors nécessaire de faire intervenir un expert humain (journaliste) pour lever les ambiguïtés et compléter les informations.

Lors de cette phase, PTOM propose au journaliste trois éléments d'aide dans sa tâche :

- une formulation de l'expression temporelle traitée dans un langage formel proche du langage naturel et donc aisément compréhensible (la grammaire évoquée plus haut) ;
- des propositions par défaut (auto-complétion) pour compléter les informations manquantes ;
- une vérification finale de la correction et de la cohérence de la règle temporelle candidate que le journaliste a élaborée. Pour réaliser cela, le module d'aide interroge

Ptom et le modèle de calendrier pour extraire les éléments demandés [Faucher 12a, Faucher 11].

Interface avec les environnements de raisonnement temporel

Ptom est conçu avec la volonté de représenter un ensemble de concepts et de représentations proches du langage naturel. La sémantique des propriétés des événements et des calendriers y est exprimée et utilisée par des outils internes en particulier à des fins de vérification et de validation. Cependant sur le plan du raisonnement, il est illusoire et non pertinent de vouloir doter ces outils de la puissance et de l'efficacité de ceux construits sur des DSL nativement dédiés à la mise en œuvre des logiques temporelles.

Notre attitude en la matière est de conserver à Ptom son rôle de pivot et, grâce à l'IDM, de construire des passerelles vers les environnements spécialisés de raisonnement. Les éléments d'une question logique sont automatiquement traduits et leur analyse est déléguée aux outils *ad hoc*. Une fois encore, les traducteurs sont spécifiques à un environnement, mais génériques par rapport aux termes de la question à traiter. De tels traducteurs restent à développer.

Interface avec les utilisateurs finaux

Nous donnons dans Ptom, les fondements nécessaires pour bâtir un système d'interrogation de base de connaissance. Nous avons développé des algorithmes et des prototypes qui démontrent la faisabilité et résolvent les requêtes élémentaires. Ainsi un utilisateur final formule une requête composée par les propriétés du métamodèle temporal, et le système retourne les événements correspondants s'ils existent dans la base.

Les plates-formes de visualisation dédiées aux informations temporelles proposent une gamme variée de composants graphiques (widgets) pour illustrer les propriétés temporelles d'événements. Ces applications doivent être configurées pour s'adapter aux types et aux valeurs des données utilisateur. Les propriétés d'introspection des langages dans lesquels sont développés ces plates-formes, permettent de scruter leurs bibliothèques et d'en tirer des métamodèles spécifiques représentant l'ensemble des paramètres de configuration et les modes d'appariement des composants et des données à représenter [Faucher 12a, Faucher 10a].

Un adaptateur peut être conçu qui configure les données Ptom selon les formats requis, réalise l'appariement souhaité et finalement configure l'application cible externe. Cette démarche est automatisée *via* l'IDM, mais nécessite naturellement une expertise pour superviser l'appariement.

8.2.3 Théories sous-jacentes et raisonnement

Une grande partie de la sémantique des événements composites (répétitifs) est contenue dans Ptom lui-même et exprimée par les structures UML et les contraintes OCL

additionnelles. D'autres sont explicitées dans le code des applications mettant en œuvre le modèle. Une autre partie de la sémantique est portée par le modèle de calendrier utilisé, lui aussi exprimé avec les concepts de PTOM.

Cependant, l'expression des contraintes selon les modes décrits ci-dessus ne peut porter que sur les occurrences des événements. L'expressivité (le niveau d'abstraction) est donc faible, et doit mettre en jeu des quantificateurs universels et existentiels, ce qui induit une complexité forte du processus de raisonnement. On connait le côté pénalisant d'un tel système, par rapport à ceux où les règles concernent les constructions d'un niveau d'abstraction supérieur.

Nous avons construit un tel système, qui exprime de façon native les relations topologiques mutuelles entre événements non convexes. Nous explicitons ces correspondances entre les concepts de haut niveau (les relations d'ALLEN*) et leur contrepartie décrite avec le langage de niveau inférieur utilisant les quantificateurs sur les occurrences.

Les résultats que nous donnons directement sur les relations d'ALLEN* (composition, filtres) permettent d'exprimer un système de règles et de développer un raisonnement efficace au niveau des abstractions, et d'ignorer les occurrences qui restent sous-jacentes.

8.2.4 Mise en œuvre de l'IDM

Comme nous l'avons évoqué ci-dessus, le sigle PTOM recouvre un ensemble conséquent de modèles, métamodèles et langages. Pour mémoire, les métamodèles d'événements composites et d'expression temporelles, une partie de la norme ISO 19108, le modèle de calendrier et la base de concepts et de règles propres à ALLEN*. Il englobe aussi la grammaire formelle qui fait *écho* aux concepts de PTOM sous une forme textuelle. Il convient également d'inclure les modèles de transformation (directs ou de niveau supérieur) qui permettent, en interne, de synchroniser l'ensemble de ces éléments et de gérer l'interopérabilité avec les sources de données et les experts, les environnements de persistance, les outils de raisonnement et de visualisation.

Hors du paradigme et des outils IDM, il nous semble illusoire de tenter d'intégrer de manière efficace et maintenable l'ensemble des modèles et la panoplie d'application évoqués ci-dessus. Outre les bénéfices de l'usage de l'IDM (EMF, KERMETA, XTEXT), quant à l'organisation générale et à la gestion du projet et de son cycle de vie, nous mettrons en avant les facilités de prototypage et de test, la synchronisation automatique inter-modèles.

8.3 Perspectives

Comme nous l'affirmons en début de ce chapitre, il n'est pas réaliste dans le cadre d'une thèse, de mener à leurs termes l'ensemble des réflexions initialisées dans notre travail. Parmi les points restant à développer, nous citerons ceux nous apparaissant comme prépondérants.

Granularité

Sur un plan fondamental, notre travail aborde et gère la question de la gestion de la granularité, mais de manière diffuse, lors de la modélisation des règles temporelles des événements, de celles du calendrier et de l'expression de contraintes. La granularité est intrinsèquement liée à la structure composite des événements et en particulier, au modèle (agrégatif) des éléments calendaires. Cette question est également centrale dans le processus d'interrogation du modèle. Nous donnons un ensemble de références bibliographiques sur le sujet, mais un travail pertinent serait d'exploiter les concepts de granularité au regard de ceux de PTOM et d'ALLEN*.

Allen*

Nous donnons les propriétés essentielles de l'ensemble des relations ALLEN*. Elles constituent le socle minimal qui permet de mettre en œuvre nos modèles.

Sur un plan fondamental et opératoire, d'autres propriétés seraient à étudier. En particulier, il serait fructueux de chercher à réduire le *Garbage* pour améliorer l'efficacité des calculs et des applications. Une direction serait de considérer un opérateur qui associe son complémentaire à toute relation d'ALLEN*, e.g. : le complément de (2,4,6,9) dans $\Pi(2,3)$ est (0,2,4,6,9,12) dans $\Pi(3,3)$. L'appartenance d'une relation au *Garbage* peut coexister avec le fait que son complément figure dans ALLEN*. Ainsi pourrait-on, selon les cas, considérer une relation ou son complément et étendre le champ d'application de nos propositions initiales. Dans cette approche, l'usage d'intervalles cycliques permet d'augmenter le pouvoir d'expression et la pertinence de l'opérateur complément.

Interrogation du modèle

Comme nous l'avons dit précédemment, nous donnons dans PTOM les fondements nécessaires pour bâtir un système d'interrogation. En revanche, nous n'avons pas développé de système de raisonnement proprement dit. Nous entendons par là, un système qui exploiterait en profondeur, les calculs de transitivité sur les relations d'ALLEN et ALLEN*, qui réécrirait et résoudrait les requêtes à différents niveaux de granularité et enfin tirerait parti de la connaissance intrinsèque au calendrier et exprimée par la sémantique de son modèle dans PTOM.

Une somme importante de travail peut être menée dans cette direction, en choisissant parmi les différents environnements et technologies de raisonnements possibles, e.g. : moteurs de règles, logiques temporelles, ontologies et logiques de description, problématique SAT, etc.

Interface avec outils raisonnement

Sur un plan opérationnel, et en liaison directe avec le point précédent, il serait impératif, de construire, *via* l'IDM, des ponts vers les environnements de raisonnement

temporels existants (LUSTRE, MARTE. . .). Cela apporterait des retombées immédiates sur le plan opératoire ainsi qu'un bénéfice évident en termes de rapprochement de communautés scientifiques (temps réel, systèmes d'information, ingénierie des connaissances . . .).

Interface avec la communauté TAL

Notre collaboration avec les linguistes et spécialistes du traitement de la langue naturelle a été particulièrement fructueuse. D'abord sur le plan des concepts puis sur celui des outils. Il reste encore beaucoup à apprendre et à exploiter des relations avec ces communautés. La sémantique de nos modèles émane de leurs travaux et les algorithmes qui pourront être développés gagneront à poursuivre cette collaboration.

Le travail en ce domaine réside dans la facilitation du dialogue, la mise en parallèle des concepts, des terminologies et des pratiques de chacun. Derrière ces vocables imprécis, se cachent des modèles partagés et des outils communs. À titre d'exemple, la plate-forme UIMA [1] (Unstructured Information Management Architecture) propose une infrastructure pour faire inter-opérer des outils de traitement de la langue et de fouille de données.

Gestion de la persistance

Disposant d'un modèle objet, il est facile, et c'était là une de nos motivations, d'utiliser directement les outils de mise en correspondance Objet/Relationnel (ActiveRecord, Hibernate) ou Objet/RDF (ActiveRDF, EMFTriple [Hillairet 11]) pour assurer la persistance du modèle.

L'approche ontologique ouvre des perspectives dans l'hypothèse du « monde ouvert », ce qui semble particulièrement adapté à la modélisation élémentaire de propriétés temporelles et de calendriers dont la spécification est rarement close. De plus, l'usage d'ontologies et de bases de connaissance RDF, ouvre des possibilités intéressantes de développer des applications pratiques dans le contexte des données ouvertes.

Au-delà de ces solutions généralistes, il serait intéressant d'envisager des solutions de persistance *ad hoc*, qui tiennent compte de la sémantique de Ptom et exploite sa structure pour optimiser la gestion d'une base de données ou de connaissances. Nous avons déjà insisté sur l'intérêt majeur de traiter (persistance, interrogation) des expressions en intension plutôt qu'en extension.

Enfin, si le modèle Ptom nous paraît abouti et stable, plusieurs points de cette thèse peuvent être approfondis sur un plan applicatif, à côté de ceux plus fondamentaux que nous venons de citer.

Un premier travail consiste à poursuivre l'expérimentation, en utilisant notre modèle et son outillage sur d'autres exemples réels, cela étant mené dans un souci de valida-

1. `http://uima.apache.org`

tion plus poussée, et de mesure plus précise de l'efficacité et de la flexibilité de nos propositions. Dans cet esprit, l'utilisation de PTOM par d'autres équipes, comme le laboratoire GÉOMER dans le cadre de MAGIS [Faucher 10e, Faucher 10d], que celle qui est à l'origine de sa conception est une étape nécessaire et bienvenue.

Le modèle étant stabilisé et validé, il sera maintenant possible de finaliser la documentation technique sous la forme d'un document normatif en bonne et due forme, organisé et complet.

L'ambition de notre travail est d'avoir mis en place les éléments nécessaires pour mener à bien les perspectives énoncées ci-dessus et également contribué, par nos expérimentations, à prouver leur faisabilité.

Annexes

Annexe A

Textes règlementaires

A.1 Arrêtés municipaux ou préfectoraux

« Article 25 : Les heures de fonctionnement des marchés découverts sont fixées de 7 heures à 14 heures 30, sauf les samedis et dimanches où les emplacements seront évacués au plus tard à 15 heures, soit une demi-heure après la clôture des ventes, et sauf pour les marchés d'après midi qui feront l'objet d'un arrêté municipal spécifique fixant leurs horaires. »

FIGURE A.1.1: Extrait de l'arrêté municipal de la Ville de Paris du 20 septembre 2006 - Règlement des marchés découverts alimentaires

ARRETE N° 371/2001

réglementant l'exercice de la pêche à pied des donax (tellines)
sur le littoral du service des affaires maritimes
de Douarnenez-Camaret (Finistère)
modifié par l'arrêté 411/2001 DRAM du 30.09.2004

Article 6 :

La pêche à pied à titre professionnelle des donax sur le gisement classé du littoral du service des affaires maritimes de Douarnenenez-Camaret est interdite :

- de nuit, entre 21 heures et 06 heures,
- le samedi
- ainsi que du 1er juillet au 31 août inclus.

La pêche sur les plages de Kerloc'h et de l'Aber est interdite toute l'année.

Toutefois, un arrêté du préfet de la région Bretagne peut fixer annuellement une période d'ouverture de la pêche sur la plage de l'Aber, après une visite de prospection et sur avis de l'Ifremer.
Le temps de pêche autorisé est limité à 3 heures avant et 3 heures après la basse mer, selon l'horaire indiqué par l'annuaire des marées de Douarnenez

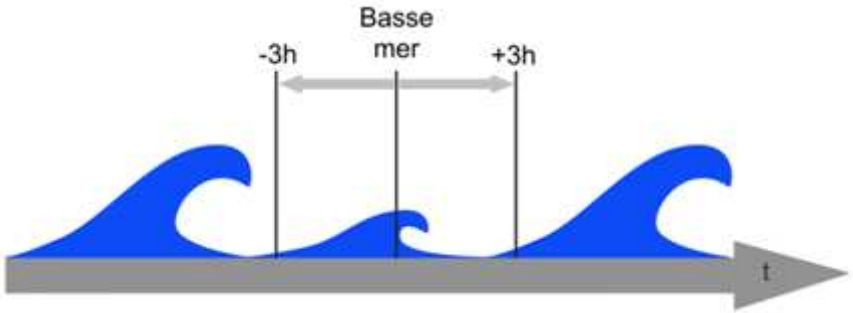


FIGURE A.1.2: Extrait de la règlementation de la pêche à pied professionnelle à la Telline dans la zone de Douarnenez (France)

ANNEXE II
à l'arrêté n° 55/2007 du 25 mai 2007 réglementant l'exercice de la pêche maritime de loisir pratiquée à pied, à la nage ou sous marine dans le département de la Manche

Tailles de capture, périodes de pêche, engins autorisés et quantités maximales de pêche par jour et par pêcheur pour les espèces de coquillages, poissons, crustacés et céphalopodes.

Nom de l'espèce	Taille minimale de capture	Période de pêche autorisée	Engin autorisé	Quantité maximale de pêche autorisée par pêcheur et par jour
COQUILLAGES				
Praires (*Venus verrucosa*)	4 cm	Du 1° septembre au 30 avril	fourche, fourche à cailloux (entre Pirou et Agon) pelle triangulaire, griffe à dents	100 individus
Amandes de mer (*Glycymeris glycymeris*)				
Coquilles Saint Jacques (*Pecten maximus*)	11 cm	Du 1° octobre au 15 mai	couteau, croc, épuisette	30 individus
Ormeaux (*Haliotis tuberculata*)	9 cm	Du 1°septembre au 1° mai, 3 jours avant et 3 jours après une pleine ou une nouvelle lune	Couteau, croc	12 individus
Huîtres creuses (*Crassostrea gigas*)	7 cm dans la plus grande dimension	Du 1° septembre au 30 avril		72 individus
Huîtres plates (*Crassostrea edulis*)				40 individus
Moules (*Mytilus edulis*)	4 cm		griffe à dents, couteau	350 individus ou 5 litres

FIGURE A.1.3: Extrait de l'arrêté préfectoral concernant la pêche à pied de loisir aux ormeaux en Manche (France)

Nom de l'espèce	**Taille minimale de capture**	**Période de pêche autorisée**	**Engin autorisé**	**Quantité maximale de pêche autorisée par pêcheur et par jour**
Congre (*Conger conger*)	58 cm	Toute l'année (en fonction de l'ouverture du quota de l'espèce concernée)	Ligne, palangre, gaffe, paillot	Non limité
Orphie (*Belone belone*)	45 cm		Ligne, nasse, senne à mulets (*soumis à autorisation*)	
Anguille (*Anguilla anguilla*)	40 cm	Du 1° janvier au 15 août	Ligne, nasse, paillot, palangre	
Saumon (*Salmo salar*)	70 cm du 15 mars au 15 juillet 50 cm du 15 juillet au 15 octobre	Du 15 mars au 15 octobre Entre le lever et le coucher du soleil	Ligne, palangre *soumis à autorisation* : filet droit, carrelet, senne à mulets.	1 individu

FIGURE A.1.4: Extrait de l'arrêté préfectoral concernant la pêche du saumon en Manche (France)

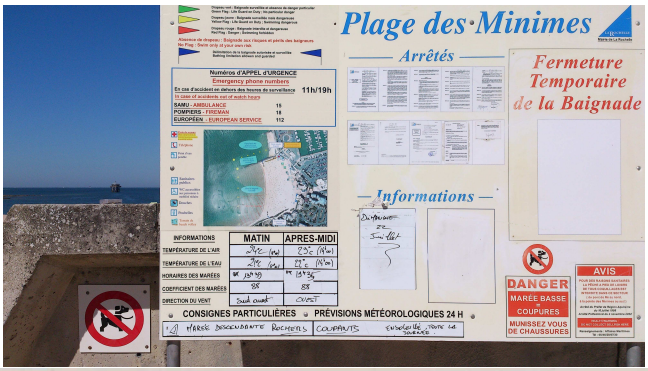
A.2 Pêche à pied de loisirs sur le front de mer de La Rochelle

L'autorisation de la pêche à pied de loisirs est définie en utilisant un intervalle périodique « du 1er février au 30 novembre ». Les horaires sont quant à eux précisés en utilisant des positions relatives, i.e. du lever au coucher du soleil. Voir figure A.2.1.

