Langage de spécification et outil de vérification pour le consentement et la nécessité des données fondés sur une classification relative au respect de la vie privée

Myriam Clouet

université PARIS-SACLAY

Langage de spécification et outil de vérification pour le consentement et la nécessité des données fondés sur une classification relative au respect de la vie privée

Specification language and verification tool for consent and data necessity based on a classification related to privacy compliance

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 : sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique
Graduate School : Informatique et sciences du numérique, Référent : Faculté des sciences d'Orsay

Thèse préparée dans l'unité de recherche Institut LIST (Université Paris-Saclay, CEA), sous la direction de Julien Signoles, directeur de recherches, la co-encadrement de Mathilde Arnaud, ingénieure chercheuse, la co-encadrement de Thibaud Antignac, ingénieur chercheur

Thèse soutenue à Paris-Saclay, le 3 mai 2024, par

Myriam CLOUET

Composition du jury

Membres du jury avec voix délibérative

Pascale Le Gall Professeure, Centrale Supélec	Présidente
Mathieu Cunche Professeur, INSA-Lyon	Rapporteur & Examinateur
Yves Ledru Professeur, Université Grenoble Alpes	Rapporteur & Examinateur
Cédric Eichler Associate Professor, INSA Val de Loire	Examinateur

THESE DE DOCTORAT

NNT : 2024UPASG023

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Langage de spécification et outil de vérification pour le consentement et la nécessité des données fondés sur une classification relative au respect de la vie privée

Mots clés : Vie privée, Classification, Modélisation, Méthodes formelles, Langage de spécification, Vérification de programmes et de modèles

Résumé : La vie privée est un droit fondamental que l'on retrouve dans plusieurs lois dans le monde entier. Donc, vérifier qu'un système respecte la vie privée est crucial. Cependant, la vie privée est une notion complexe. Assurer qu'un système respecte la vie privée nécessite de vérifier que les propriétés de vie privées sont respectées durant tout le cycle de vie, ce qui complique le processus de vérification. Une grande variété de solutions ont été proposées pour améliorer le respect de la vie privée. Il est souvent difficile de précisément identifier quand elles ciblent la même problématique.

Dans cette thèse, j'adresse ces problèmes en proposant une façon de classifier des articles concernant la vie privée et une approche pour vérifier des propriétés de vie privée à deux étapes du cycle de vie. Je propose GePyR, une nouvelle représentation de la vie privée, générique et spécialisable, et une ontologie instanstiable, PyCO, qui modélise les éléments clés de la vie privée et leurs relations. Cette classification est évaluée sur la littérature, en réalisant un *Systematic Mapping Study*. Dans cette thèse, je formalise également deux propriétés de vie privée, la *conformité aux finalités consenties* et la *conformité à la nécessité des données*. Je propose une nouveau langage de spécification, CSpeL, qui permet de formaliser les éléments nécessaires pour vérifier ces propriétés, et introduis un nouvel outil, CASTT, pour vérifier ces propriétés sur des traces d'exécutions, sur un modèle ou un programme. Cette approche est appliquée sur deux cas d'utilisation à deux niveaux d'abstraction, pour évaluer sa correction, son efficacité et son utilité.

Title : Specification language and verification tool for consent and data necessity based on a classification related to privacy compliance

Keywords : Privacy, Classification, Modeling, Formal methods, Specification Language, Model and Program Verification

Abstract : Privacy is a fundamental right implemented in many laws around the world. Therefore, verifying that a system complies with privacy is crucial. However, privacy is also a complex notion. Besides, ensuring compliance of a software system with respect to privacy requires to verify that the expected privacy properties hold during the whole system lifecycle. It usually involves different abstraction levels, which complicates the verification process. Many different solutions have been proposed to enhance privacy. It is often quite hard to precisely identify whether they target the same problem.

This thesis addresses these issues by proposing a way to classify privacy papers and an approach to verify privacy properties at two different development stages. It proposes GePyR, a new generic and specializable representation, and an instantiable ontology, PyCO, that models key privacy elements and their relationships. This classification is evaluated on the literature, by using a Systematic Mapping Study. This thesis also formalizes two privacy properties, *purpose compliance* and *necessity compliance*. It proposes a new specification language, named CSpeL, that allows engineers to formally specify key system elements with regards to these properties, and introduces a new tool, CASTT, to verify the aforementioned properties on execution traces, on a model or on a program. This approach is applied on two use cases, both at two abstraction levels (model and code), in order to evaluate its correctness, its efficiency and its usefulness.

Remerciements institutionnels

Abréviations

AL$_{Cpx}$	Achat en Ligne - version Complexe
AL$_{Spl}$	Achat en Ligne - version Simple
BPMN	*Business Process Model and Notation* (Modèle de procédé d'affaire et notation)
CASTT	*Context Annotation Translation Tool* (Outil de traduction d'annotation de contexte)
CC	Compte Client
CNIL	Commission nationale de l'informatique et des libertés
CSpeL	*Context Specification Language* (Langage de spécification de contexte)
CTL	*Computation Tree Logic* (Logique des arbres de calcul)
EPR	*Electronic Patient Record* (Dossier électronique du patient)
GePyR	*Generic Privacy Representation* (Représentation générique liée à la vie privée)
GDM$_{Cpx}$	Gestion Données Médicales - version Complexe
GDM$_{Spl}$	Gestion Données Médicales - version Simple
GDPR	*General Data Protection Regulation* (Règlement Général sur la Protection des Données)
GSpeL	*Generation Specification Language* (Langage de spécification pour la génération)
LTL	*Linear-time temporal logic* (Logique temporelle linéaire)
ML	*Model Level* (Niveau Modèle)
PbD	*Privacy by Design* (Respect de la vie privée par conception)
PL	*Program Level* (Niveau Programme)
PPRL	*Privacy Preserving Record Linkage* (Liaison d'enregistrements préservant la vie privée)
PyCO	*Privacy Context Ontology* (Ontologie du contexte lié à la vie privée)
RL	*Requirement Level* (Niveau Exigence)
RGPD	Règlement Général sur la Protection des Données
SMS	*Systematic Mapping Study* (étude de cartographie systématique)