FIGURE A.2.1: Règlementation de la Pêche à pied de loisirs sur le front de mer de La Rochelle

A.3 Horaire de surveillance de la plage

La figure A.3.1 vise à montrer une ambiguïté sur la définition des horaires de surveillance d'une plage. En effet, le « slash » dans l'expression « 11h/19h » peut être interprété hors contexte de différentes manières, i.e. : « et » ou bien « à ». Pour interpréter cette expression, l'on suppose que la plage est surveillée la journée et pas seulement à 11h et 19h, par conséquent le « slash » doit se comprendre « à ». Donc en cas d'accident, il est nécessaire d'appeler les secours aux numéros indiqués entre 19h et 11h du jour suivant.



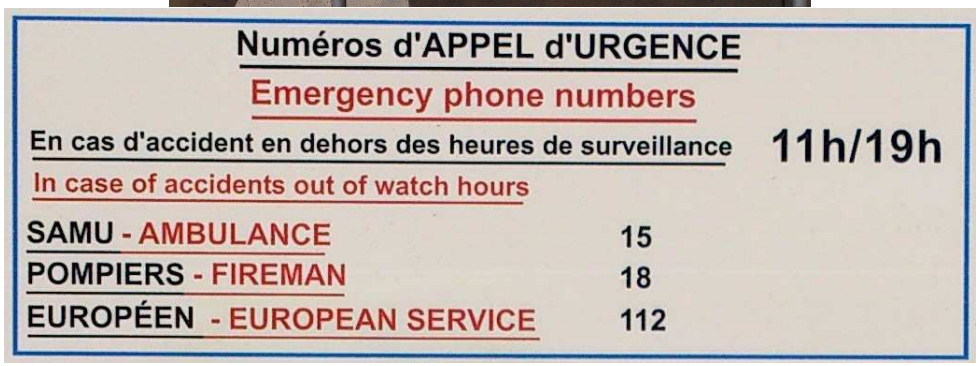


FIGURE A.3.1: Information sur les horaires de surveillance de la plage

Annexe B

Exemples d'expressions temporelles en intension dans la vie courante

B.1 Périodes d'accès à des lieux publics

B.1.1 Instants périodiques multiples

Les instants périodiques multiples (IPM) définissent les heures de départ de la visite. Ainsi pour le cas de la figure B.1.1, cinq règles de périodicité sont à associer à cet événement.

FIGURE B.1.1: Instants périodiques multiples pour le départ de visites

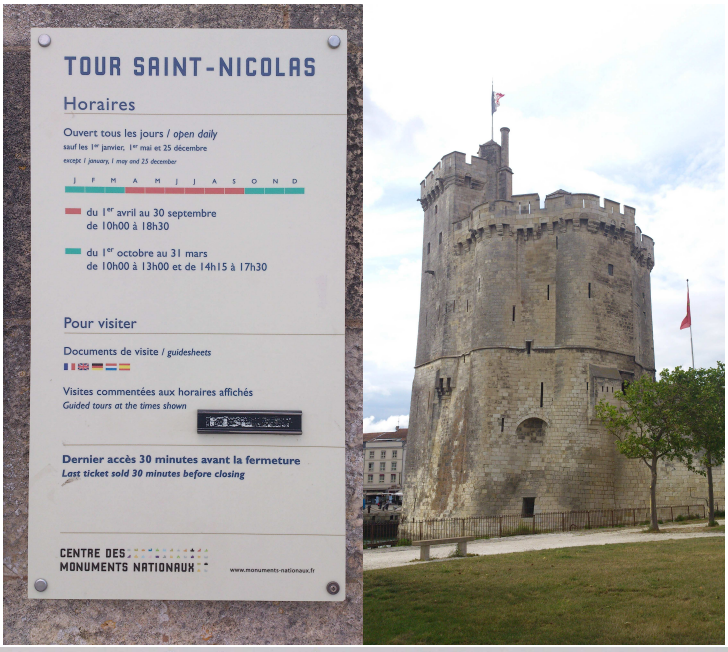
B.1.2 Intervalles périodiques multiples

Les intervalles périodiques multiples (InPM) définissent les horaires d'ouverture du parc sur une année entière. Ainsi pour le cas de la figure B.1.2, quatre règles de périodicité sont à associer à cet événement.

FIGURE B.1.2: Promenade du Peyrou à Montpellier

B.1.3 Intervalle périodique avec un début et une fin utilisant des références différentes

Dans le cas de la figure B.1.3, une première expression désigne des horaires d'ouverture : « *du 1er avril au 30 septembre de 10h00 à 18h30* », une seconde concerne la période d'accessibilité « *du 1er octobre au 31 mars de 10h00 à 13h00 et de 14h15 à 17h30* ». Pour la seconde, l'expression implique que la fin soit « *chaque 31 mars* », mais de « *chaque année suivante* ». Dans PTOM, nous avons introduit la notion de *jump* pour indiquer qu'il faut se référer à l'année qui suit celle utilisée par le début. Un nombre également associé au *jump* pour fixer le nombre d'unités à décaler (e.g. : 1 année, 1 jour, etc). Un *jump* est un entier positif toujours associé à l'instant périodique décrivant la fin d'un intervalle.



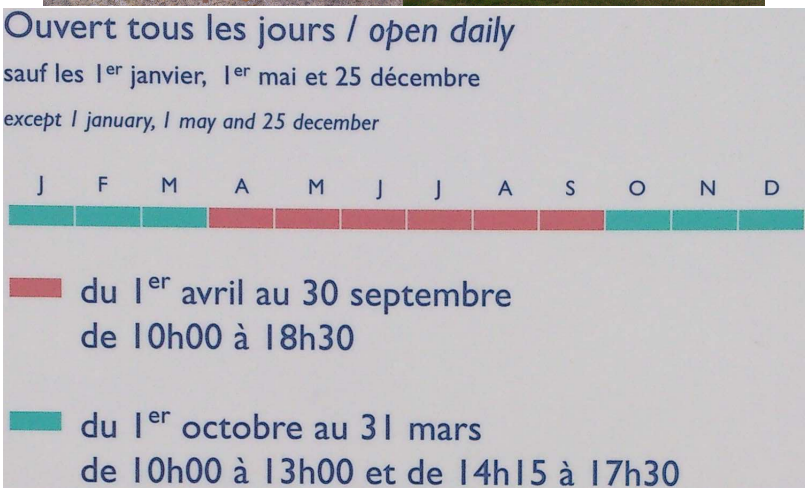
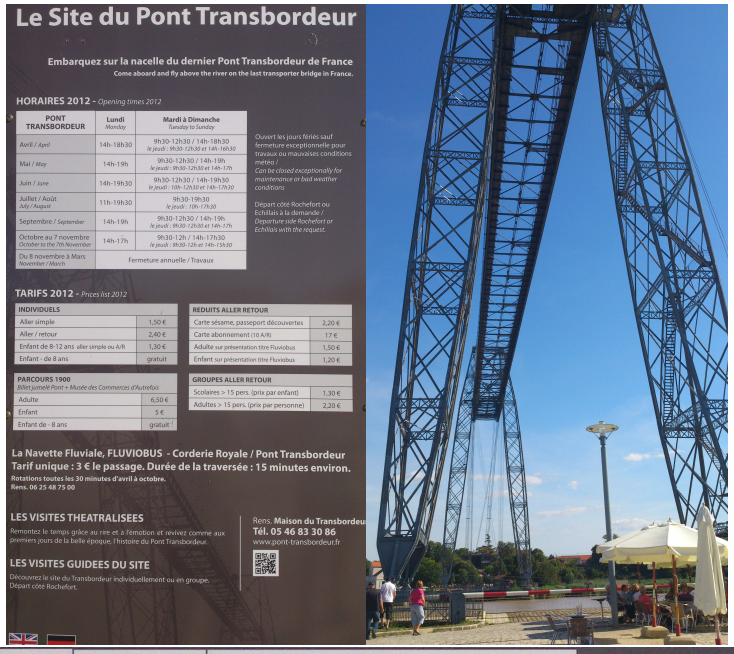


FIGURE B.1.3: La Tour Saint Nicolas de La Rochelle - utilisation d'un *jump*

B.1.4 Dépendance entre événements

La figure B.1.4 donne un exemple de dépendance entre des événements. En effet, ce site est fermé exceptionnellement « pour travaux ou mauvaises conditions météo » qui sont deux phénomènes non-périodiques et très aléatoires, aucune prévision ne peut être faite.



PONT TRANSBORDEUR	Lundi *Monday*	Mardi à Dimanche *Tuesday to Sunday*
Avril / *April*	14h-18h30	9h30-12h30 / 14h-18h30 *le jeudi : 9h30-12h30 et 14h-16h30*
Mai / *May*	14h-19h	9h30-12h30 / 14h-19h *le jeudi : 9h30-12h30 et 14h-17h*
Juin / *June*	14h-19h30	9h30-12h30 / 14h-19h30 *le jeudi : 10h-12h30 et 14h-17h30*
Juillet / Août *July / August*	11h-19h30	9h30-19h30 *le jeudi : 10h-17h30*
Septembre / *September*	14h-19h	9h30-12h30 / 14h-19h *le jeudi : 9h30-12h30 et 14h-17h*
Octobre au 7 novembre *October to the 7th November*	14h-17h	9h30-12h / 14h-17h30 *le jeudi : 9h30-12h et 14h-15h30*
Du 8 novembre à Mars *November / March*	Fermeture annuelle / Travaux	

Ouvert les jours fériés sauf fermeture exceptionnelle pour travaux ou mauvaises conditions météo / *Can be closed exceptionally for maintenance or bad weather conditions*

Départ côté Rochefort ou Echillais à la demande / *Departure side Rochefort or Echillais with the request.*

FIGURE B.1.4: Le Pont Transbordeur de Rochefort - dépendance entre événements

La figure B.1.5 montre un autre cas d'utilisation d'un *jump*, cette fois-ci pour les horaires « *11h à 2h du matin* », sous-entendu « *de 11h de chaque jour à 2h de chaque jour suivant* ». Dans ce cas, il apparaît une ambiguïté avec deux informations contradictoires entre ce qui est écrit en haut et sur la banderole en bas respectivement « *d'avril à septembre - 11h à 2h du matin* » et « *de 10h à 2h - 7jours/7* », l'heure de début est différente : $11h \neq 10h$.

Le langage naturel est parfois utilisé pour compléter des données déjà structurées, comme le montre l'annexe C avec une note qui, par exemple, précise une exception :

« *fermé le mardi après-midi et le jeudi matin* ».

FIGURE B.1.6: Interdiction de stationner relative au coucher et lever du soleil près de Rochefort

B.2 Horaires de gares avec des d'ambiguïtés

La figure B.2.1 donne les horaires d'ouverture du guichet de trois gares. On peut voir les ambiguïtés induites par le non formatage des dates et plus particulièrement des périodes. Pour les trois exemples, les périodes horaires sont définies par une heure de début et de fin séparées par un « slash » sauf pour les horaires de la gare de Royan « *Du Lundi au samedi et le 14 juillet et le 15 août : 09h00 20h00* » où le « slash » a été certainement oublié, i.e. : « *09h00 / 20h00* ». Ces oublis et imprécisions engendrent lors du traitement automatique des ambiguïtés. D'ailleurs le « slash » est dans d'autres cas utilisés comme un séparateur entre instants et non comme un séparateur pour identifier le début et la fin d'une période.

http://www.ter-sncf.com/Region/poitou_charentes/gare/La-Rochelle.aspx
http://www.ter-sncf.com/Region/poitou_charentes/gare/Rochefort.aspx
http://www.ter-sncf.com/Region/poitou_charentes/gare/Royan.aspx

FIGURE B.2.1: Imprécisions et ambiguïtés sur les horaires d'ouverture de gares

FIGURE B.1.5: Bar du France 1, La Rochelle - utilisation d'un *jump* et une ambiguïté sur l'heure de début d'ouverture (« *11h* » en haut, « *10h* » sur la banderole en bas)

Annexe C

Périodes d'accessibilité en intension structurées

Cette annexe présente des données issues de l'OpenData Etalab concernant l'annuaire des administrations françaises, le jeu de données se nomme « Service-public.fr - Annuaire de l'administration - Base de données locales » [1]. Ici nous présentons des extraits pour les services de la mairie de La Rochelle qui concernent leurs périodes d'accessibilité.

Dans ce corpus, les données sont sous forme très structurées. Les informations sont organisées et les champs bien définis. En ce qui concerne les dates et plus particulièrement les périodes d'ouverture des administrations, un bloc XML `Ouverture` est défini dans lequel on peut trouver des plages journalières `PlageJ`, chaque `PlageJ` contient une ou des plages horaires : `PlageH`. Ceci permet donc de définir des plages horaires et journalières pour décrire des périodicités et finalement définir en intension les périodes d'ouverture des administrations.

```
---- Espace public ----
<Ouverture>
  <PlageJ début="lundi" fin="vendredi">
    <PlageH début="09:00:00" fin="12:30:00" />
    <PlageH début="14:00:00" fin="17:00:00" />
  </PlageJ>
</Ouverture>
```

FIGURE C.0.1: Un modèle d'accessibilité avec deux périodes d'ouverture journalières

1. `http://www.data.gouv.fr/donnees/view/Service-public.fr`

```
---- Elections ----
<Ouverture>
  <PlageJ début="lundi" fin="lundi">
    <PlageH début="08:45:00" fin="16:30:00" />
  </PlageJ>
  <PlageJ début="mardi" fin="vendredi">
    <PlageH début="08:30:00" fin="16:30:00" />
  </PlageJ>
</Ouverture>
```

FIGURE C.0.2: Un modèle d'accessibilité avec deux périodes d'ouverture différentes suivant deux périodes de la semaine

```
---- Archives municipales ----
<Ouverture>
  <PlageJ début="lundi" fin="vendredi">
    <PlageH début="09:00:00" fin="12:30:00" />
    <PlageH début="13:30:00" fin="17:00:00" />
    <Note>fermé le mardi après-midi et le jeudi matin</Note>
  </PlageJ>
</Ouverture>

---- Urbanisme ----
<Ouverture>
  <PlageJ début="lundi" fin="vendredi">
    <PlageH début="08:15:00" fin="12:30:00" />
    <PlageH début="13:30:00" fin="17:00:00" />
    <Note>Pour déposer un dossier</Note>
  </PlageJ>
  <PlageJ début="lundi" fin="vendredi">
    <PlageH début="08:30:00" fin="12:30:00" />
    <PlageH début="13:30:00" fin="17:00:00" />
    <Note>Pour les commerces du secteur sauvegardé mardi, jeudi et
       vendredi de 8h30 à 12h30 et le lundi de 13h30 à 17h.</Note>
  </PlageJ>
  <PlageJ début="lundi" fin="vendredi">
    <PlageH début="08:30:00" fin="12:30:00" />
    <PlageH début="13:30:00" fin="17:00:00" />
    <Note>Pour un conseil personnalisé sauf le lundi matin, mardi matin et
        jeudi matin</Note>
  </PlageJ>
</Ouverture>
```

FIGURE C.0.3: Un modèle d'accessibilité avec des exceptions en note

Annexe D

Contrainte de sur- ou sous-spécification

	cas n°	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**
	statut	√	∼	√	√	∼	√	√	∼	√	√	∼	√	√	∼	√	√	∼	√	√	∼	√	√	∼	∼	×	√	×	√
N	not rank.isNull()	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	1	0	1	0
N	isNamed	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
N	isLast	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
N+1	not rank.isNull()	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0				
N+1	isNamed	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0				
N+1	isLast	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1				

√ : valide

∼ : sur-spécification ⇒ alerte

× : sous-spécification ⇒ erreur

TABLEAU D.1: Table de décision pour la contrainte de sur- ou sous-spécification

cas n°	Exemple d'expression illustrant le cas	Statut	Diagnostic
1	3ème lundi de chaque 1er mois de janvier de chaque siècle	√	-
2	3ème lundi de chaque mois de janvier de chaque siècle	∼	à simplifier ▶ cas 3
3	3ème lundi de chaque mois de janvier	√	-
4	3ème lundi de chaque 1er mois de chaque siècle	√	-
5	3ème lundi de chaque mois de chaque siècle	∼	à simplifier ▶ cas 6
6	3ème lundi de chaque mois	√	-
7	Chaque lundi de chaque 1er mois de janvier de chaque siècle	√	-
8	Chaque lundi de chaque mois de janvier de chaque siècle	∼	à simplifier ▶ cas 9
9	Chaque lundi de chaque mois de janvier	√	-
10	Chaque lundi de chaque 1er mois de chaque siècle	√	-
11	Chaque lundi de chaque mois de chaque siècle	∼	à simplifier ▶ cas 12
12	Chaque lundi de chaque mois	√	-
13	Chaque 3ème jour de chaque 1er mois de janvier de chaque siècle	√	-
14	Chaque 3ème jour de chaque mois de janvier de chaque siècle	∼	à simplifier ▶ cas 15
15	Chaque 3ème jour de chaque janvier	√	-
16	Chaque 3ème jour de chaque 1er mois de chaque siècle	√	-
17	Chaque 3ème jour de chaque mois de chaque siècle	∼	à simplifier ▶ cas 18
18	Chaque 3ème jour de chaque mois	√	-
19	Chaque jour de chaque 1er mois de janvier de chaque siècle	√	-
20	Chaque jour de chaque mois de janvier de chaque siècle	∼	à simplifier ▶ cas 21
21	Chaque jour de chaque mois de janvier	√	-
22	Chaque jour de chaque 1er mois de chaque siècle	√	-
23	Chaque jour de chaque mois de chaque siècle	∼	à simplifier ▶ cas 28
24	Chaque jour de chaque mois	∼	à simplifier ▶ cas 28
25	Chaque 3ème lundi	×	expression incomplète
26	Chaque lundi	√	-
27	Chaque 3ème jour	×	expression incomplète
28	Chaque jour	√	-

√ : valide

∼ : sur-spécification ⇒ alerte

× : sous-spécification ⇒ erreur

TABLEAU D.2: Exemples pour la table de décision D.1

Annexe E

Grammaires

Les grammaires, qui suivent, sont définies avec l'outil XTEXT, elles ont été utilisées pour générer à la fois des éditeurs textuels, des analyseurs et générateurs de textes. Chaque éditeur de texte et analyseur permettent de générer un modèle conforme au métamodèle concerné, alors que les générateur produisent un texte conforme à la grammaire à partir de modèles. Il est ainsi possible de passer très facilement de textes contrôlés à des modèles et inversement.

E.1 Expression temporelle répétitive

```
// automatically generated by Xtext
grammar l3i.time.tom.cstx.Tom with org.eclipse.xtext.common.Terminals
import
"platform:/resource/l3i.time.model/model/ecore/periodicaltime.ecore"
import "http://www.eclipse.org/emf/2002/Ecore" as ecore

// Racine du métamodèle
EventSet returns EventSet :
  (imports+=Import)*
  (events+=Event)+
;

Import:
  'import' importURI=STRING
;

Event returns Event :
  'The event' (name=STRING)? 'occurs'
    (concreteTemporalOccurrence=ConcreteTemporalOccurrence |
    periodicTemporalOccurrence=PeriodicTemporalOccurrence)
    ('Temporal Definition Mapping = {'
      mappingsTom2Calendar+=STRING
      (',' mappingsTom2Calendar+=STRING)*
    '}')?
  'end of the event'
;

ConcreteTemporalOccurrence returns ConcreteTemporalOccurrence :
  'concretely' ('(named ' name=STRING')')?

  ('[verifies the intention of '
  intention=[PeriodicTemporalOccurrence] ']')?

  temporalProperties+=TM_Primitive
  ('and ' temporalProperties+=TM_Primitive)*

;
```

FIGURE E.1.1: Grammaire Xtext pour les expressions temporelles

```
PeriodicTemporalOccurrence returns PeriodicTemporalOccurrence :
  {PeriodicTemporalOccurrence}
  'periodically' ('(named ' name=STRING') ')?
  ('- rule:' periodicRules+=PeriodicRule)*

  // Exceptions
  (exceptionsContainer=TemporalException)?
;

TemporalException returns TemporalException :
  ('except'
    (
    exceptPrimitives+=TM_Primitive |
    exceptPeriodicRules+=PeriodicRule |
    ('a relative position' exceptRelativePositions+=
       PeriodicRelativePosition)
    )
  )
  ('and except'
    (
    exceptPrimitives+=TM_Primitive |
    exceptPeriodicRules+=PeriodicRule |
    ('a relative position' exceptRelativePositions+=
       PeriodicRelativePosition)
    )
  )*
;

TM_Primitive returns TM_Primitive :
  ('on ' (TM_Instant|NamedInstant)) | TM_Period
;

TM_Period returns TM_Period :
  (
    'from ' begin=(TM_Instant|NamedInstant) 'to ' end=(TM_Instant|
       NamedInstant)
  )
;

TM_Instant returns TM_Instant :
  position=TM_Position
;

Jump returns Jump:
    '[n+'value=INT']'
;
```

FIGURE E.1.2: Grammaire Xtext pour les expressions temporelles (suite 1)

```
NamedInstant returns NamedInstant :
  position=TM_Position
  ('(identified by 'name=ID')')
  //position gives an indication to calculate the
  //ConcreteRelativePosition: position=2001-04-03 [..] before [..]
  ('in relation with another occurrence '
  featureRelativePositions=ConcreteRelativePosition)?
;

TM_Position returns TM_Position :
  dateTime8601=DateTimeRule
;

DateTimeRule returns DateTime :
  STRING
;

PeriodicRule returns PeriodicRule :
  PRuleWithGivenFrequency | PRuleWithImplicitFrequency
;

PRuleWithGivenFrequency returns PRuleWithGivenFrequency :
  FreqWithDurationRef | FreqWithCalendarRef
;

FreqWithDurationRef returns FreqWithDurationRef :
  ('(identified by 'name=ID')')?
  (
    (times=INT 'times during an undetermined period')
    |
    // times is fixed to 1
    ('during one' referenceDuration=Duration 'period')
    |
    (times=INT 'times during one' referenceDuration=Duration 'period')
  )

  ('and starts on' startTime=TM_Instant)?
  ('using a time span as ' restriction+=PeriodicTimeSpan)*

  // Duration
  ('an occurrence duration is ' occurrenceDuration=Duration)?
  (
    ('time extent ' ruleExtent=TM_Period)
    |
    ('time extent starting from ' ruleExtentStart=TM_Instant)
  )?
;
```

FIGURE E.1.3: Grammaire Xtext pour les expressions temporelles (suite 2)

```
FreqWithCalendarRef returns FreqWithCalendarRef :
  ('(identified by 'name=ID')')?
  (
    // times is fixed to 1
    ('during ' calendarPeriodicInstant=PeriodicInstant) |
    (times=INT 'times ' periodicTimeInterval=PeriodicTimeInterval) |
    (times=INT 'times during ' calendarPeriodicInstant=PeriodicInstant)
  )

  ('and starts on' startTime=TM_Instant)?
  ('using a time span as ' restriction+=PeriodicTimeSpan)*

  // Duration
  ('an occurrence duration is ' occurrenceDuration=Duration)?
  (
    ('time extent ' ruleExtent=TM_Period)
    |
    ('time extent starting from ' ruleExtentStart=TM_Instant)
  )?
;

PRuleWithImplicitFrequency returns PRuleWithImplicitFrequency :
  ('(identified by 'name=ID')')?
  periodicTimeInterval=PeriodicTimeInterval

  ('and starts on' startTime=TM_Instant)?
  ('using a time span as ' restriction+=PeriodicTimeSpan)*

  // Duration
  ('an occurrence duration is ' occurrenceDuration=Duration)?
  (
    ('time extent ' ruleExtent=TM_Period)
    |
    ('time extent starting from ' ruleExtentStart=TM_Instant)
  )?
;
```

FIGURE E.1.4: Grammaire Xtext pour les expressions temporelles (suite 3)

```
Duration returns Duration :
  {Duration}
  (accuracy=AccuracyType)?
  (centuries=INT 'centuries')? (years=INT 'years')?
  (months=INT 'months')? (weeks=INT 'weeks')?
  (days=INT 'days')? (hours=INT 'hours')?
  (minutes=INT 'minutes')? (seconds=INT 'seconds')?
;

RelativeDuration returns Duration :
  {Duration}
  'with a gap of'
  (accuracy=AccuracyType)?
  (centuries=INT 'centuries')? (years=INT 'years')?
  (months=INT 'months')? (weeks=INT 'weeks')?
  (days=INT 'days')? (hours=INT 'hours')?
  (minutes=INT 'minutes')? (seconds=INT 'seconds')?
;

CalendarPeriodicDescriptor returns CalendarPeriodicDescriptor :
  ( DayDescriptor | MonthDescriptor |
  CalendarDescriptor | DayEventDescriptor)
;

Rank returns Rank :
  NumericRank | QualitativeRank | Follower
;

NumericRank returns NumericRank :
  value=INT( 'th' | 'st' | 'nd' | 'rd' )
;

QualitativeRank returns QualitativeRank :
  value=RankEnum
;

Follower returns Follower:
  '[n+'value=INT']' ;

enum RankEnum :
  first = 'first' | first = 'premier' |
  last = 'last' | last = 'dernier' ;
```

FIGURE E.1.5: Grammaire Xtext pour les expressions temporelles (suite 4)

```
PeriodicInstant returns PeriodicInstant :
  AbsoluteTemporalExpression | PeriodicRelativePosition
;

PeriodicTimeInterval returns PeriodicTimeInterval :
  ('from ' begin=PeriodicInstant) ('to ' end=PeriodicInstant)?
;

PeriodicTimeSpan returns PeriodicTimeSpan :
  ('from ' begin=PeriodicInstant)
  (
    ('to ' end=PeriodicInstant) |
    ('during a period of' span=Duration)
  )?
;

AbsoluteTemporalExpression returns AbsoluteTemporalExpression :
  ('each' descriptors+=CalendarPeriodicDescriptor)
  ('of each' descriptors+=CalendarPeriodicDescriptor)*
;

DayDescriptor returns DayDescriptor :
  (rank=Rank)? day=Day
;

MonthDescriptor returns MonthDescriptor :
  (rank=Rank)? month=Month
;

CalendarDescriptor returns CalendarDescriptor :
  (rank=Rank)? value=CalendarUnit
;

DayEventDescriptor returns DayEventDescriptor :
  (rank=Rank)? dayEvent=DayEvent
;


enum AccuracyType :
  equal_or_less='<=' | equal_or_more='>=' |
  less_than='<' | more_than='>' |
  approximative='~' | equal='='
;
```

FIGURE E.1.6: Grammaire Xtext pour les expressions temporelles (suite 5)

```
enum Day :
    Monday = 'Monday' | Monday = 'lundi' |
    Tuesday = 'Tuesday' | Tuesday = 'mardi' |
    Wednesday = 'Wednesday' | Wednesday = 'mercredi' |
    Thursday = 'Thursday' | Thursday = 'jeudi' |
    Friday = 'Friday' | Friday = 'vendredi' |
    Saturday = 'Saturday' | Saturday = 'samedi' |
    Sunday = 'Sunday' | Sunday = 'dimanche'
;

enum Month :
    January = 'January' | January = 'Janvier' |
    February = 'February' | February = 'Février' |
    March = 'March' | March = 'Mars' |
    April = 'April' | April = 'Avril' |
    May = 'May' | May = 'Mai' |
    June = 'June' | June = 'Juin' |
    July = 'July' | July = 'Juillet' |
    August = 'August' | August = 'Aout' |
    September = 'September' | September = 'Septembre' |
    October = 'October' | October = 'Octobre' |
    November = 'November' | November = 'Novembre' |
    December = 'December' | December = 'Décembre'
;

enum CalendarUnit :
    century | year | month | week |
    day | hour | minute | second
;

enum DayEvent :
    morning | afternoon | midnight|
    lunch | dinner | evening | night
;
```

FIGURE E.1.7: Grammaire Xtext pour les expressions temporelles - éléments calendaires (suite 6)

```
PeriodicRelativePosition returns PeriodicRelativePosition :
    (
        (timeGap=RelativeDuration)?
        relativePosition=TM_RelativePositionGap
    )
    refersTo=[PeriodicRule]
;

enum TM_RelativePositionGap returns TM_RelativePosition :
    before |
    after |
    equals;
```

FIGURE E.1.8: Grammaire Xtext pour les expressions temporelles - positions relatives (suite 7)

E.2 Structure d'un événement

```
// automatically generated by Xtext
grammar l3i.time.esh.cstx.Esh with org.eclipse.xtext.common.Terminals
import
"platform:/resource/l3i.time.model/model/ecore/periodicaltime.ecore"
import "platform:/plugin/org.eclipse.uml2.uml/model/UML.ecore" as uml
import "http://www.eclipse.org/emf/2002/Ecore" as ecore

EventSet returns EventSet :
    (imports+=Import)*
    (events+=Event)+
;

Import:
    'import' importURI=STRING
;
EventHierarchy returns EventHierarchy:
    (' as' name=STRING)?
    '{'
        (structural+=[Event])
        (','structural+=[Event])*
    '}'
;

Event returns Event :
    'The event' (name=STRING)? ('occurs' (
            concreteTemporalOccurrence=ConcreteTemporalOccurrence |
            periodicTemporalOccurrence=PeriodicTemporalOccurrence)
        )?
        ('structural' eventHierarchy+=EventHierarchy)*
        ('Temporal Definition Mapping = {'
            mappingsTom2Calendar+=STRING
            (',' mappingsTom2Calendar+=STRING)*
        '}')?
        ('umlClass' umlClass=[uml::Class])?
        ('umlInstance' umlInstance=[uml::InstanceSpecification])?
    'end of the event'
;
```

FIGURE E.2.1: Grammaire permettant d'instancier la relation *structural*

E.3 Horaire d'accessibilité en français

```
grammar l3i.time.nlp.cstx.Stnl with org.eclipse.xtext.common.Terminals
generate stnl "http://www.time.l3i/nlp/cstx/Stnl"

TemporalExpressionSet:
    temporalExpressions+=
    (PeriodicInstant | PeriodicInterval |
    PeriodicInstantMultiple | PeriodicIntervalMultiple)
;
PeriodicInstant:
    ('à'|'à partir de'|'dès')?ptime=Time
    (duration=Duration)?
;
PeriodicInterval:
    ('de'|'à'|'à partir de'|'dès')?
    ptime_start=Time('|'|' | '|'-'|'- '|' - '|' -'|' -- '|
    '->'|' -> '|' à ')ptime_end=Time
;
PeriodicInstantMultiple:
    ('à'|'à partir de'|'dès')?ptimes+=Time
    //séparateurs
    (('|'|' |'|' | '|'| '|' et
    '|' & '|'/'|'/ '|' / ')
    ptimes+=Time)*
;
PeriodicIntervalMultiple:
    pintervals+=PeriodicInterval
    //séparateurs
    (('|'|' |'|' | '|'| '|' et
    '|' & '|'/'|'/ '|' / ')
    pintervals+=PeriodicInterval)*
;
Time:
    hours=INT('h'|'heures'|':')(minutes=INT)?
;
Duration:
    (' Durée 'hours=INT'h'(minutes=INT)?)|
    (' (durée 'minutes=INT')')
;
```

FIGURE E.3.1: Grammaire d'horaire d'accessibilité en français

Annexe F

Du métamodèle linguistique de période d'accessibilité au métamodèle d'événements composites récurrents

F.1 Ressource pour l'exemple #3 du corpus Relaxnews RMM2

9h30 à 11h30 et de 12h30 à 18h30, du mardi au samedi.