Remerciements

Je remercie les membres du jury, Pascale Le Gall, Mathieu Cunche, Yves Ledru et Cédric Eichler pour leur attention, leurs questions pertinentes pendant la soutenance, et pour avoir permis que cet évènement – qui me semblait une épreuve insurmontable – devienne un échange constructif de recherche et de partages d'idées qui a continué au-delà de la soutenance. Merci de m'avoir permis de voir que mes travaux étaient pertinents et intéressants au-delà du contexte de la thèse – qui peut vite ressembler à un vase clos, à une simple tâche à réaliser – et ainsi l'ouvrir sur le reste de la communauté scientifique et me rappeler son inclusion dans le monde de la recherche.

Je remercie également les rapporteurs, Mathieu Cunche et Yves Ledru, pour leur lecture attentive de mon manuscrit, malgré le très grand nombre de pages, et pour leurs remarques pertinentes. La lecture de leurs rapports m'a réconfortée dans la qualité de mon travail. Je remercie tout particulièrement Yves Ledru d'avoir pris le temps de m'indiquer le grand nombre de corrections orthographiques qu'il restait à faire, malgré le grand travail de relecture réalisé par mes encadrants et leurs nombreuses remarques et corrections.

Je remercie aussi mes encadrants Thibaud Antignac, Mathilde Arnaud et Julien Signoles, qui m'ont guidée, aidée, challengée, supportée et soutenue durant toutes ces années. Merci d'avoir supporté tous mes questionnements personnels sur mon devenir dans la recherche. Merci à Thibaud d'avoir voulu me pousser toujours plus loin dans la réflexion autour de la vie privée, dans ces questionnements et remises en cause du travail déjà réalisé afin d'en consolider les résultats. Merci à Julien d'avoir dirigé cette thèse au plus proche malgré son emploi du temps surchargé et d'avoir toujours pris le temps, bien souvent sur son propre temps, de relire en détail mes articles et chapitres de manuscrit. Merci à Mathilde qui a bien voulu reprendre l'encadrement de la thèse en cours de route et qui s'est tout particulièrement investie, tant humainement que professionnellement, dans cette tâche complexe et ardue. Toujours disponible et à l'écoute pour toutes mes questions ou états d'âmes.

Je remercie aussi tous mes collègues du CEA qui m'ont permis de vivre ces années de travail dans une ambiance chaleureuse et amicale. Par leur bienveillance et leurs discussions ils ont réussi à faire sortir l'ours que je suis de sa caverne et à me révéler bien plus sociale que je ne le pensais. Je ne pourrais bien sûr pas tous les nommer, tellement il y en a et au vu de ma petite mémoire des noms (que ceux qui ne sont pas mentionnés ici ne m'en veuillent pas trop mais soient rassurés sur le fait que je les remercie quand même de tout le temps passé avec eux). Merci notamment à Yackolley et à Elouan pour leurs discussions et leurs partages de mangas. Merci à Augustin qui supportait mes visites impromptues à son bureau, même pour simplement partager une musique idiote. Merci à Ewa pour nos nombreux échanges sur nos familles respectives, moments volés durant nos tâches respectives. Merci aux gens de la liste lsl-misc qui m'ont permis de partager des moments agréables en dehors des murs du laboratoire et de me faire vivre une vie sociale peut-être un peu atypique mais tellement normale pour moi. Merci à Julien G., David, Tibo, Elouan, Florent et Jules qui ont su supporter, occuper mon fils et apprécier sa présence durant toutes nos soirées ou sorties, et ainsi me faire profiter de moments vraiment conviviaux sans que je sois obligée de choisir entre mes amis et mon fils. Merci à Patrick Tessier, Xavier, Vincent, Quentin et tous les autres qui m'ont permis d'échanger sur des sujets plus techniques et sur la vie en entreprise au sein du CEA. Merci à Guillaume d'avoir toujours été une oreille attentive. Merci à mon chef de laboratoire, Stéphane Salmons, qui a accepté mes extensions de thèse, qui a cru en moi et en mon travail jusqu'au bout afin que je puisse le mener le plus loin possible.

Je remercie également mes amis de Grenoble, que je n'oublie pas malgré la distance. Merci à Solenne, Charlie, Bruno qui m'ont soutenue et encouragée à faire cette thèse, qui ont cru en moi et en mes capacités. Merci de nos longues discussions sur des sujets tellement divers, sur nos partages d'expériences, toujours bienveillants et à l'écoute même lors de sujets plus personnels. Merci à Alexis qui a été un ami fidèle durant toutes mes années universitaires. Une simple rencontre du hasard qu'il n'a pas voulu laisser passer. Merci de m'avoir rattrapée dans les couloirs pour ne pas laisser s'échapper cette amitié.