FIGURE F.1.1: Texte en langage naturel pour l'exemple #3 du corpus Relaxnews RMM2

```
1  <xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:
       accessperiod="http://accessperiod/1.0">
2  <accessperiod:AccessPeriod label="9h30 &#xe0; 11h30 et de 12h30 &#xe0; 18
       h30, du mardi au samedi..">
3    <base xmi:type="accessperiod:CalendarExpressionInterval">
4      <start minute="30" hour="9"/><end minute="30" hour="11"/>
5    </base>
6    <specification xmi:type="accessperiod:CalendarExpressionInterval">
7      <start dayOfWeek="2"/><end dayOfWeek="6"/>
8    </specification>
9  </accessperiod:AccessPeriod>
10 <accessperiod:AccessPeriod label="9h30 à 11h30 et de 12h30 à 18h30, du
       mardi au samedi..">
11   <base xmi:type="accessperiod:CalendarExpressionInterval">
12     <start minute="30" hour="12"/><end minute="30" hour="18"/>
13   </base>
14   <specification xmi:type="accessperiod:CalendarExpressionInterval">
15     <start dayOfWeek="2"/><end dayOfWeek="6"/>
16   </specification>
17 </accessperiod:AccessPeriod>
18 </xmi:XMI>
```

FIGURE F.1.2: Modèle APM pour l'exemple #3 du corpus Relaxnews RMM2

```
<?xml version="1.0" encoding="ASCII"?>
<periodicaltime:EventSet xmi:version="2.0" xmlns:xmi="http://www.omg.org/
    XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:
    periodicaltime="http://periodicaltime/1.0">
<events name="9h30 à 11h30 et de 12h30 à 18h30, du mardi au samedi..">
<periodicTemporalOccurrence>
<periodicRules xsi:type="periodicaltime:PRuleWithImplicitFrequency">
  <restriction>
    <begin xsi:type="periodicaltime:AbsoluteTemporalExpression">
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="day
          "><rank xsi:type="periodicaltime:NumericRank" value="2"/></
          descriptors>
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="
          week"/>
    </begin>
    <end xsi:type="periodicaltime:AbsoluteTemporalExpression">
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="day
          "><rank xsi:type="periodicaltime:NumericRank" value="6"/></
          descriptors>
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="
          week"/>
    </end>
  </restriction>
<periodicTimeInterval>
    <begin xsi:type="periodicaltime:AbsoluteTemporalExpression">
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="
          minute"><rank xsi:type="periodicaltime:NumericRank" value
          ="30"/></descriptors>
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="
          hour"><rank xsi:type="periodicaltime:NumericRank" value="9"/></
          descriptors>
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="day
          "/>
    </begin>
    <end xsi:type="periodicaltime:AbsoluteTemporalExpression">
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="
          minute"><rank xsi:type="periodicaltime:NumericRank" value
          ="30"/></descriptors>
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="
          hour"><rank xsi:type="periodicaltime:NumericRank" value="11"/></
          descriptors>
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="day
          "/>
    </end>
</periodicTimeInterval>
</periodicRules>
</periodicTemporalOccurrence>
</events>
```

FIGURE F.1.3: Modèle PTOM pour l'exemple #3 du corpus Relaxnews RMM2

```
<events name="9h30 à 11h30 et de 12h30 à 18h30, du mardi au samedi..">
<periodicTemporalOccurrence>
<periodicRules xsi:type="periodicaltime:PRuleWithImplicitFrequency">
  <restriction>
    <begin xsi:type="periodicaltime:AbsoluteTemporalExpression">
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="day
          "><rank xsi:type="periodicaltime:NumericRank" value="2"/></
          descriptors>
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="
          week"/>
    </begin>
    <end xsi:type="periodicaltime:AbsoluteTemporalExpression">
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="day
          "><rank xsi:type="periodicaltime:NumericRank" value="6"/></
          descriptors>
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="
          week"/>
    </end>
  </restriction>
  <periodicTimeInterval>
    <begin xsi:type="periodicaltime:AbsoluteTemporalExpression">
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="
          minute"><rank xsi:type="periodicaltime:NumericRank" value
          ="30"/></descriptors>
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="
          hour"><rank xsi:type="periodicaltime:NumericRank" value="12"/></
          descriptors>
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="day
          "/>
    </begin>
    <end xsi:type="periodicaltime:AbsoluteTemporalExpression">
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="
          minute"><rank xsi:type="periodicaltime:NumericRank" value
          ="30"/></descriptors>
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="
          hour"><rank xsi:type="periodicaltime:NumericRank" value="18"/></
          descriptors>
      <descriptors xsi:type="periodicaltime:CalendarDescriptor" value="day
          "/>
    </end>
    </periodicTimeInterval>
  </periodicRules>
</periodicTemporalOccurrence>
</events>
</periodicaltime:EventSet>
```

FIGURE F.1.4: Modèle PTOM pour l'exemple #3 du corpus Relaxnews RMM2 (suite)

The event "9h30 a 11h30 et de 12h30 a 18h30, du mardi au samedi.." occurs periodically according to the rule(s) below
- rule : from each 30 th minute of each 9 th hour of each day to each 30 th minute of each 11 th hour of each day
using a time span as from each 2 th day of each week to each 6 th day of each week
end of the event

The event "9h30 a 11h30 et de 12h30 a 18h30, du mardi au samedi.." occurs periodically according to the rule(s) below
- rule : from each 30 th minute of each 12 th hour of each day to each 30 th minute of each 18 th hour of each day
using a time span as from each 2 th day of each week to each 6 th day of each week
end of the event

FIGURE F.1.5: Texte en langage naturel contrôlé pour l'exemple #3 du corpus Relaxnews RMM2

F.2 Ressource pour l'exemple #81 du corpus Relaxnews RMM2

Lundi à samedi de 10h30 à 12h30 et de 14h à 18h30.

FIGURE F.2.1: Texte en langage naturel pour l'exemple #81 du corpus Relaxnews RMM2

```
1  <accessperiod:AccessPeriod xmi:version="2.0" xmlns:xmi="http://www.omg.org
       /XMI" xmlns:accessperiod="http://accessperiod/1.0" label="Lundi &#xe0;
       samedi de 10h30 &#xe0; 12h30 et de 14h &#xe0; 18h30..">
2    <base xmi:type="accessperiod:CalendarExpressionInterval">
3      <start dayOfWeek="1"/><end dayOfWeek="6"/>
4    </base>
5    <specification xmi:type="accessperiod:CalendarExpressionInterval">
6      <start minute="30" hour="10"/><end minute="30" hour="12"/>
7    </specification>
8    <specification xmi:type="accessperiod:CalendarExpressionInterval">
9      <start hour="14"/><end minute="30" hour="18"/>
10   </specification>
11 </accessperiod:AccessPeriod>
```

FIGURE F.2.2: Modèle APM pour l'exemple #37 du corpus Relaxnews RMM2

```
1  <?xml version="1.0" encoding="ASCII"?>
2  <periodicaltime:EventSet xmi:version="2.0" xmlns:xmi="http://www.omg.org/
       XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:
       periodicaltime="http://periodicaltime/1.0">
3    <events name="Lundi à samedi de 10h30 à 12h30 et de 14h à 18h30..">
4      <periodicTemporalOccurrence>
5        <periodicRules xsi:type="periodicaltime:PRuleWithImplicitFrequency">
6          <restriction>
7            <begin xsi:type="periodicaltime:AbsoluteTemporalExpression">
8              <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                   value="day">
9                <rank xsi:type="periodicaltime:NumericRank" value="1"/>
10             </descriptors>
11             <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                   value="week"/>
12           </begin>
13           <end xsi:type="periodicaltime:AbsoluteTemporalExpression">
14             <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                   value="day">
15               <rank xsi:type="periodicaltime:NumericRank" value="6"/>
16             </descriptors>
17             <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                   value="week"/>
18           </end>
19         </restriction>
20         <periodicTimeInterval>
21           <begin xsi:type="periodicaltime:AbsoluteTemporalExpression">
22             <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                   value="minute">
23               <rank xsi:type="periodicaltime:NumericRank" value="30"/>
24             </descriptors>
25             <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                   value="hour">
26               <rank xsi:type="periodicaltime:NumericRank" value="10"/>
27             </descriptors>
28             <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                   value="day"/>
29           </begin>
30           <end xsi:type="periodicaltime:AbsoluteTemporalExpression">
31             <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                   value="minute">
32               <rank xsi:type="periodicaltime:NumericRank" value="30"/>
33             </descriptors>
34             <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                   value="hour">
35               <rank xsi:type="periodicaltime:NumericRank" value="12"/>
36             </descriptors>
37             <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                   value="day"/>
38           </end>
39         </periodicTimeInterval>
40       </periodicRules>
```

 FIGURE F.2.3: Modèle PTOM pour l'exemple #81 du corpus Relaxnews RMM2

```
1       <periodicRules xsi:type="periodicaltime:PRuleWithImplicitFrequency">
2         <restriction>
3           <begin xsi:type="periodicaltime:AbsoluteTemporalExpression">
4             <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                  value="day">
5               <rank xsi:type="periodicaltime:NumericRank" value="1"/>
6             </descriptors>
7             <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                  value="week"/>
8           </begin>
9           <end xsi:type="periodicaltime:AbsoluteTemporalExpression">
10            <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                  value="day">
11              <rank xsi:type="periodicaltime:NumericRank" value="6"/>
12            </descriptors>
13            <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                  value="week"/>
14          </end>
15        </restriction>
16        <periodicTimeInterval>
17          <begin xsi:type="periodicaltime:AbsoluteTemporalExpression">
18            <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                  value="hour">
19              <rank xsi:type="periodicaltime:NumericRank" value="14"/>
20            </descriptors>
21            <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                  value="day"/>
22          </begin>
23          <end xsi:type="periodicaltime:AbsoluteTemporalExpression">
24            <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                  value="minute">
25              <rank xsi:type="periodicaltime:NumericRank" value="30"/>
26            </descriptors>
27            <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                  value="hour">
28              <rank xsi:type="periodicaltime:NumericRank" value="18"/>
29            </descriptors>
30            <descriptors xsi:type="periodicaltime:CalendarDescriptor"
                  value="day"/>
31          </end>
32        </periodicTimeInterval>
33      </periodicRules>
34    </periodicTemporalOccurrence>
35  </events>
36 </periodicaltime:EventSet>
```

FIGURE F.2.4: Modèle PTOM pour l'exemple #81 du corpus Relaxnews RMM2 (suite)

The event "Lundi a samedi de 10h30 a 12h30 et de 14h a 18h30." occurs periodically according to the rule(s) below
- rule : from each 30th minute of each 10th hour of each day to each 30th minute of each 12th hour of each day
using a time span as from each 1st day of each week to each 6 th day of each week

- rule : from each 14th hour of each day to each 30th minute of each 18th hour of each day
using a time span as from each 1st day of each week to each 6th day of each week

end of the event

FIGURE F.2.5: Texte en langage naturel contrôlé pour l'exemple #81 du corpus Relax-news RMM2

Annexe G

Base de données du moteur d'interrogation PTOM-S

G.1 Expression temporelle en intension

```
-- --------------------------------------------------------
-- Table structure for table `corpus`
CREATE TABLE IF NOT EXISTS `corpus` (
  `id_corpus` int(11) NOT NULL AUTO_INCREMENT,
  `label` varchar(255) COLLATE utf8_bin NOT NULL,
  PRIMARY KEY (`id_corpus`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

-- --------------------------------------------------------
-- Table structure for table `event`
CREATE TABLE IF NOT EXISTS `event` (
  `id_event` int(11) NOT NULL,
  `label` longtext COLLATE utf8_bin NOT NULL,
  `id_corpus` int(11) NOT NULL,
  PRIMARY KEY (`id_event`),
  KEY `id_corpus` (`id_corpus`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

-- --------------------------------------------------------
-- Table structure for table `eventcto`
CREATE TABLE IF NOT EXISTS `eventcto` (
  `id_eventptom` int(11) NOT NULL,
  `start` datetime NOT NULL,
  `end` datetime DEFAULT NULL,
  KEY `id_eventptom` (`id_eventptom`),
  KEY `start` (`start`),
  KEY `end` (`end`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

-- --------------------------------------------------------
-- Table structure for table `eventptom`
CREATE TABLE IF NOT EXISTS `eventptom` (
  `id_eventptom` int(11) NOT NULL AUTO_INCREMENT,
  `label` varchar(255) COLLATE utf8_bin NOT NULL,
  PRIMARY KEY (`id_eventptom`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

-- --------------------------------------------------------
-- Table structure for table `paggregate`
CREATE TABLE IF NOT EXISTS `paggregate` (
  `id_prule` int(11) NOT NULL,
  `id_pinterval` int(11) DEFAULT NULL,
  `id_pinstant` int(11) DEFAULT NULL,
  `deep` int(11) NOT NULL,
  KEY `id_prule` (`id_prule`),
  KEY `id_pinterval` (`id_pinterval`),
  KEY `id_pinstant` (`id_pinstant`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

FIGURE G.1.1: Schéma de la base de données « *intension* » : relations

```
-- ----------------------------------------------------------
-- Table structure for table `pinstant`
CREATE TABLE IF NOT EXISTS `pinstant` (
  `id_pinstant` int(11) NOT NULL AUTO_INCREMENT,
  `label` longtext COLLATE utf8_bin NOT NULL,
  `second` int(11) DEFAULT NULL,
  `minute` int(11) DEFAULT NULL,
  `hour` int(11) DEFAULT NULL,
  `day` int(11) DEFAULT NULL,
  `rankday` int(11) DEFAULT NULL,
  `week` int(11) DEFAULT NULL,
  `month` int(11) DEFAULT NULL,
  `rankmonth` int(11) DEFAULT NULL,
  `year` int(11) DEFAULT NULL,
  `century` int(11) DEFAULT NULL,
  `code` int(11) NOT NULL,
  PRIMARY KEY (`id_pinstant`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

-- ----------------------------------------------------------
-- Table structure for table `pinterval`
CREATE TABLE IF NOT EXISTS `pinterval` (
  `id_pinterval` int(11) NOT NULL AUTO_INCREMENT,
  `label` longtext COLLATE utf8_bin NOT NULL,
  `start_pinstant` int(11) NOT NULL,
  `end_pinstant` int(11) NOT NULL,
  PRIMARY KEY (`id_pinterval`),
  KEY `start_pinstant` (`start_pinstant`),
  KEY `end_pinstant` (`end_pinstant`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

-- ----------------------------------------------------------
-- Table structure for table `prule`
CREATE TABLE IF NOT EXISTS `prule` (
  `id_prule` int(11) NOT NULL AUTO_INCREMENT,
  `id_event` int(11) NOT NULL,
  `extentstart` date DEFAULT NULL,
  `extentend` date DEFAULT NULL,
  PRIMARY KEY (`id_prule`),
  KEY `id_event` (`id_event`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

FIGURE G.1.2: Schéma de la base de données « *intension* » : relations (suite)

```
-- --------------------------------------------------------

-- Structure for view `canonical_rule_pi`
CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER
    VIEW `canonical_rule_pi` AS (select distinct `a`.`id_prule` AS `
    id_prule`,sum(`start`.`second`) AS `start_second`,sum(`start`.`minute`)
     AS `start_minute`,sum(`start`.`hour`) AS `start_hour`,sum(`start`.`day
    `) AS `start_day`,sum(`start`.`rankday`) AS `start_rankday`,sum(`start
    `.`week`) AS `start_week`,sum(`start`.`month`) AS `start_month`,sum(`
    start`.`rankmonth`) AS `start_rankmonth`,sum(`start`.`year`) AS `
    start_year`,sum(`start`.`century`) AS `start_century`,sum(`start`.`code
    `) AS `start_code`,sum(`end`.`second`) AS `end_second`,sum(`end`.`
    minute`) AS `end_minute`,sum(`end`.`hour`) AS `end_hour`,sum(`end`.`day
    `) AS `end_day`,sum(`end`.`rankday`) AS `end_rankday`,sum(`end`.`week`)
     AS `end_week`,sum(`end`.`month`) AS `end_month`,sum(`end`.`rankmonth`)
     AS `end_rankmonth`,sum(`end`.`year`) AS `end_year`,sum(`end`.`century
    `) AS `end_century`,sum(`end`.`code`) AS `end_code` from (((`paggregate
    ` `a` join `pinterval` `pi`) join `pinstant` `start`) join `pinstant` `
    end`) where ((`pi`.`id_pinterval` = `a`.`id_pinterval`) and isnull(`a
    `.`id_pinstant`) and (`start`.`id_pinstant` = `pi`.`start_pinstant`)
    and (`end`.`id_pinstant` = `pi`.`end_pinstant`)) group by `a`.`id_prule
    `) union (select distinct `a`.`id_prule` AS `id_prule`,`pist`.`second`
    AS `start_second`,`pist`.`minute` AS `start_minute`,`pist`.`hour` AS `
    start_hour`,`pist`.`day` AS `start_day`,`pist`.`rankday` AS `
    start_rankday`,`pist`.`week` AS `start_week`,`pist`.`month` AS `
    start_month`,`pist`.`rankmonth` AS `start_rankmonth`,`pist`.`year` AS `
    start_year`,`pist`.`century` AS `start_century`,`pist`.`code` AS `
    start_code`,`pist`.`second` AS `end_second`,`pist`.`minute` AS `
    end_minute`,`pist`.`hour` AS `end_hour`,`pist`.`day` AS `end_day`,`pist
    `.`rankday` AS `end_rankday`,`pist`.`week` AS `end_week`,`pist`.`month`
     AS `end_month`,`pist`.`rankmonth` AS `end_rankmonth`,`pist`.`year` AS
    `end_year`,`pist`.`century` AS `end_century`,`pist`.`code` AS `end_code
    ` from (`paggregate` `a` join `pinstant` `pist`) where ((`pist`.`
    id_pinstant` = `a`.`id_pinstant`) and isnull(`a`.`id_pinterval`)));
```

FIGURE G.1.3: Schéma de la base de données « *intension* » : instants périodiques sous forme de vue

```
-- ----------------------------------------------------------------------
-- Structure for view `canonical_rule`
CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER
    VIEW `canonical_rule` AS select distinct `canonical_rule_pi`.`id_prule
    ` AS `id_prule`,sum(`canonical_rule_pi`.`start_second`) AS `
    start_second`,sum(`canonical_rule_pi`.`start_minute`) AS `start_minute
    `,sum(`canonical_rule_pi`.`start_hour`) AS `start_hour`,sum(`
    canonical_rule_pi`.`start_day`) AS `start_day`,sum(`canonical_rule_pi
    `.`start_rankday`) AS `start_rankday`,sum(`canonical_rule_pi`.`
    start_week`) AS `start_week`,sum(`canonical_rule_pi`.`start_month`) AS
    `start_month`,sum(`canonical_rule_pi`.`start_rankmonth`) AS `
    start_rankmonth`,sum(`canonical_rule_pi`.`start_year`) AS `start_year`,
    sum(`canonical_rule_pi`.`start_century`) AS `start_century`,sum(`
    canonical_rule_pi`.`start_code`) AS `start_code`,sum(`canonical_rule_pi
    `.`end_second`) AS `end_second`,sum(`canonical_rule_pi`.`end_minute`)
    AS `end_minute`,sum(`canonical_rule_pi`.`end_hour`) AS `end_hour`,sum(`
    canonical_rule_pi`.`end_day`) AS `end_day`,sum(`canonical_rule_pi`.`
    end_rankday`) AS `end_rankday`,sum(`canonical_rule_pi`.`end_week`) AS `
    end_week`,sum(`canonical_rule_pi`.`end_month`) AS `end_month`,sum(`
    canonical_rule_pi`.`end_rankmonth`) AS `end_rankmonth`,sum(`
    canonical_rule_pi`.`end_year`) AS `end_year`,sum(`canonical_rule_pi`.`
    end_century`) AS `end_century`,sum(`canonical_rule_pi`.`end_code`) AS `
    end_code` from `canonical_rule_pi` group by `canonical_rule_pi`.`
    id_prule`;
```

FIGURE G.1.4: Schéma de la base de données « *intension* » : intervalles périodiques des règles de périodicité sous forme de vue (suite)

```
-- ----------------------------------------------------------------------
-- Constraints for table `event`
ALTER TABLE `event`
  ADD CONSTRAINT `event_ibfk_1` FOREIGN KEY (`id_corpus`) REFERENCES `
      corpus` (`id_corpus`) ON DELETE CASCADE ON UPDATE CASCADE;
--
-- Constraints for table `paggregate`
ALTER TABLE `paggregate`
  ADD CONSTRAINT `paggregate_ibfk_1` FOREIGN KEY (`id_prule`) REFERENCES `
      prule` (`id_prule`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `paggregate_ibfk_2` FOREIGN KEY (`id_pinterval`)
      REFERENCES `pinterval` (`id_pinterval`) ON DELETE CASCADE ON UPDATE
      CASCADE,
  ADD CONSTRAINT `paggregate_ibfk_3` FOREIGN KEY (`id_pinstant`)
      REFERENCES `pinstant` (`id_pinstant`) ON DELETE CASCADE ON UPDATE
      CASCADE;
--
-- Constraints for table `pinterval`
ALTER TABLE `pinterval`
  ADD CONSTRAINT `pinterval_ibfk_1` FOREIGN KEY (`start_pinstant`)
      REFERENCES `pinstant` (`id_pinstant`) ON DELETE CASCADE ON UPDATE
      CASCADE,
  ADD CONSTRAINT `pinterval_ibfk_2` FOREIGN KEY (`end_pinstant`)
      REFERENCES `pinstant` (`id_pinstant`) ON DELETE CASCADE ON UPDATE
      CASCADE;
--
-- Constraints for table `prule`
ALTER TABLE `prule`
  ADD CONSTRAINT `prule_ibfk_1` FOREIGN KEY (`id_event`) REFERENCES `event
      ` (`id_event`) ON DELETE CASCADE ON UPDATE CASCADE;
```

FIGURE G.1.5: Schéma de la base de données « *intension* » : contraintes

G.2 Expression temporelle en extension

```
-- Database: `ptomrmm2`
-- --------------------------------------------------------
-- Table structure for table `eventcto`
CREATE TABLE IF NOT EXISTS `eventcto` (
  `id_eventptom` int(11) NOT NULL,
  `start` datetime NOT NULL,
  `end` datetime DEFAULT NULL,
  KEY `id_eventptom` (`id_eventptom`),
  KEY `start` (`start`),
  KEY `end` (`end`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

-- Constraints for table `eventcto`
ALTER TABLE `eventcto`
  ADD CONSTRAINT `eventcto_ibfk_1` FOREIGN KEY (`id_eventptom`) REFERENCES
      `eventptom` (`id_eventptom`) ON DELETE CASCADE ON UPDATE CASCADE;
--
```

FIGURE G.2.1: Schéma de la base de données : expression temporelle en « *extension* »

Annexe H

MODSEA

H.1 Déclaration des SWidgets à générer puis à composer avec Widgets

Afin de piloter la génération des `SWidget` (cf. section 6.5.5.3) nous utilisons un fichier de configuration. Un `SWidget` est associé à un type de composant graphique. Dans le métamodèle `Widgets`, ces types sont les classes abstraites qui spécialisent la classe `Widget` comme `AbstractTimeline`. La configuration consiste à désigner qu'elle est la classe abstraite étendue par le composant graphique.

La figure H.1.1 donne le contenu de ce fichier pour le cas de Simile Exhibit. Pour chaque composant graphique (i.e. chaque ligne) est indiqué : le « nom_du_composant » qui doit hériter d'une classe abstraite (« classe_abstraite_à_étendre », e.g. `AbstractTimeline`). Une classe abstraite est ajoutée dans chaque `SWidget`, c'est elle qui sera reconnue par le moteur de composition comme ayant une signature similaire lors de l'intégration de chaque `SWidget` dans le métamodèle `Widgets`.

Ce fichier est également utilisé lors de la composition comme le fournisseur de la

liste des `SWidget` à intégrer dans `Widgets`. Le développeur n'a pas à définir de directive de composition supplémentaire, seul le nom de la classe abstraite est nécessaire pour exécuter le processus.

```
#---------#
# Légende #
#[nom_du_composant] = [nouveau_nom_du_composant_si_nécessaire]->
[classe_abstraite_à_étendre]
#---------#
#------#
# MAPS #
#------#
MapView = MapView->AbstractMap
OLMapView = OLMapView->AbstractMap
#----------#
# Timeline #
#----------#
TimelineView = TimelineView->AbstractTimeline
#-------------#
# OTHER VIEWS #
#-------------#
OrderedViewFrame = OrderedViewFrame->View
CalendarView = CalendarView->View
TileView = TileView->OrderedViewFrame
ThumbnailView = ThumbnailView->OrderedViewFrame
TabularView = TabularView|TabularView->OrderedViewFrame
TimeplotView = TimeplotView->View
HTMLView = HTMLView->View
#--------#
# FACETS #
#--------#
TextSearchFacet = TextSearchFacet->Facet
DatePickerFacet = DatePickerFacet->Facet
ListFacet = ListFacet->Facet
ImageFacet = ImageFacet->Facet
```

FIGURE H.1.1: Fichier de déclaration des `SWidgets`

H.2 Métamodèles

H.2.1 Métamodèle de positionnement des composants graphiques : « Layout »

`Layout` est un métamodèle dédié au positionnement des composants graphiques dans l'application, il permet de séparer la configuration des composants de leur position graphique dans l'application. Ainsi la réutilisation des deux modèles indépendamment de

l'un et de l'autre est possible. Le métamodèle proposé (cf. figure H.2.1) est inspiré des tableaux HTML, on y retrouve les notions de `Grid`, `Row` et `Cell`. Une `Grid` est composée de `Row`, elles-mêmes composées de `Cell`. Une `Cell` est soit constituée de `Widget` (référence `widgets`) définit dans un modèle de `Widgets` ou bien d'une `Grid` (référence `grid`). Il est possible de créer des compositions graphiques riches pour positionner les différents composants. Un exemple d'instanciation est fourni en figure H.3.2.

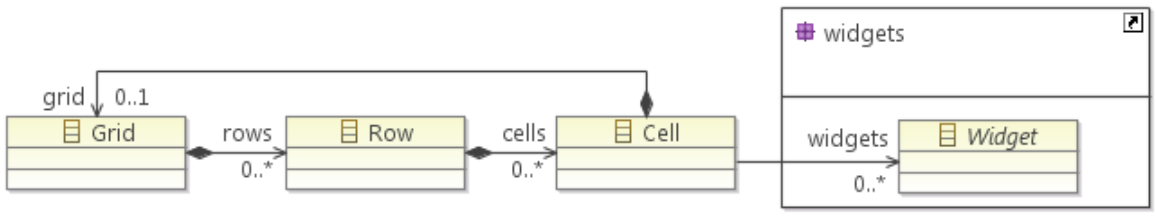


FIGURE H.2.1: Métamodèle de `Layout`

H.2.2 Métamodèles de correspondance

Dans cette section nous présentons les métamodèles de correspondance qui mettent en relation des différents modèles dans MODSEA. Le premier métamodèle concerne la liaison entre les composants et les données, il est nommé `Widget_Content`. Les liens entre les composants et les définitions de disposition sont stockés comme instances du métamodèle `Widget_Layout`. L'apport de métamodèles dédiés est de séparer la définition des composants de leur disposition et des sources de données afin de pouvoir réutiliser indépendamment ces modèles et ainsi favoriser la constitution de famille d'applications Simile Exhibit.

Métamodèle de correspondance « Widget_Content »

Un `DataBinding` est un ensemble de mises en correspondance : `WidgetBinding`. Un `WidgetBinding` est associé à un `Widget` et possède des `WidgetContentBindings` qui relie une propriété de `Widget` (`WidgetFeature`) avec une propriété (`ContentFeature`) du métamodèle `Content` lui servant de source de données.

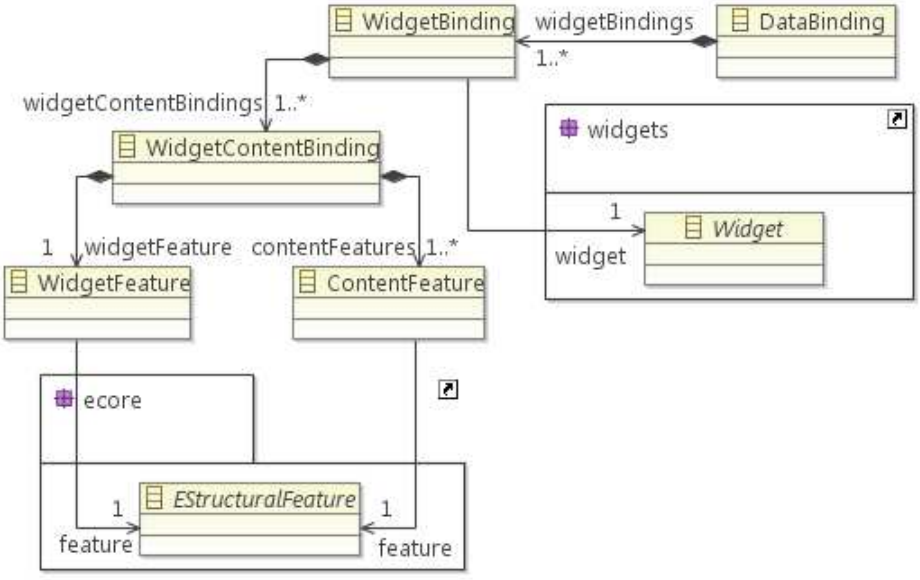


FIGURE H.2.2: Métamodèle de correspondance `Widget_Content`

Pour faciliter l'édition des instances de ce métamodèle, dans MODSEA nous proposons à l'utilisateur de générer un modèle de correspondance de données par défaut. Cette génération est effectuée grâce à l'instance du métamodèle `Widgets` qui est utilisée pour configurer l'application web. Pour chaque composant graphique utilisé, il est possible d'inspecter le `SWidget` correspondant pour connaître les propriétés nécessitant une correspondance. Ainsi l'ensemble des `WidgetFeature` à configurer sont générés, il suffit à l'utilisateur de choisir les `ContentFeature` correspondants. Un exemple d'instanciation est fourni en figure H.3.3.

Métamodèle de correspondance « Widget_Layout »

Un `LayoutBinding` est un ensemble de correspondances : `CellWidgetBinding` entre une `Cell`(-ule) et des `Widget`. Lors de l'édition d'une instance de ce métamodèle, l'utilisateur choisit où doit être positionné un composant graphique (configuré dans le modèle de `Widgets`) en utilisant une `Cell`(-ule) provenant du modèle de `Layout`. Un exemple d'instanciation est fourni en figure H.3.4.

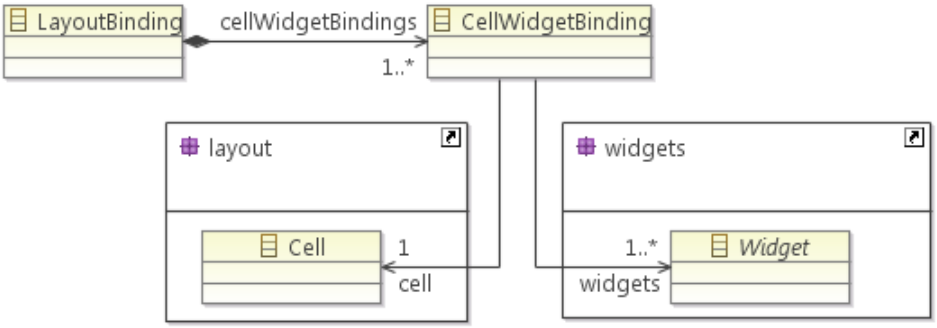


FIGURE H.2.3: Métamodèle de correspondance `Widget_Layout`

H.3 Ressources pour la configuration d'une application Simile Exhibit avec MODSEA

Cet annexe fournit des exemples de ressources nécessaires à la configuration d'une application Simile Exhibit avec MODSEA qui a été créée pour le projet RMM2. Nous proposons ainsi un modèle de `Widgets` (figure H.3.1) qui contient les configurations des composants graphiques. La figure H.3.2 présente un exemple d'instance du métamodèle de `Layout`. Les figures H.3.3 et H.3.4 proposent des modèles de correspondance entre `Content` et `Widgets`, et `Layout` et `Widgets`. Le programme qui génère l'application prend en paramètre le fichier donné en figure H.3.5, il recense quelques informations dont les chemins vers les deux modèles de correspondances à utiliser. Les chemins vers les autres ressources étant résolus par les dépendances structurelles entre les différents modèles.