Je tiens aussi à remercier ma famille qui m'a encouragée pendant toutes mes études et durant ma thèse. Merci à mes parents. Merci à mon père qui m'a toujours poussée à donner le meilleur de moi-même et à continuer d'aller toujours plus loin. Merci à ma mère pour toute l'aide qu'elle m'a apportée pendant mes études lorsque j'étais seule pour élever mon fils. Merci pour tous ces coups de téléphone, que ce soit pour des discussions banales ou pour me remonter le moral et m'aider à traverser le long chemin de la parentalité. Merci à mon frère, ma sœur, mon cousin et à ma tante qui m'ont toujours soutenue. Merci à ma grand-mère Louba qui a traversé tellement d'épreuves mais qui malgré tout a toujours su garder son sourire et m'accueillir avec, et dont la ténacité m'inspire au quotidien.

Je remercie grandement mon fils Shun qui m'a soutenue durant tout mon parcours universitaire, qui partageait la joie de mes bonnes notes et mes publications, me soutenant toujours. Être parent célibataire et réaliser des études en même temps, ce n'est pas évident, et c'est même bien souvent épuisant ; mais quel bonheur cela m'a apporté. Pour son enfant, on est toujours prêt à déplacer des montagnes, à faire des efforts dont on ne se sent pas capable pour soi-même (même simplement se lever tôt les weekends ou manger équilibré), ce qui nous pousse à toujours donner le meilleur de soi. J'adorais lui raconter mes journées, mes cours (même si ça lui faisait dessiner des automates et des règles d'inférence en crèche au lieu de dessiner des bonshommes ou des animaux). Grâce à lui j'ai pu être moins dure envers moi-même et relativiser mes difficultés et mes échecs comme je voulais l'aider pour les siens. Merci pour tous ses câlins, ses bisous, ses rires, sa joie et son amour.

Je remercie également mon mari André que j'ai rencontré au CEA durant ma thèse. Quelle chance de l'avoir rencontré. Déménager pour réaliser ma thèse dans une autre ville, quitter mes amis, ma vie établie, ma routine ne m'enchantait guère et pourtant si je n'étais pas venue jusqu'ici, je ne l'aurais sûrement jamais rencontré. Rien que pour cela, je suis bien heureuse que ma thèse ait eu lieu ici et non sur Grenoble. Merci de m'avoir toujours soutenue, aidée et supportée dans les moments difficiles. Merci pour tout le bonheur qu'il nous apporte à moi et à Shun. En plus de l'aide apportée sur le plan personnel, merci pour toute l'aide qu'il m'a apportée sur le plan professionnel. Merci de m'avoir aidé grandement en relecture, dépannage informatique, utilisation d'outils. Même s'il n'était pas officiellement mon correspondant informatique ni mon encadrant, il était toujours prêt à m'aider, même si cela lui rajoutait du travail supplémentaire. Merci également d'avoir préparé le pot de thèse selon mes envies et d'avoir même pensé à tout ce que j'oubliais (la nourriture sucrée ou même des gobelets). On forme une super équipe.

Table des matières

Table des figures

Liste des tableaux

Liste des définitions

Liste des exemples

Liste des algorithmes

Liste des théorèmes et preuves associées

1 - INTRODUCTION

1.1 . Contexte

L'informatique est de plus en plus présente dans notre vie de tous les jours. Les logiciels ont envahi nos espaces : nos ordinateurs, nos téléphones, mais aussi dans les voitures, les hôpitaux, ou les bibliothèques. Avec la démocratisation d'internet, de plus en plus d'actions sont réalisées en ligne : faire des achats, prendre un rendez-vous médical, ou bien encore payer ses impôts. D'autres objets ont également fait leur apparition : les objets connectés (par exemple, les assistants vocaux et la domotique). Pratiques et requérant peu d'interventions explicites de la part de l'utilisateur, leur présence au quotidien est en constante augmentation.

Tous ces systèmes informatiques nous aident au quotidien et facilitent les tâches que l'on doit réaliser, mais en même temps leur fonctionnement est de plus en plus opaque, surtout pour les non experts. Par exemple, bien des gens ne connaissent pas le fonctionnement des *cookies* sur Internet. Récemment on m'a encore demandé : « Mais pourquoi ont-ils besoin de *cookies* ? C'est quoi ? À quoi servent-ils ? », et ce n'était pas une personne ayant vécu loin du numérique une grande partie de sa vie mais bien une personne ayant grandi dans l'ère numérique et ayant l'habitude d'utiliser ces technologies. Le fonctionnement de ces systèmes est opaque, mais l'utilisation faite des données personnelles l'est encore plus. De nos jours, qui peut être sûr, à tout moment, de quelles données personnelles sont collectées, utilisées, pour quoi et par qui ?

Ces problématiques liées au traitement des données personnelles relèvent de la vie privée. De telles questions existent depuis longtemps : elles existaient déjà lors du développement de la photographie au dix-neuvième siècle [27]. L'avènement des réseaux sociaux, des smartphones et l'informatisation des infrastructures font que la collecte et l'utilisation des données ne cessent d'augmenter et renforcent ces problématiques. Cette utilisation massive a conduit à plusieurs scandales dont celui célèbre de *Facebook-Cambridge Analytica* [1]. Ce scandale a mis en évidence le fait que Facebook a partagé illégalement les informations de plus de quatre-vingt sept millions de ses utilisateurs avec la société Cambridge Analytica, une entreprise de conseil en gestion politique. Ces données auraient été utilisées, notamment en 2016, pour influencer des votes (par exemple, lors du référendum sur le Brexit et pour l'élection de Donald Trump) [2].