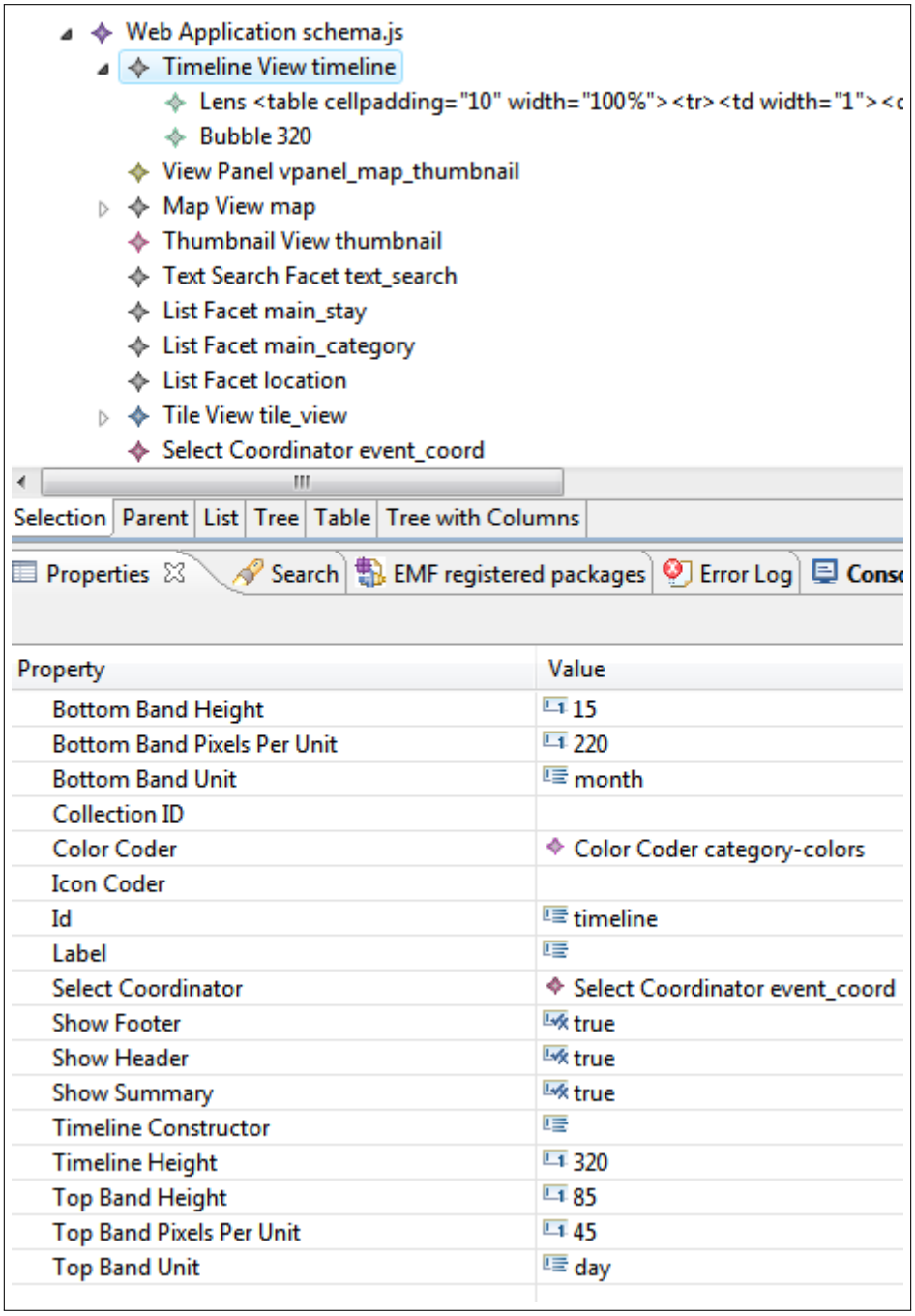


FIGURE H.3.1: Modèle de `Widgets`

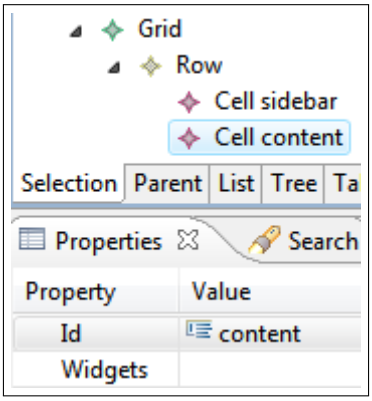


FIGURE H.3.2: Modèle de positionnement de `Layout`

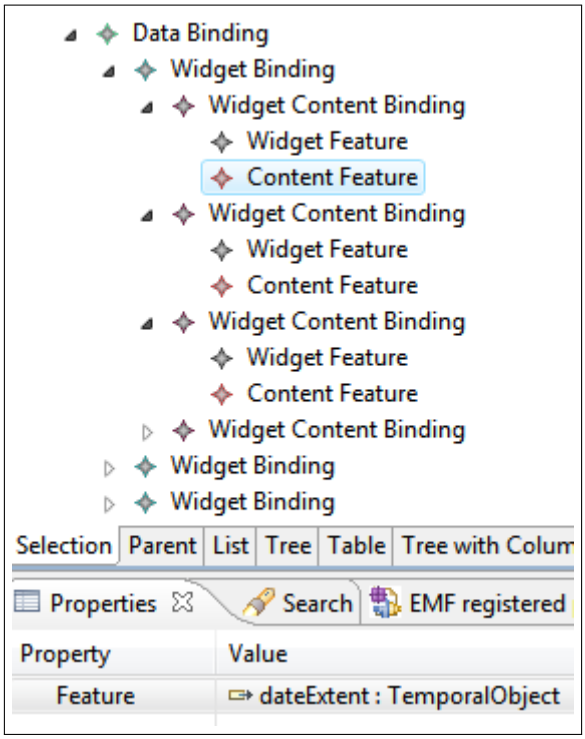


FIGURE H.3.3: Modèle de correspondance `Widget_Content` entre le métamodèle de `Content` et un modèle de `Widgets`

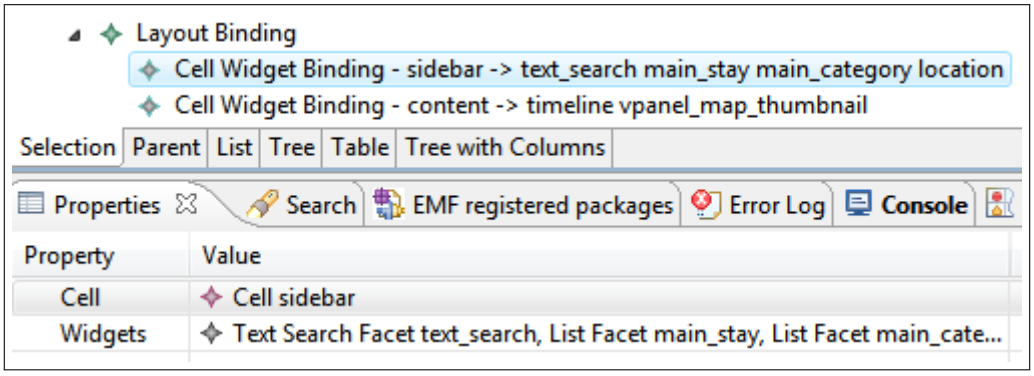


FIGURE H.3.4: Modèle de correspondance `Widget_Layout` entre un modèle de `Layout` et un modèle de `Widgets`

```
webapp_type = modsea
#-----------#
# Variables #
#-----------#
$input_path = "platform:/resource/l3i.event.modsea/model/modsea/"
$output_path = "platform:/resource/l3i.event.modsea/model/output/"
#---------#
# Chemins #
#---------#
widget_content = $input_path + "variants/relax/RMM2App.databinding"
widget_layout = $input_path + "variants/relax/RMM2App.layoutbinding"
webapp = $output_path + "RMM2App_relax.html"
```

FIGURE H.3.5: Fichier de configuration contenant les chemins vers les ressources à utiliser pour la génération de l'application Simile Exhibit

H.4 Directives de filtrage sur des propriétés pour transformer des modèles EMF en JSON

Les directives (sous forme d'annotations) sont spécifiées sur des classes ou des propriétés de classe d'un métamodèle pour paramétrer la génération d'un flux JSON à partir de modèles EMF (cf. section 6.5.6).

1. `isItemType` : désigne le type de données racine (`items`, figure H.4.2 - l.1), qui dans le cas du métamodèle `Content` utilisé, est la classe `Event` (figure H.4.1 - l.2).
2. `excluded` : permet d'exclure tous les attributs et propriétés qui ne sont pas à générer en JSON. Dans l'exemple ci-dessous cela permet de retirer une référence qui est utilisée pour contenir des catégories (`categories`, figure H.4.1 - l.4) alors que ces mêmes catégories seront déjà générées *via* des références ayant une sémantique

plus forte comme `mainStay` ou bien `mainCategory`. Par conséquent dans le JSON généré, `categories` est vide (`[]`, figure H.4.2 - l.8).

3. `isLabel` : duplique un attribut en lui donnant comme nom `label`. Cette annotation est utile notamment pour des composants affichant des données nécessitant un attribut nommé `label` comme Simile Exhibit, voir figure H.4.1 - l.6 et figure H.4.2 - ls. 2 et 3. L'attribut est dupliqué car l'original peut être utilisé par ailleurs.
4. `valuesInContainer.label` : déplace la valeur de l'attribut label dans le conteneur de l'objet concerné. Cela est utile pour limiter le nombre d'objets générés. En effet, `TemporalObject` (figure H.4.2 l.13) et `SpatialObject` (figure H.4.2 l.17) sont générés dans des objets indépendants. Selon les applications utilisant le JSON produit, il peut être pertinent de ne pas générer ces objets de manière séparée. Par exemple, les `label`s des catégories sont stockés directement dans un objet `Event` au lieu d'un objet `Category`. `Event` représente un conteneur de `Category` donc c'est dans ce type d'objet que sont déplacées les valeurs. Pour réaliser cela, nous avons défini l'annotation `valuesInContainer`. La seconde partie de l'annotation « `.label` » désigne les valeurs de l'attribut a déplacé. Par exemple : `mainStay` (figure H.4.1 - l.8 et figure H.4.2 - l.6), `mainCategory` (figure H.4.1 - l.10 et figure H.4.2 - l.7) et `secondaryCategoriesAndSubjects` (figure H.4.1 - l.12).

```
1  /** EMF2JSON Directives */
2  @EMF2json "isItemType"
3  aspect class Event {
4    @EMF2json "excluded"
5    attribute categories : Category [0..*]
6    @EMF2json "isLabel"
7    attribute title : String [1..1]
8    @EMF2json "valuesInContainer.label"
9    reference mainStay : Category [0..1]
10   @EMF2json "valuesInContainer.label"
11   reference mainCategory : Category [0..1]
12   @EMF2json "valuesInContainer.label"
13   reference secondaryCategoriesAndSubjects : Category [0..*]
14 }
```

FIGURE H.4.1: Annotations en KERMETA pour personnaliser la génération du JSON avec EMF2JSON

```
1  {items: [
2    { label: 'Francofolies de La Rochelle',
3      title: 'Francofolies de La Rochelle',
4      type: 'Event',
5      dateExtent: 'TemporalObject3', location: 'SpatialObject3',
6      mainStay: 'Entertainment',
7      mainCategory: 'Music',
8      categories: []
9    }, {label: 'TemporalObject3',
10     type: 'TemporalObject',
11     begin: '2011-07-12', end: '2011-07-16'
12   }, {label : 'SpatialObject3',
13     type: 'SpatialObject',
14       name: 'La Rochelle, France ', latlng_coordinates:
             '46.15701,-1.15926'
15 }]}
```

FIGURE H.4.2: Extrait de données JSON générées par l'application générique EMF2JSON

Bibliographie

[Allen 81] James F. Allen. *An Interval-Based Representation of Temporal Knowledge*. In Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI'81), pages 221–226, Vancouver, BC, Canada, 1981. *Cité page 91.*

[Allen 83] James F. Allen. *Maintaining knowledge about temporal intervals*. Communications of the ACM, vol. 26, no. 11, pages 832–843, 1983. www *3 citations pages 15, 91 et 204.*

[André 07] Charles André, Frédéric Mallet & Robert de Simone. *Modeling Time(s)*. In Gregor Engels, Bill Opdyke, Douglas Schmidt & Frank Weil, editeurs, Model Driven Engineering Languages and Systems, volume 4735 of *Lecture Notes in Computer Science*, pages 559–573. Springer Berlin / Heidelberg, 2007. www *Cité page 46.*

[André 08] Charles André & Frédéric Mallet. *Clock Constraints in UML/MARTE CCSL*. Rapport de recherche RR-6540, INRIA, 2008. www *Cité page 47.*

[André 10] Charles André, Julien DeAntoni, Frédéric Mallet & Robert Simone. *The Time Model of Logical Clocks Available in the OMG MARTE Profile*. In Sandeep K Shukla & Jean-Pierre Talpin, editeurs, Synthesis of Embedded Software, pages 201–227. Springer US, 2010. www *Cité page 46.*

[Andrienko 03] Natalia V. Andrienko, Gennady L. Andrienko & Peter Gatalsky. *Exploratory spatio-temporal visualization : an analytical review*. Journal of Visual Languages & Computing, vol. 14, no. 6, pages 503–541, 2003. www *Cité page 195.*

[Anselma 09] Luca Anselma, Stefania Montani & Paolo Terenziani. *An intensional approach to qualitative and quantitative periodicity-dependent temporal constraint.* International Journal of Intelligent Systems, vol. 24, no. 8, pages 902–918, 2009. www *Cité page 46.*

[Apel 07] Sven Apel, Christian Lengauer, Don Batory, Bernhard Möller & Christian Kästner. *An algebra for feature-oriented software development - Technical Report, Number MIP-0706.* Rapport technique, University of Passau, 2007. www *Cité page 59.*

[Atkinson 03] Colin Atkinson & Thomas Kühne. *Model-Driven Development : A Metamodeling Foundation.* IEEE Software, vol. 20, no. 5, pages 36–41, 2003. *Cité page 55.*

[Authosserre-Cavarero 12] Annie Authosserre-Cavarero, Frederic Bertrand, Mireille Blay- Fornarino, Philippe Collet, Hubert Dubois, Stéphane Ducasse, Sophie Dupuy-Chessa, Catherine Faron-Zucker, Cyril Faucher, Jean-Yves Lafaye, Philippe Lahire, Olivier Le Goaer, Johan Montagnat & Anne-Marie Pinna-Dery. *Interopérabilité des systèmes d'information : approches dirigées par les modèles.* In Inforsid, pages 11–30, Montpellier, France, 2012. www *Cité page 7.*

[Battistelli 08] Delphine Battistelli, Javier Couto, Jean-Luc Minel & Sylviane R Schwer. *Representing and Visualizing Calendar Expressions in Texts.* In Johan Bos & Rodolfo Delmonte, editeurs, Semantics in Text Processing. STEP 2008 Conference Proceedings, volume 1 of *Research in Computational Semantics*, pages 365–373, Venice, Italy, 2008. College Publications. www *Cité page 200.*

[Battistelli 11] Delphine Battistelli. Linguistique et recherche d'information : la problématique du temps. Traitement de l'information. Hermes Science Lavoisier, 2011. *4 citations pages 6, 36, 63 et 200.*

[Becher 06] Gérard Becher, Patrice Enjalbert, Estelle Fievé, Laurent Gosselin, François Lévy & Gérard Ligozat. *Rapport technique du projet OGRE.* CoRR, vol. abs/cs/061, 2006. *6 citations pages 13, 32, 35, 36, 63 et 159.*

[Bettini 04] Claudio Bettini, Sergio Mascetti & Xiaoyang Sean Wang. *Mapping Calendar Expressions into Periodical Granularities.* In TIME, pages 96–102, 2004. *Cité page 44.*

[Bézivin 03] Jean Bézivin, Grégoire Dupé, Frédéric Jouault, G Pitette & J Rougui. *First Experiments with the ATL Mo-*

del Transformation Language : Transforming XSLT into XQuery. In OOPSLA 2003 Workshop, Anaheim, USA, 2003. *Cité page 57.*

[Bézivin 05] Jean Bézivin. *On The Unification Power of Models*. Software and System Modelling, vol. 4, no. 2, pages 171–188, 2005. *Cité page 55.*

[Bittar 11] André Bittar, Pascal Amsili, Pascal Denis & Laurence Danlos. *French TimeBank : An ISO-TimeML Annotated Reference Corpus*. In ACL 2011 (Short Papers) - 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies, pages 130–134, 2011. www *Cité page 39.*

[Blay-Fornarino 09] Mireille Blay-Fornarino. *Interprétations de la composition d'activités*. Habilitation à diriger des recherches, Université de Nice - Sophia Antipolis, 2009. www *Cité page 59.*

[Brambilla 08] M Brambilla, S Comai, P Fraternali & M Matera. *Designing Web Applications with WebML and WebRatio*. Human-Computer Interaction Series, vol. 12, pages 221–261, 2008. *Cité page 169.*

[Cabot 03] Jordi Cabot, Antoni Olivé & Ernest Teniente. *Representing Temporal Information in UML*. In UML'03, Lecture Notes in Computer Science, pages 44–59. Springer Berlin / Heidelberg, 2003. *Cité page 24.*

[Carlson 99] Andy Carlson, Sharon Estepp & Martin Fowler. *Temporal Patterns*. In Harrison Foote, editeur, Pattern Languages of Program Design 4, pages 241–262. Addison-Wesley, 1999. *Cité page 24.*

[Carnap 47] Rudolf Carnap. Meaning and Necessity. University of Chicago Press, 1947. *4 citations pages 4, 5, 32 et 73.*

[Caspi 87] Paul Caspi, Daniel Pilaud, Nicolas Halbwachs & John A. Plaice. *LUSTRE : a declarative language for real-time programming*. In POPL '87 Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, pages 178–188. ACM New York, 1987. www *Cité page 33.*

[Chen 11] Lianping Chen & Muhammad Ali Babar. *A systematic review of evaluation of variability management approaches in software product lines*. Information & Software Technology, vol. 53, no. 4, pages 344–362, 2011. *Cité page 170.*

[Chittaro 02] Luca Chittaro & Carlo Combi. *Temporal Granularity and Indeterminacy in Reasoning About Actions and Change : An Approach Based on the Event Calculus*. Ann. Math. Artif. Intell., vol. 36, no. 1-2, pages 81–119, 2002. *Cité page 44.*

[Clarke 01] Siobhán Clarke & Robert J Walker. *Composition Patterns : An Approach to Designing Reusable Aspects*. In ICSE'01, 23rd International Conference on Software Engineering, pages 5–14, Washington, DC, USA, 2001. IEEE Computer Society. *Cité page 169.*

[Clements 06] Paul C Clements. *Managing Variability for Software Product Lines : Working with Variability Mechanisms*. In SPLC, pages 207–208, 2006. *Cité page 170.*

[Condotta 06] Jean-François Condotta, Mahmoud Saade & Gérard Ligozat. *A Generic Toolkit for n-ary Qualitative Temporal and Spatial Calculi*. In 13th International Symposium on Temporal Representation and Reasoning (TIME 2006), pages 78–86, 2006. *Cité page 46.*

[Consel 98] Charles Consel & Renaud Marlet. *Architecturing Software Using A Methodology for Language Development*. In Proceedings of the 10 th International Symposium on Programming Language Implementation and Logic Programming, number 1490 in Lecture Notes in Computer Science, pages 170–194, 1998. *Cité page 57.*

[Consortium 10] Glocal Consortium. *Glocal, Event-based Retrieval of networked Media, Delivrable D1.2 : Initial set of multimedia content and event models*. Rapport technique, 2010. *3 citations pages 28, 29 et 63.*

[Crockford 06] D. Crockford. *The application/json Media Type for JavaScript Object Notation (JSON)*, 2006. `http://tools.ietf.org/html/rfc4627` *Cité page 180.*

[Dawson 98] F Dawson & D Stenerson. *Internet Calendaring and Scheduling Core Object Specification (iCalendar) - RFC2445*, 1998. `http://www.ietf.org/rfc/rfc2445.txt` *Cité page 42.*

[Deantoni 12a] Julien Deantoni & Frédéric Mallet. *ECL : the Event Constraint Language, an Extension of OCL with Events*. Rapport de recherche RR-8031, INRIA, 2012. WWW *Cité page 47.*

[DeAntoni 12b] Julien DeAntoni & Frédéric Mallet. *TimeSquare : Treat Your Models with Logical Time*. In Objects, Models, Com-

ponents, Patterns - 50th International Conference, TOOLS 2012, pages 34–41, 2012. *Cité page 46.*

[Dechter 91] Rina Dechter, Itay Meiri & Judea Pearl. *Temporal Constraint Networks.* Artif. Intell., vol. 49, no. 1-3, pages 61–95, 1991. *Cité page 82.*

[Desruisseaux 09] B. Desruisseaux. *Internet Calendaring and Scheduling Core Object Specification (iCalendar) - RFC5545*, 2009. *4 citations pages 7, 41, 42 et 43.*

[Drakengren 96] Thomas Drakengren. *Uniqueness of Scott's Reflexive Domain in P-omega.* Theor. Comput. Sci., vol. 155, no. 1, pages 267–276, 1996. *Cité page 90.*

[Dubois 00] Michel Dubois & Sylviane Schwer R. *Classification topologique des ensembles convexes de Allen.* In proceeding of the 12th Congr{è}s Francophone AFRIF-AFIA., volume R.F.I.A. 2, pages 59–68, Paris, France, 2000. www *Cité page 99.*

[Eclipse 12] Eclipse. *EMF*, 2012. `http://www.eclipse.org/modeling/emf/` *Cité page 169.*

[Edsall 05] R.M. Edsall & L.R. Sidney. *Applications of a cognitively informed framework for the design of interactive spatio-temporal representations.* In J. Dykes, Alan M. MacEachren & M. J. Kraak, editeurs, Exploring geovisualization, International Cartographic Association, page 730. Elsevier Science, Amsterdam, The Netherlands, 2005. www *Cité page 194.*

[Eysholdt 10] Moritz Eysholdt & Heiko Behrens. *Xtext : implement your language faster than the quick and dirty way.* In Companion to the 25th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, SPLASH/OOPSLA 2010, pages 307–309, Reno/Tahoe, Nevada, 2010. ACM. *Cité page 58.*

[Faucher 08] Cyril Faucher, Frédéric Bertrand & Jean-Yves Lafaye. *Génération d'ontologie à partir d'un modèle métier UML annoté.* Revue des Nouvelles Technologies de l'Information (RNTI), vol. E, no. 12, pages 65–84, 2008. www *Cité page 4.*

[Faucher 09a] Cyril Faucher & Mickael Clavreul. *KET reference manual.* Triskell group - IRISA/INRIA, 2009. www *Cité page 60.*

[Faucher 09b] Cyril Faucher, Françoise Gourmelon, Jean-Yves Lafaye & Mathias Rouan. *Mise en oeuvre d'une mémoire environnementale adaptée aux besoins d'un observatoire du domaine*

côtier : MenIr. Revue Internationale de Géomatique (RIG), vol. 19, no. 1, pages 7–26, 2009. www *Cité page 24.*

[Faucher 10a] Cyril Faucher, Frédéric Bertrand & Jean-Yves Lafaye. *Génération d'un métamodèle de composants graphiques à partir de la spécification d'une bibliothèque de composants Web.* In Atelier IDM-IHM, Pau, France, 2010. www *2 citations pages 202 et 222.*

[Faucher 10b] Cyril Faucher, Jean-Yves Lafaye, Frédéric Bertrand & Charles Teissèdre. *Modélisation et reformulation d'expressions temporelles extraites de textes en langage naturel.* In AFADL 2010 (10es Journées Francophones Internationales sur les Approches Formelles dans l'Assistance au Développement de Logiciels), pages 213–216, Futuroscope-Poitiers, France, 2010. www *Cité page 220.*

[Faucher 10c] Cyril Faucher, Charles Teissèdre, Jean-Yves Lafaye & Frédéric Bertrand. *Temporal Knowledge Acquisition and Modeling.* In EKAW 2010 - Knowledge Engineering and Knowledge Management by the Masses, volume 6317 of *LNCS (LNAI)*, pages 371–380, Lisbon, Portugal, 2010. Springer-Verlag. www *5 citations pages 12, 200, 202, 206 et 221.*

[Faucher 10d] Cyril Faucher, Cyril Tissot, Jean-Yves Lafaye & Frédéric Bertrand. *Benefits of a periodic temporal model for the simulation of human activities.* In GeoVA(t) (Geospatial Visual Analytics : Focus on Time), Guimaraes, Portugal, 2010. *Cité page 226.*

[Faucher 10e] Cyril Faucher, Cyril Tissot, Jean-Yves Lafaye, Frédéric Bertrand, David Brosset & Mathias Rouan. *Location- and temporal-based services for nature-society interaction regulation.* Journal of Location Based Services (JLBS), vol. 4, no. 3-4, pages 147–165, 2010. www *4 citations pages 14, 194, 199 et 226.*

[Faucher 11] Cyril Faucher, Samnang Chea, Frédéric Bertrand & Jean-Yves Lafaye. *Validation sémantique d'objets à l'aide d'un modèle de référence et de contraintes.* In IDM 2011, pages 109–113, Lille, France, 2011. www *4 citations pages 146, 204, 221 et 222.*

[Faucher 12a] Cyril Faucher, Frédéric Bertrand, Jean-Yves Lafaye, Denis Teyssou & Mathieu Bully. *Une approche fondée sur l'IDM pour le développement d'un environnement de production journalistique.* TSI, numéro spécial, IDM. Vers

l'usage industriel, vol. 31, no. 7, pages 917–942, 2012. *4 citations pages 3, 73, 220 et 222.*

[Faucher 12b] Cyril Faucher, Jean-Yves Lafaye & Frédéric Bertrand. *Modelling composite periodic Events*. In CIEL 2012, page 15, Rennes, 2012. www *Cité page 220.*

[Fleurey 07] Franck Fleurey, Benoit Baudry, Robert France & Sudipto Ghosh. *A Generic Approach For Automatic Model Composition*. In Aspect Oriented Modeling (AOM) Workshop, pages 7–15, Nashville, USA, 2007. www *Cité page 179.*

[Fowler 03] Martin Fowler. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2003. *Cité page 24.*

[France 07] Robert B France, Franck Fleurey, Raghu Reddy, Benoit Baudry & Sudipto Ghosh. *Providing Support for Model Composition in Metamodels*. In EDOC, pages 253–266, 2007. *2 citations pages 59 et 74.*

[Freksa 97] Christian Freksa. *Spatial and Temporal Structures in Cognitive Processes*. In Foundations of Computer Science : Potential - Theory - Cognition, pages 379–387, 1997. *Cité page 94.*

[Guillou 82] J. Guillou. *Variabilité des populations de Donax trunculus et Donax vittatus en baie de Douarnenez*. Netherlands Journal of Sea Research, vol. 16, pages 88–95, 1982. www *Cité page 193.*

[Gutierrez 07] Claudio Gutierrez, Carlos A Hurtado & Alejandro A Vaisman. *Introducing Time into RDF*. IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 2, pages 207–218, 2007. *Cité page 26.*

[Hillairet 11] Guillaume Hillairet. *EMFTriple*, 2011. `http://code.google.com/a/eclipselabs.org/p/emftriple/` *Cité page 225.*

[ISO 02] ISO. *Text of 19108 Geographic information - Temporal schema*, 2002. *3 citations pages 9, 23 et 71.*

[ISO 04] ISO. *ISO 8601*, 2004. `http://www.iso.org/iso/fr/catalogue_detail?csnumber=40874` *Cité page 24.*

[ISO/IEC 96] ISO/IEC. *ISO/IEC 14977 Information technology - Syntactic metalanguage - Extended BNF*, 1996. `http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf` *Cité page 58.*

[ISO/TC211 10] ISO/TC211. *ISO 19100 Models*, 2010. `http://www.isotc211.org/hmmg/HTML/index.htm` *3 citations pages 5, 23 et 70.*

[Jain 08] Ramesh Jain. *EventWeb : Developing a Human-Centered Computing System.* Computer, vol. 41, pages 42–50, 2008. *Cité page 26.*

[Jézéquel 12] Jean-Marc Jézéquel, Benoit Combemale & Didier Vojtisek. Ingénierie Dirigée par les Modèles : des concepts à la pratique. Références sciences. Ellipses, 2012. www *Cité page 55.*

[Jouault 06] Frédéric Jouault & Ivan Kurtev. *Transforming Models with ATL.* In Jean-Michel Bruel, editeur, Satellite Events at the MoDELS 2005 Conference, volume 3844 of *Lecture Notes in Computer Science*, pages 128–138. Springer Berlin / Heidelberg, 2006. www *Cité page 169.*

[Klyne 04] Graham Klyne & Jeremy J. Carroll. *Resource Description Framework (RDF) : Concepts and Abstract Syntax. Recommandation W3C*, 2004. *Cité page 23.*

[Koch 08] N. Koch, A. Knapp, G. Zhang & H. Baumeister. *UML-Based Web Engineering : An Approach Based on Standards.* Human-Computer Interaction Series, vol. 12, pages 157–191, 2008. *2 citations pages 169 et 171.*

[Konrad 05] Sascha Konrad & Betty H C Cheng. *Real-time specification patterns.* In Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on, pages 372–381, 2005. *Cité page 24.*

[Ladkin 86] Peter B Ladkin. *Time Representation : A Taxonomy of Internal Relations.* In Proceedings of the 5th National Conference on Artificial Intelligence, AAAI, pages 360–366, 1986. *Cité page 90.*

[Ladkin 87] Peter B. Ladkin. *Specification of time dependencies and synthesis of concurrent processes.* In ICSE '87 Proceedings of the 9th international conference on Software Engineering, pages 106–115. IEEE Computer Society Press, 1987. www *Cité page 91.*

[Le Tixerant 10] Matthieu Le Tixerant, Françoise Gourmelon, Cyril Tissot & David Brosset. *Modelling of human activity development in coastal sea areas.* Journal of Coastal Conservation, 2010. *2 citations pages 26 et 193.*

[Leban 86] Bruce Leban, David McDonald & David Forster. *A Representation for Collections of Temporal Intervals.* In AAAI'86 Proceedings, pages 367–371. AAAI Press, 1986. www *4 citations pages 44, 68, 75 et 124.*

[Lebranchu 11] Julien Lebranchu. *Étude des phénomènes itératifs en langue : Inscription discursive et Calcul aspectuo-temporel, vers un traitement automatisé.* These, Universit{é} de Caen, 2011. www *Cité page 35.*

[Ligozat 91] Gérard Ligozat. *On generalized interval calculi.* In AAAI-91, pages 234–240, 1991. *4 citations pages 21, 90, 98 et 114.*

[Ligozat 10] Gérard Ligozat. Raisonnement qualitatif sur le temps et l'espace. Ingénierie des langues. Hermes Science Publications, 2010. *2 citations pages 21 et 90.*

[López-Landa 12] Rosa López-Landa, Julieta Noguez, Esther Guerra & Juan de Lara. *EMF on Rails.* In Slimane Hammoudi, Marten van Sinderen & José Cordeiro, editeurs, ICSOFT 2012 - Proceedings of the 7th International Conference on Software Paradigm Trends, pages 273–278, Rome, 2012. SciTePress. *Cité page 169.*

[Mallet 10] Frédéric Mallet, Julien Deantoni, Charles André & Robert De Simone. *The Clock Constraint Specification Language for building timed causality models.* Innovations in Systems and Software Engineering, vol. 6, pages 99–106, 2010. www *Cité page 46.*

[Mallet 11] Frédéric Mallet. Logical Time @ Work for the Modeling and Analysis of Embedded Systems. LAP LAMBERT Academic Publishing, 2011. *Cité page 46.*

[Marchand 12] Jonathan Marchand, Benoit Combemale & Benoit Baudry. *A Categorical Model of Model Merging and Weaving.* In MiSe 2012 - 4th International Workshop on Modeling in Software Engineering, Zurich, Switzerland, 2012. Conference Publishing Solutions. www *Cité page 59.*

[Meiri 96] Itay Meiri. *Combining Qualitative and Quantitative Constraints in Temporal Reasoning.* Artificial Intelligence, vol. 87, no. 1-2, pages 343–385, 1996. *Cité page 82.*

[Meliá 06] Santiago Meliá & Jaime Gómez. *The WebSA Approach : Applying Model Driven Engineering to Web Applications.* Journal of Web Engineering, vol. 5, no. 2, pages 121–149, 2006. *Cité page 169.*

[Miller 03] Joaquin Miller & Jishnu Mukerji. *MDA Guide Version 1.0.1*, 2003. *Cité page 55.*

[Milner 82] Robin Milner. A Calculus of Communicating Systems. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982. *Cité page 54.*

[MIT 11] MIT. *Simile Exhibit*, 2011. `http://simile-widgets.org/exhibit/` *Cité page 171.*

[Moellering 76] H Moellering. *The potential uses of a computer animated film in the analysis of geographical patterns of traffic crashes.* Accident Analysis & Prevention, vol. 8, no. 4, pages 215–227, 1976. www *Cité page 194.*

[Mondeca 10] Mondeca, Modyco & L3i. *TKA (Temporal Knowledge Acquisition)*, 2010. `http://client2.mondeca.com/AccessPeriodEditor/` *2 citations pages 37 et 206.*

[Morin 08] Brice Morin, Jacques Klein, Olivier Barais & Jean-Marc Jézéquel. *A Generic Weaver for Supporting Product Lines.* In International Workshop on Early Aspects at ICSE'08, Leipzig, Germany, 2008. *Cité page 59.*

[Muller 05] Pierre-Alain Muller, Franck Fleurey & Jean-Marc Jézéquel. *Weaving Executability into Object-Oriented Meta-Languages.* In S Kent L. Briand, editeur, Proceedings of MODELS/UML'2005, volume 3713 of *LNCS*, pages 264–278, Montego Bay, Jamaica, 2005. Springer. www *4 citations pages 8, 57, 74 et 169.*

[Muller 08] Pierre-Alain Muller, Frédéric Fondement, Franck Fleurey, Michel Hassenforder, Rémi Schneckenburger, Sébastien Gérard & Jean-Marc Jézéquel. *Model Driven analysis and synthesis of textual concrete syntax.* Journal of Software and Systems Modeling (SoSyM), vol. 7, no. 4, pages 423–442, October 2008. www *Cité page 58.*

[Nebel 95] Bernhard Nebel & Hans-Jürgen Bürckert. *Reasoning about Temporal Relations : A Maximal Tractable Subclass of Allen's Interval Algebra.* J. ACM, vol. 42, no. 1, pages 43–66, 1995. *Cité page 90.*

[Niezette 92] Marc Niezette & Jean-Marc Stevenne. *An Efficient Symbolic Representation of Periodic Time.* In Proc. of the ISMM International Conference on Information and Knowledge Management CIKM'92, pages 161–168, Baltimore, MD, 1992. *2 citations pages 44 et 124.*

[Noyrit 12] Florian Noyrit, Sébastien Gérard & Bran Selic. *FacadeMetamodel : Masking UML.* In Model Driven Engineering Languages and Systems - 15th International Conference, MODELS 2012, volume 7590 of *Lecture Notes in Computer Science*, pages 20–35, Innsbruck, 2012. *Cité page 28.*

[Olivé 06] Antoni Olivé & Ruth Raventós. *Modeling events as entities in object-oriented conceptual modeling languages.* Data & Knowledge Engineering - Special issue : ER 2004, vol. 58, no. 3, pages 243–262, 2006. www *2 citations pages 28 et 66.*

[OMG 03] OMG. *Common Warehouse Metamodel (CWM) Specification. Version 1.1*, 2003. `http://www.omg.org/spec/CWM/1.1` *2 citations pages 23 et 56.*

[OMG 05] OMG. *UML Profile for Schedulability, Performance and Time (SPTP). Version 1.1*, 2005. `http://www.omg.org/spec/SPTP/1.1` *Cité page 24.*

[OMG 08] OMG. *Software Process Engineering Metamodel (SPEM). Version 2.0*, 2008. *Cité page 56.*

[OMG 11a] OMG. *Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT). Version 1.1*, 2011. `http://www.omg.org/spec/QVT/1.1` *3 citations pages 56, 57 et 169.*

[OMG 11b] OMG. *Meta Object Facility (MOF) Core Specification. Version 2.4.1*, 2011. `http://www.omg.org/spec/MOF/2.4.1` *Cité page 56.*