Des lois et des règlements ont été établis autour du monde afin d'améliorer

1. https://www.francetvinfo.fr/internet/reseaux-sociaux/facebook/scandale-facebook-les-donnees-de-2-7-millions-d-europeens-ont-pu-etre-concernees_2692974.html
2. https://www.lesechos.fr/tech-medias/hightech/cambridge-analytica-facebook-ecope-dune-amende-symbolique-142961

la protection de la vie privée, notamment le règlement général sur la protection des données (RGPD) dans l'Union européenne [55] mais aussi le *Privacy Act* en Australie [13] ou la loi sur la protection des données personnelles au Japon [128]. Les traitements de données personnelles dans ces régions doivent obligatoirement les respecter sous peine de sanctions. De telles sanctions ont récemment été prononcées contre Google et WhatsApp. En 2019, Google a ainsi été condamné par la CNIL (Commission nationale de l'informatique et des libertés) à payer 50 millions d'euros car, d'après elle, l'entreprise ne respectait pas le RGPD. Google manquait de transparence et présentait des informations inadéquates aux personnes concernées, ce qui rendait invalide le consentement qu'elle avait recueilli pour le traitement des données personnelles [3]. De nouveau, en 2021, Google a été condamné par la CNIL à payer 150 millions d'euros car ses sites phares, `google.fr` et `youtube.com`, ne permettaient pas de refuser les cookies facilement [4]. Ce n'est pas la seule grande entreprise à avoir été condamnée pour non-respect de la législation en matière de protection des données. En 2021, c'est au tour de WhatsApp de devoir payer 225 millions d'euros, cette fois-ci suite à la décision de l'*Irish Data Protection Commission*, à cause, une fois encore, d'un manque de transparence envers les utilisateurs sur la manière dont l'entreprise traite leurs données personnelles [5]. Suite à cela, WhatsApp a demandé explicitement le consentement de ses utilisateurs pour traiter leurs données personnelles, mais l'*Irish Data Protection Commission* a jugé que la façon de faire ne respectait pas le RGPD. Les utilisateurs étaient, d'une certaine manière, « forcés » d'accepter les nouvelles conditions d'utilisation de l'application. Ainsi, WhatsApp a encore dû débourser 5,5 millions d'euros [6].

Différentes raisons peuvent expliquer de tels écarts à la réglementation parmi lesquelles se trouve le manque de méthodes et d'outils de conception et de vérification du respect de la vie privée par un système logiciel. De tels outils ou méthodes permettraient ainsi de pouvoir vérifier qu'un tel système respecte la vie privée, cela avant même qu'il ne soit utilisé.

La création de tels outils et méthodes est complexe pour plusieurs raisons. En particulier, le concept de vie privée et de son respect ne signifie pas la même chose pour tout le monde. Par exemple, le concept de vie privée n'est pas le même aux États-Unis, en Chine ou dans l'Union européenne. La vie privée est un terme qui est utilisé pour désigner un grand nombre de notions différentes et le terme en lui-

3. https://www.lemonde.fr/pixels/article/2019/01/21/donnees-personnelles-la-cnil-condamne-google-a-une-amende-record-de-50-millions-d-euros_5412337_4408996.html

4. https://www.lesechos.fr/tech-medias/hightech/cookies-la-cnil-inflige-des-amendes-records-a-google-et-facebook-1377179

5. https://www.lemonde.fr/pixels/article/2021/09/02/donnees-personnelles-whatsapp-sanctionne-d-une-amende-record-par-le-regulateur-irlandais_6093152_4408996.html

6. https://www.dataprotection.ie/en/news-media/data-protection-commission-announces-conclusion-inquiry-whatsapp

même est vague [159]. En effet, chaque partie prenante d'un système a différentes façons d'appréhender la vie privée selon ses besoins. Par exemple, les instances légales n'ont pas la même façon d'appréhender la vie privée en informatique que les ingénieurs qui développent le système ou même que les utilisateurs dudit système.

Ainsi, « qu'est-ce que la vie privée en informatique ? » Pour répondre à cette question, je propose un premier ensemble de contributions, correspondant à la première partie du manuscrit. J'identifie les types de représentation de la vie privée en informatique dans la littérature, que ce soit d'un point de vue légal ou technique. Je propose une nouvelle représentation générique, GePyR, qui repose sur des catégories permettant de regrouper ces représentations selon les notions qu'elles considèrent (telles que la visibilité des données ou leurs traitements). Ainsi, il est possible de réaliser une classification d'articles répondant à des problématiques de vie privée similaire, en s'appuyant sur les notions de vie privée qu'ils considèrent. Ensuite, en restreignant aux notions propres à la vie privée, et non communes au domaine de la sécurité, je propose une ontologie, PyCO, qui permet de représenter les éléments à prendre en compte, et leurs relations, pour vérifier le respect de la vie privée en informatique. Ainsi, il est possible de réaliser une classification d'articles selon leur façon de représenter ces éléments.

Une fois avoir défini ce que signifiait la vie privée en informatique, il reste encore une question : « Comment vérifier formellement qu'un système la respecte ? » Comme vu précédement, il serait utile d'avoir plus de méthodes et d'outils de conception et de vérification du respect de la vie privée par un système logiciel. De plus, proposer une approche formelle permet d'augmenter l'assurance dans cette vérification. Pour répondre à cette question, je propose un autre ensemble de contributions, correspondant à la seconde partie du manuscrit. Je propose un langage de spécification, CSpeL, qui permet de formaliser un système dans un contexte lié à la vie privée. CSpeL s'appuie sur les éléments identifiés par PyCO et leur relations. Je formalise deux propriétés de vie privée, une liée au consentement et une autre à la nécessité des données. Ces propriétés ne couvrent pas toutes les notions de la vie privée mais permettent de prendre en compte des notions importantes quoique moins couvertes que d'autres, comme la visibilité des données, dans la littérature. Je propose également un outil de vérification formelle, CASTT, permettant de vérifier qu'un système respecte ces propriétés, à partir d'une spécification écrite en CSpeL.

Ainsi, ensemble ces contributions permettent d'aider à vérifier formellement qu'un système informatique respecte la vie privée.

1.1.1 . Vie privée et protection des données

Parfois, la littérature, et notamment le RGPD, mentionne non pas la vie privée mais la protection des données. Quelle est la différence ?

On pourrait penser que la protection des données ne correspond qu'aux mesures mises en places pour protéger les données, et donc que cette notion serait incluse dans une notion plus large de vie privée. Gellert et Gutwirth ont étudié

cette question [62]. Selon eux, la vie privée peut couvrir un large ensemble de problématiques, liées à l'intégrité corporelle, au secret de la correspondance et des communications, ou bien encore liées à l'identité, tandis que le but de la protection des données est de réguler le traitement des données personnelles. Pourtant, ils considèrent que les sémantiques des termes de vie privée et de protection des données se recoupent. L'un comme l'autre sont à la fois plus larges et plus précis que l'un l'autre. En effet, selon Gellert et Gutwirth, la protection des données peut concerner des traitements de données personnelles qui n'impactent pas la vie privée. Quant à la vie privée, elle peut concerner des notions qui ne sont pas liées au traitement des données personnelles. Ces deux termes seraient donc distincts même si leurs sémantiques se recoupent.

Pourtant, Hughes, qui a notamment étudié en détail la vie privée pour en identifier deux concepts centraux [79], n'a pas la même façon de définir ce qu'est la vie privée. En effet, il écrit que même les cours de justice et le monde académique n'arrivent pas à définir de façon uniforme et constante ce qu'est la vie privée. D'après lui, la théorie la plus adoptée est que la vie privée correspond au contrôle des informations personnelles. Les problématiques de la vie privée ne concerneraient ainsi que les données personnelles. Or, Hughes adopte un autre point de vue. Il considère que la vie privée repose sur « le désir subjectif » d'un individu « à éviter que ses données personnelles ne soient utilisées pour le blesser ».

Il n'existe ainsi pas de différence formellement établie entre vie privée et protection des données. Dans la suite du manuscrit, je considère que les problématiques liées au respect de la vie privée **en informatique** et à la protection des données sont similaires, qu'elles concernent les données personnelles sans être limitées à leur traitement. J'utilise plutôt le terme « (respect de la) vie privée » que « protection des données » car cela évite au lecteur de penser que le terme utilisé ne concerne que les mécanismes mis en place pour protéger les données personnelles.

Au delà de la complexité liée à la notion de vie privée, garantir qu'un système est correct est déjà une tâche complexe en soi. Plusieurs types de solutions ont été proposés. Certaines sont des méthodes de développement (afin d'appliquer de bonnes pratiques de développement). Je présente ci-dessous quelques-uns de ces types de solutions, les plus connus, et le positionnement de ma thèse par rapport à ceux-ci.

1.1.2 . Développement logiciel et cycle de vie

Le développement d'un logiciel est constitué d'un ensemble de phases comme la planification, la création, le test et le déploiement du système. Ces phases constituent le cycle de vie du logiciel. Ce cycle représente des étapes que doivent suivre les développeurs d'un système pour aboutir à un système de haute qualité. Plusieurs modèles de cycle de vie ont été proposés dans la littérature. Chacun a ses avantages et ses inconvénients. Chacun est donc plus ou moins adapté aux besoins

du système à développer et à son contexte. Ci-dessous, je donne quelques exemples de modèles très connus.

Le modèle en cascade (*Waterfall model*) [152] - Il s'agit d'un modèle séquentiel. Cela signifie que les phases sont réalisées les unes à la suite des autres et une seule phase est effectuée à la fois. De plus, une phase ne peut commencer que si la précédente est terminée. Le cycle de développement associé est représenté sur la figure 1.1 et ses différentes phases sont :

- analyse des exigences (*Requirement Analysis*) ;
- conception du système (*System Design*) ;
- implémentation (*Implementation*) ;
- réalisation de tests (*Testing*) ;
- déploiement (*Deployment*) ; et
- maintenance (*Maintenance*).

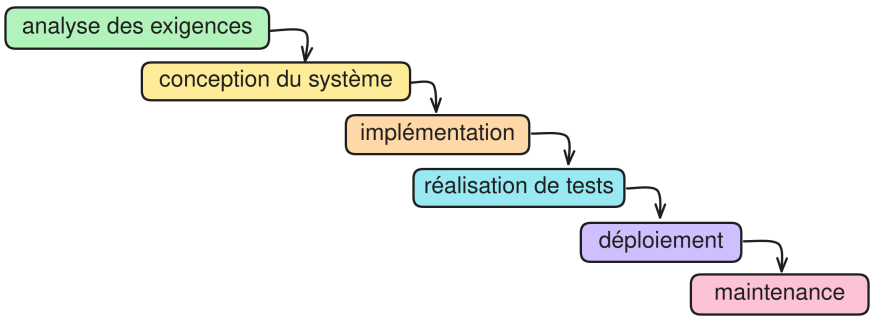


Figure 1.1 – Exemple de cycle de vie en cascade.

Le principal avantage de ce cycle de vie est qu'il est simple à comprendre. Chaque phase ayant un livrable spécifique, ce modèle permet d'avoir un système bien documenté et d'évaluer clairement les résultats. Par contre, comme le logiciel n'est disponible que tardivement dans le cycle de vie, ce dernier est très sensible aux risques et ne peut pas prendre en compte des changements dans les spécifications.