[OMG 11c] OMG. *MOF 2 XMI Mapping (XMI). Version 2.4.1*, 2011. `http://www.omg.org/spec/XMI/2.4.1/` *Cité page 56.*

[OMG 11d] OMG. *UML Profile For MARTE : Modeling And Analysis Of Real-Time Embedded Systems*, 2011. `http://www.omg.org/spec/MARTE/1.1` *Cité page 24.*

[OMG 12a] OMG. *Object Constraint Language (OCL). Version 2.3.1*, 2012. `http://www.omg.org/spec/OCL/2.3.1/` *2 citations pages 23 et 60.*

[OMG 12b] OMG. *Unified Modeling Language (UML). Version 2.4.1*, 2012. `http://www.omg.org/spec/UML/2.4.1/` *2 citations pages 23 et 56.*

[Osmani 12] Aomar Osmani. *Modélisation et raisonnement sur des données relationnelles.* Habilitation à diriger des recherches, Université Paris 13, 2012. *Cité page 90.*

[Pan 05] Feng Pan & Jerry R Hobbs. *Temporal Aggregates in OWL-Time.* In FLAIRS 2005, pages 560–565, 2005. *Cité page 34.*

[Parent 06] Christine Parent, Stefano Spaccapietra & Esteban Zimanyi. Conceptual Modeling for Traditional and Spatio-temporal Applications : The MADS Approach. Springer-Verlag, 2006. www *Cité page 25.*

[Patel-Schneider 04] P Patel-Schneider, P Hayes & I Horrocks. *OWL Web Ontology Language Semantics and Abstract Syntax. Recommandation W3C*, 2004. *Cité page 23.*

[Perry 06] Matthew Perry, Farshad Hakimpour & Amit P Sheth. *Analyzing theme, space, and time : an ontology-based approach.* In GIS, pages 147–154, 2006. *Cité page 26.*

[Perry 07] Matthew Perry, Amit P Sheth, Farshad Hakimpour & Prateek Jain. *Supporting Complex Thematic, Spatial and Temporal Queries over Semantic Web Data.* In GeoS, pages 228–246, 2007. *Cité page 26.*

[Perry 08] Matthew Perry. *A framework to support spatial, temporal and thematic analytics over semantic web data.* PhD thesis, Wright State University, 2008. *Cité page 26.*

[Perry 11] Matthew Perry, Prateek Jain & Amit P Sheth. *SPARQL-ST : Extending SPARQL to Support Spatiotemporal Queries.* In Geospatial Semantics and the Semantic Web, pages 61–86, 2011. *Cité page 26.*

[Pessemier 11] Toon De Pessemier, Sam Coppens, Erik Mannens, Simon Dooms, Luc Martens & Kristof Geebelen. *An Event Distribution Platform for Recommending Cultural Activities.* In WEBIST, pages 231–236, 2011. *Cité page 40.*

[Peuquet 94] Donna J. Peuquet. *It's about time : A conceptual framework for the representation of spatiotemporal dynamics in geographic information systems.* Annals of the Association of American Geographers, vol. 84, no. 3, pages 441–461, 1994. www *Cité page 193.*

[Peuquet 95] Donna J. Peuquet & Niu Duan. *An event-based spatiotemporal data model (ESTDM) for temporal analysis of geographical data.* International Journal of Geographical Information Science, vol. 9, no. 1, pages 7–24, 1995. www *Cité page 25.*

[Peuquet 02] Donna J. Peuquet. Representations of Space and Time. Guilford Press, 2002. *2 citations pages 25 et 194.*

[Price 00] Rosanne Price, Nectaria Tryfona & Christian S. Jensen. *Extended SpatioTemporal UML : Motivations, Requirements and Constructs.* Journal on Database Management, vol. 11, no. 4, pages 14–27, 2000. www *Cité page 25.*

[Price 02] Rosanne Price, Nectaria Tryfona & Christian S Jensen. *Extending UML for Space- and Time-Dependent Applications.* In Advanced Topics in Database Research, volume 1, pages 342–366. 2002. *Cité page 25.*

[Pustejovsky 03] James Pustejovsky, Jos'e Castano, Robert Ingria, Roser Sauri, Robert Gauzauskas, A Setzer & Graham Katz. *TimeML : Robust Specification of Event and Temporal Expression inText.* In Mark T Maybury, editeur, IWCS-5, Fifth International Workshop on Computational Semantics, New Directions in Question Answering, pages 28–34, Tilburg, Netherlands, 2003. AAAI Press. www *Cité page 37.*

[Pustejovsky 10] James Pustejovsky, Kiyong Lee, Harry Bunt & Laurent Romary. *ISO-TimeML : An International Standard for Semantic Annotation.* In Bente Maegaard Joseph Mariani Jan Odjik Stelios Piperidis Mike Rosner Daniel Tapias Nicoletta Calzolari (Conference Chair) Khalid Choukri, editeur, Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10), pages 394–397, Valletta, Malta, 2010. European Language Resources Association (ELRA). www *Cité page 34.*

[Reddy 05] Raghu Reddy, Robert France, Sudipto Ghosh, Franck Fleurey & Benoit Baudry. *Model Composition - A Signature-Based Approach.* In Aspect Oriented Modeling (AOM) Workshop, Montego Bay, Jamaica, 2005. *2 citations pages 59 et 179.*

[Rossi 08] G. Rossi, O. Pastor, D. Schwabe & L. Olsina. *Web Engineering : Modelling and Implementing Web Applications.* Human-Computer Interaction Series, vol. 12, 2008. *Cité page 171.*

[Roth 09] Robert E. Roth & Kevin S. Ross. *Extending the Google Maps API for event animation mashups.* Cartographic Perspectives, vol. 64, no. Fall 2009, pages 21–40, 2009. *Cité page 195.*

[Scherp 09] Ansgar Scherp, Thomas Franz, Carsten Saathoff & Steffen Staab. *F–a model of events based on the foundational ontology dolce+DnS ultralight.* In Proceedings of the fifth international conference on Knowledge capture, K-CAP '09, pages 137–144, New York, NY, USA, 2009. ACM. www *3 citations pages 29, 30 et 63.*

[Schmidt 06] Douglas C. Schmidt. *Model-Driven Engineering.* IEEE Computer Society, 2006. *Cité page 55.*

[Schobbens 07] Pierre-Yves Schobbens, Patrick Heymans, Jean-Christophe Trigaux & Yves Bontemps. *Generic semantics of feature diagrams.* Computer Networks, vol. 51, no. 2, pages 456–479, 2007. *Cité page 170.*

[Shaw 08] Ryan Shaw & Ray Larson. *Event Representation in Temporal and Geographic Context*. In Birte Christensen-Dalsgaard, Donatella Castelli, Bolette Ammitzbøll Jurik & Joan Lippincott, editeurs, Research and Advanced Technology for Digital Libraries, volume 5173 of *Lecture Notes in Computer Science*, pages 415–418. Springer Berlin Heidelberg, 2008. www *Cité page 25.*

[Shaw 09] Ryan Shaw, Raphaël Troncy & Lynda Hardman. *LODE : Linking Open Descriptions of Events*. In ASWC, pages 153–167, 2009. *Cité page 31.*

[Sheth 08] Amit P Sheth & Matthew Perry. *Traveling the Semantic Web through Space, Time, and Theme*. IEEE Internet Computing, vol. 12, no. 2, pages 81–86, 2008. *Cité page 26.*

[Shoham 09] Yoav Shoham & Kevin Leyton-Brown. Multiagent Systems : Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press, 2009. www *Cité page 193.*

[Svinterikou 99] Marianthi Svinterikou & Babis Theodoulidis. *TUML : A Method for Modelling Temporal Information Systems*. In Matthias Jarke & Andreas Oberweis, editeurs, Advanced Information Systems Engineering, volume 1626 of *Lecture Notes in Computer Science*, pages 456–461. Springer Berlin / Heidelberg, 1999. www *Cité page 24.*

[Taddesse 10a] Fekade Getahun Taddesse & Richard Chbeir. *Semantic aware RSS query algebra*. In iiWAS'2010 - The 12th International Conference on Information Integration and Web-based Applications and Services, pages 291–298, 2010. www *Cité page 170.*

[Taddesse 10b] Fekade Getahun Taddesse, Joe Tekli, Richard Chbeir, Marco Viviani & Kokou Yétongnon. *Semantic-based Merging of RSS Items*. World Wide Web, vol. 13, no. 1-2, pages 169–207, 2010. www *Cité page 170.*

[Teissèdre 10] Charles Teissèdre, Delphine Battistelli & Jean-Luc Minel. *Resources for Calendar Expressions Semantic Tagging and Temporal Navigation through Texts*. In Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10), pages 3572–3577, Valletta, Malta, 2010. European Language Resources Association (ELRA). www *6 citations pages 6, 12, 36, 37, 147 et 200.*

[Terenziani 00] Paolo Terenziani. *Integrated Temporal Reasoning with Periodic Events*. Computational Intelligence, vol. 16, no. 2, pages 210–256, 2000. *4 citations pages xi, 31, 45 et 187.*

[Terenziani 02] Paolo Terenziani. *Toward a Unifying Ontology Dealing with Both User-Defined Periodicity and Temporal Constraints About Repeated Events*. Computational Intelligence, vol. 18, no. 3, pages 336–385, 2002. *2 citations pages 44 et 124.*

[Tillier 10] Ion Tillier & Cyril Tissot. Modelling Nature-Society interactions and spatial use conflicts in coastal areas coupling MAS with GIS, chapitre Modelling, page 20. Nova Science Publishers, 2010. *2 citations pages 26 et 196.*

[Tissot 04] Cyril Tissot, Matthieu Le Tixerant, Françoise Gourmelon & Iwan Le Berre. *Modelling interactions between human activities and coastal zone environment*. In Littoral 2004, Aberdeen, Scotland, 2004. Cambridge Publications. *2 citations pages 26 et 196.*

[Troncy 08] Raphaël Troncy. *Bringing the IPTC News Architecture into the Semantic Web*. In International Semantic Web Conference, pages 483–498, 2008. *Cité page 29.*

[Tryfona 99] Nectaria Tryfona & Christian S. Jensen. *Conceptual Data Modeling for Spatiotemporal Applications*. Geoinformatica, vol. 3, no. 3, pages 245–268, 1999. WWW *Cité page 25.*

[Tryfona 00] Nectaria Tryfona & Christian S Jensen. *Using Abstractions for Spatio-Temporal Conceptual Modeling*. In Proceedings of the 2000 ACM Symposium on Applied Computing, volume 1, pages 313–322, 2000. *Cité page 25.*

[van Deursen 00] Arie van Deursen, Paul Klint & Joost Visser. *Domain-Specific Languages : An Annotated Bibliography*. SIGPLAN Notices, vol. 35, no. 6, pages 26–36, 2000. *Cité page 57.*

[Versichele 12] M Versichele, T Neutens, M Delafontaine & N Van de Weghe. *The use of Bluetooth for analysing spatiotemporal dynamics of human movement at mass events : A case study of the Ghent Festivities*. Applied Geography, vol. 32, pages 208–220, 2012. *2 citations pages 189 et 190.*

[Vilain 82] Marc B Vilain. *A system for reasoning about time*. In AAAI-82, pages 197–201, 1982. *2 citations pages 91 et 96.*

[W3C 06] W3C. *Time Ontology in OWL*, 2006. `http://www.w3.org/TR/owl-time/` *4 citations pages 32, 33, 34 et 35.*

[W3C 11] W3C. *Widget Packaging and XML Configuration*, 2011. `http://www.w3.org/TR/widgets/` *Cité page 171.*

[Wallgrün 06] Jan Oliver Wallgrün, Lutz Frommberger, Diedrich Wolter, Frank Dylla & Christian Freksa. *Qualitative Spatial Representation and Reasoning in the SparQ-Toolbox*. In Spatial Cognition, pages 39–58, 2006. *2 citations pages 46 et 112.*

[Weida 92] Robert A Weida & Diane J Litman. *Terminological Reasoning with Constraint Networks and an Application to Plan Recognition*. In KR, pages 282–293, 1992. *Cité page 90.*

[Weiser 08] Stéphanie Weiser, Philippe Laublet & Jean-Luc Minel. *Automatic Identification of Temporal Information in Tourism Web Pages*. In Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 2008. www *2 citations pages 6 et 36.*

[Weiss 00] Gerhard Weiss. Multiagent systems, A modern approach to distributed artificial intelligence. The MIT Press, the mit pr edition, 2000. www *Cité page 193.*

[Westphal 09] Matthias Westphal, Stefan Wölfl & Zeno Gantner. *GQR : A Fast Solver for Binary Qualitative Constraint Networks*. In AAAI Spring Symposium : Benchmarking of Qualitative Spatial and Temporal Reasoning Systems, pages 51–52, 2009. www *2 citations pages 46 et 112.*

[Worboys 04] Michael Worboys & Kathleen Hornsby. *From Objects to Events : GEM, the Geospatial Event Model*. In Max J Egenhofer, Christian Freksa & Harvey J Miller, editeurs, Geographic Information Science, volume 3234 of *Lecture Notes in Computer Science*, pages 327–343. Springer Berlin / Heidelberg, 2004. www *Cité page 25.*

[Zaki 11] Chamseddine Zaki, Myriam Servieres & Guillaume Moreau. *Implementing conceptual spatiotemporal model into object DBMS with semantic preserving*. In IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services, ICSDM 2011, pages 160–165, 2011. *2 citations pages 25 et 66.*

[Ziemann 03] Paul Ziemann & Martin Gogolla. *OCL Extended with Temporal Logic*. In Manfred Broy & Alexandre Zamulin, editeurs, Perspectives of System Informatics, 5th International Andrei Ershov Memorial Conference, PSI 2003, volume 2890 of *Lecture Notes in Computer Science*, pages 351–357. Springer Berlin / Heidelberg, 2003. www *Cité page 27.*

Index

Modélisation d'événements composites répétitifs, propriétés et relations temporelles

Résumé :

La modélisation des événements et de leurs propriétés temporelles concerne des types variés d'utilisateurs et de communautés scientifiques. Nous nous plaçons dans le cadre du paradigme Objet et construisons un métamodèle opérationnel servant de représentation pivot indépendante du métier pour représenter des événements composites avec leurs propriétés structurelles et temporelles.
Le métamodèle PTOM (Periodic Temporal Occurrence Metamodel) prend en compte l'expression de contraintes, structurelles sur les événements, ou géométriques, topologiques et relationnelles sur la temporalité de leurs occurrences. Il privilégie la représentation en intension (*vs* extension) des occurrences d'événements périodiques.
PTOM étend la norme ISO 19108 et s'adapte aux standards EventsML G2 et iCalendar. Sur un plan théorique, nous étendons les algèbres d'intervalles d'ALLEN et de LIGOZAT et proposons un système de relations topologiques entre intervalles non convexes (ALLEN*) dont nous étudions les propriétés. Ces résultats sont intégrés dans PTOM.
La première application de PTOM est la spécification de la sémantique du calendrier grégorien. Les éléments calendaires sont réintroduits en tant qu'événements périodiques dans PTOM ce qui renforce son pouvoir expressif.
PTOM a été mis en œuvre lors d'un projet ANR sur des corpus d'événements journalistiques (agences de presse) concernant les loisirs et la culture. L'Ingénierie Dirigée par les Modèles a été utilisée pour la conception et l'exploitation de PTOM. Cela permet de gérer la complexité, d'assurer la maintenabilité et la cohérence de l'ensemble et enfin de générer automatiquement des interfaces pour les pourvoyeurs ou utilisateurs de données.

Mots-clés : Evénement, Périodicité, Relations d'ALLEN, Propriétés temporelles, IDM.

Modeling periodic composite events, temporal properties and relations

Summary :

Modelling events with their temporal properties concerns many users and scientific communities. We adopted the Object paradigm and designed an operational metamodel which stands as a pivot business independent representation for composite events accompanied with their structural and temporal properties.
PTOM metamodel (Periodic Temporal Occurrence Metamodel), accounts for structural constraints upon events and geometric, topologic or relational constraints upon their temporal occurrences. It gives prominence to intensional representations of periodic events occurrences vs extensional ones.
PTOM extends ISO 19108 standard and fits EventsML G2 and iCalendar. From a theoretical viewpoint, we extend ALLEN's and LIGOZAT's interval algebras and propose a special set of topological relations between non convex intervals (ALLEN*), and study its properties. These results are part of PTOM. The first application of PTOM, results in a specification of the Gregorian calendar semantics. Calendar elements are reinserted as periodic events in PTOM thus enhancing its expressiveness.
PTOM was also experimented upon a corpus of journalistic (press agencies) events dedicated to leisure and culture at the occasion of an ANR project. Model Driven Engineering was extensively used for PTOM design and use. It allows to manage complexity and to ensures maintainability, consistency and eventually can automatically generate data provider or end user interfaces as well.

Keywords : Event, Periodicity, ALLEN's relations, Temporal properties, MDE.

Laboratoire L3i - Informatique, Image, Interaction

Pôle Sciences et Technologies, Université de La Rochelle, avenue Michel Crépeau

17042 LA ROCHELLE cedex 01