Le modèle en V (*V-Shaped Model*) [117] - Ce modèle est proche du modèle en cascade. Les phases sont également réalisées séquentiellement, chaque phase ne commençant qu'après la fin de la précédente. Dans ce modèle, la phase de test (*testing*) est mise en avant. En effet, à chaque fois qu'une tâche de développement est définie, une tâche de test correspondante est définie au même moment. Par exemple, lorsque les exigences sont définies, un ensemble de tests permettant de vérifier ultérieurement que le système les couvre est également défini.

Ce cycle est représenté sur la figure 1.2 et ses différentes phases sont :

— identification des exigences (*Requirements*) ;
— conception haut niveau du système (*High Level Design*) ;
— conception bas niveau du système (*Low Level Design*) ;
— implémentation (*Implementation*) ;
— réalisation de tests unitaires (*Unit Testing*) ;
— réalisation de tests d'intégration (*Integration Testing*) ;
— réalisation de tests système (*System Testing*).

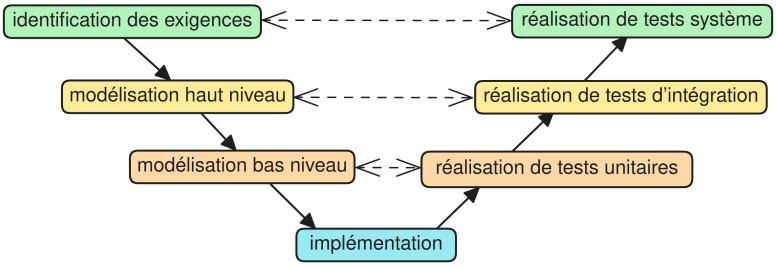


Figure 1.2 – Exemple de cycle de vie en V.

Ce cycle de vie a les avantages du cycle en cascade, mais il augmente les chances de réussite du développement grâce à la préparation des tests au plus tôt dans le cycle de vie. Malgré tout, il reste aussi rigide que le cycle de vie en cascade et il faut attendre la phase d'implémentation pour réellement tester le système.

Le modèle en spirale (*Spiral Model*) [152] - Bien que ce modèle suive une approche systématique, comme les deux modèles précédents, celui-ci est de nature itérative, incrémentale, et non séquentielle. En effet, après une première phase de définition des exigences, le système grandit de façon itérative en alternant les phases.

Ce cycle est représenté sur la figure 1.3 et ses différentes phases sont :

— **identification** des exigences (*Identification*) ;
— **conception** du système (*Design*) ;
— **contruction** du système, c'est à dire son implémentation (*Construction*) ;
— **évaluation** du système et des **risques** liés au développement (*Evaluation and Risks*).

Les risques peuvent être, par exemple, des risques impactant le planning ou le coût du développement.

Ce type de cycle de vie possède plusieurs avantages : des fonctionnalités peuvent être ajoutées au fil des itérations, il y a un fort contrôle de la documentation et le logiciel est produit rapidement. Par contre, ce cycle de vie dépend

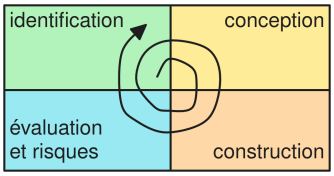


Figure 1.3 – Exemple de cycle de vie en spirale.

fortement de l'analyse des risques et nécessite donc une forte expertise dans ce domaine [148].

On peut remarquer que, quel que ce soit le cycle de vie considéré, le système existe à trois niveaux d'abstraction :

— un niveau Exigences (*Requirement Level* - RL dans la suite du manuscrit), correspondant aux phases liées aux exigences ;

— un niveau Modèle (*Model Level* - ML dans la suite du manuscrit), correspondant aux phases liées à la modélisation ; et

— un niveau Programme (*Program Level* - PL dans la suite du manuscrit), correspondant aux phases liées à l'implémentation.

Dans la suite du manuscrit je ferai référence à ces niveaux d'abstraction plutôt qu'aux étapes du cycle de vie. Aussi, je définis une nuance pour le niveau modèle, en considérant deux sous-niveaux :

— le niveau Modèle Graphique (ML Graphique) lorsque le système est modélisé à l'aide d'un language de modélisation graphique tel que BPMN [30] ;

— le niveau Modèle Formel (ML Formel) lorsque le système est modélisé à l'aide d'un language formel de modélisation tel que les algèbres de processus [14].

Le RGPD recommande de suivre le principe de protection des données dès la conception et par défaut (*data protection by design and by default*) afin d'augmenter la garantie que le système développé respecte la vie privée. Pour cela, il faut pouvoir vérifier que les systèmes la respectent à différentes étapes de développement. Ainsi, je propose une approche de vérification formelle qui peut être utilisée à la fois au niveau Modèle et au niveau Programme (ces niveaux étant plus adaptés pour une approche formelle). En effet, CSpeL permet de formaliser des systèmes à ces deux niveaux d'abstraction et CASTT peut être utilisé pour la vérification que ce soit aux niveaux Modèle ou Programme. Les propriétés ont été formalisées de façon à être indépendantes du niveau considéré.

En plus de ces méthodes de développement qui permettent de garantir la réalisation d'un logiciel de haute qualité, des types de méthodes de vérification on été proposés pour garantir le bon fonctionnement du système. Je présente ci-après les types de solution qui sont aujourd'hui les plus utilisés en pratique, et le positionnement de ma thèse par rapport à ceux-ci.

1.1.3 . Vérification et validation des systèmes

La vérification et la validation d'un système consistent à l'analyser et le tester pour déterminer s'il fonctionne correctement, c'est-à-dire s'il réalise les fonctionnalités attendues et s'il n'a pas de comportement non voulu. La vérification correspond à l'évaluation du logiciel durant les différentes phases de son cycle de vie afin d'assurer qu'il respecte les contraintes et les attentes définies durant la phase précédente. Quant à la validation, elle correspond à tester le logiciel à la fin du développement pour vérifier qu'il respecte les exigences [177]. Pham [134] résume cela par :

— Vérification : sommes nous en train de construire le produit correctement ?

— Validation : sommes nous en train de construire le bon produit ?

Dans cette thèse, je m'intéresse uniquement à la vérification et non à la validation. Il existe plusieurs techniques pour vérifier qu'un système est correct, j'en présente ci-dessous quelques unes.

Test (*testing*) - Un premier moyen de vérifier qu'un système est correct est de tester ses fonctionnalités. Les tests ont l'avantage d'être simples à mettre en place et permettent d'avoir des retours immédiats et simples à comprendre [35]. Cependant, ils ne permettent de détecter une erreur (un *bug*) qu'en observant ses conséquences (c'est-à-dire sous la forme d'une sortie non conforme ou même un *crash*) [153].

Vérification à l'exécution (*Runtime verification*)- La vérification à l'exécution est une méthode formelle d'analyse dynamique qui a pour but de vérifier qu'un système respecte des propriétés pendant son exécution [57]. Pour cela il existe deux possibilités : en observant le système à l'aide de moniteurs ou bien en analysant une trace d'exécution générée par le système. La première consiste à traduire une propriété en moniteur qui émet des verdicts selon le respect de la propriété. Le système est ensuite instrumenté pour permettre aux moniteurs d'effectuer les observations requisent du système. Ceux-ci analysent le comportement du système durant l'exécution. La seconde approche consiste à analyser des traces (généralement des journaux d'événements (*logs*)) correspondant à l'éxécution du système. Ces traces sont composées d'évènements survenus pendant l'exécution ou d'ensembles d'états représentant le comportement du système pendant l'exécution. Il existe une grande variété de formalismes pour les traces et les évènements selon les systèmes et les propriétés étudiés.

La vérification à l'exécution a plusieurs avantages : contrairement au test, elle permet de découvrir les erreurs en amont (dès qu'une assertion est *violée*) [153]. De même, le fait que les moniteurs puissent vérifier des propriétés complexes portant sur différents états internes d'un programme en cours d'exécution permet d'observer des erreurs qui sont indétectables par du test non monitoré. [36] non monitorés.

Pourtant, comme les tests, l'inconvénient principal de cette technique est qu'elle ne détecte que les erreurs de l'exécution courante. Elle ne peut donc pas prouver l'absence d'erreurs pour toutes les exécutions et ne vérifie donc le respect (ou le non-respect) de la propriété que pour l'exécution courante.

Vérification formelle statique (*Static formal verification*) - Il est également possible de vérifier formellement un système sans avoir besoin de l'exécuter. La vérification formelle statique consiste à s'assurer qu'un système respecte une notion de correction fonctionnelle précise à l'aide de techniques mathématiques [24] sans exécuter le programme. Plusieurs méthodes de vérification formelle statiques existent :

— la vérification déductive de logiciels (*Deductive software verification*) [68] : elle repose sur un processus de vérification fondé sur une certaine forme d'inférence logique (par exemple, sur l'application de règles d'inférence) ;

— l'interprétation abstraite (*Abstract interpretation*) [46] : elle consiste en l'utilisation de dénotations qui abstraient le comportement du système mais permettent de raisonner sur son comportement réel (par exemple, s'appuyant sur des intervalles de valeurs possibles au lieu d'un ensemble de valeurs précises) ;

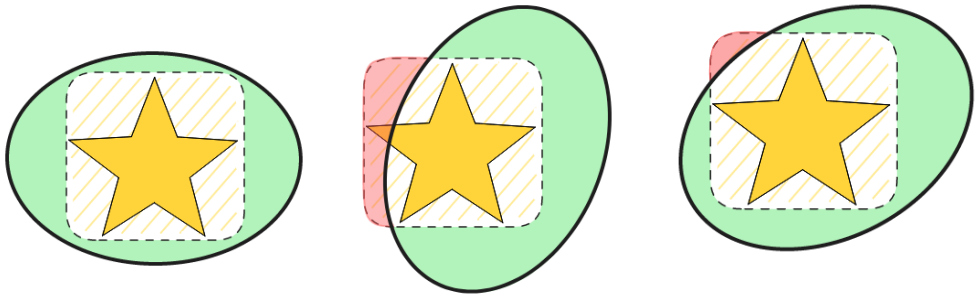
— la vérification de modèle (*Model checking*) [37] : elle consiste en la vérification d'un modèle du système, en tant que graphe de transition d'état, selon des spécifications généralement exprimées en logique temporelle.

Je donne ci-dessous un exemple pour chacune de ces méthodes de vérification.

$$\frac{\{P\}S_1\{R\} \qquad \{R\}S_2\{Q\}}{\{P\}S_1; S_2\{Q\}}$$

Figure 1.4 – Exemple de règle de preuve pour la vérification déductive de logiciels [68].

La figure 1.4 présente un exemple de règle d'inférence pour la vérification déductive de logiciels. Cette règle correspond à la règle pour la composition, lorsque l'exécution de l'instruction S_2 suit l'éxécution de S_1, avec comme précondition P et postcondition Q. Les précondition et postcondition décrivent le comportement d'un programme [68] sous forme de propriétés. Si la précondition est vraie avant l'exécution du programme, alors après l'exécution de celui-ci, la postcondition est vraie.



(a) propriété respectée

(b) propriété non respectée

(c) faux positif

Figure 1.5 – Exemple d'interprétation abstraite.

La figure 1.5 illustre le principe de l'interprétation abstraite. Le comportement du système est représenté par l'étoile jaune. L'abstraction du système est représenté par le rectangle quadrillé jaune, et l'ellipse verte correspond à l'espace dans lequel la propriété est respectée suivant une analyse par interprétation abstraite. La figure de gauche (a) illustre le cas où l'analyse considère, de façon correcte, que le système respecte la propriété. La figure du milieu (b) illustre le cas où elle considère, de façon correcte que le système ne respecte pas la propriété. Enfin, la figure de droite (c) illustre le cas où l'interprétation abstraite considère que la propriété n'est pas respectée, or c'est uniquement l'abstraction du système qui ne la respecte pas : le système en lui même la respecte. Il s'agit donc d'un faux positif lié aux approximations réalisées (qui peut nécessiter un complément d'analyse par une autre technique pour être levé).

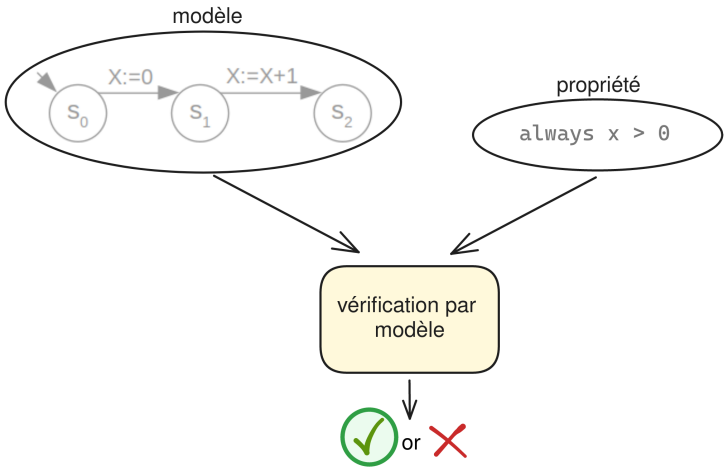


Figure 1.6 – Exemple de vérification de modèle.

La figure 1.6 présente un exemple d'utilisation de la vérification de modèle. Le système est représenté par un modèle, par exemple un automate. Une propriété à vérifier est formalisée, par exemple X toujours supérieur à 0. La vérification de modèle permet alors de vérifier si le modèle respecte la propriété.

Le principal avantage de ces vérifications formelles, tout comme la vérification à l'éxécution, est la grande confiance qu'apporte la preuve des propriétés. Un autre avantage de ces vérifications statiques est qu'il n'est pas nécessaire d'exécuter le programme [36]. Par contre ces méthodes présentent également des inconvénients :

— la vérification déductive peut nécessiter une aide manuelle pour réaliser la preuve, par exemple pour l'écriture d'invariants de boucle ;

— l'interpretation abstraite peut générer des faux positifs, c'est à dire que la détection de non respect d'une propriété peut être dûe à l'abstraction du système qui considère des états en réalité non atteignables ;

— la vérification de modèle peut mener à une explosion combinatoire.

Ainsi, chacune des méthodes a ses avantages et ces inconvénients, il n'y pas de meilleure solution absolue. En effet, cela dépend du contexte de vérification (niveau de garantie à apporter, temps de vérification ou encore selon la compléxité du programme et des propriétés). Ainsi mes contributions permettent de laisser plusieurs possibilités de vérifications formelles. CASTT permet de réaliser directement une vérification à l'exécution fondée sur l'utilisation de traces (pour une vérification de propriétés formelles sur des traces génériques) permettant de s'appliquer à différentes formalisations de systèmes. Il permet également de déléguer cette vérification à des outils de vérification formelle spécifiques au niveau Modèle [109]

ou au niveau Programme [19]. Ainsi cela permet de vérifier le système à l'aide des autres méthodes formelles citées ci-dessus.

Ces vérifications formelles utilisent des formalisations reposant sur des principes mathématiques, et en particulier logiques. Je présente ci-dessous quelques-unes de ces logiques, les plus connues, et le positionnement de ma thèse par rapport à celles-ci.

1.1.4 . Logiques pour raisonner sur des systèmes

Pour pouvoir raisonner formellement sur des systèmes, on peut utiliser différentes sortes de logiques, dont celles présentées ci-dessous.

Logique propositionnelle [81] - La logique propositionnelle repose sur la définition de formules composées de variables propositionnelles atomiques (par exemple p) et d'opérateurs tels que la conjonction ($\wedge$), la disjonction ($\vee$), la négation ($\neg$) et l'implication ($\rightarrow$). Les variables propositionnelles représentent des déclarations, par exemple « Albert Camus est un auteur français ». C'est l'une des logiques les plus simples mais son expressivité est limitée.

Logique des prédicats [81] - La logique des prédicats, appelée aussi logique du premier ordre, étend la logique propositionnelle en ne reposant pas sur des variables atomiques mais sur des prédicats. Un prédicat permet d'exprimer une propriété sur un ensemble de variables. Voici des exemples de prédicats, si on suppose que :

— $S(x)$: signifie que x est un étudiant inscrit ;

— $V(x, y)$: signifie que x a validé l'unité y ;

La logique des prédicats ajoute également deux opérateurs : « pour tout » ($\forall$) et « il existe » ($\exists$). On peut ainsi représenter par exemple que « tout étudiant qui a validé une unité est bien inscrit », ce qui s'écrit $\forall x.\forall y.V(x, y) \rightarrow S(x)$.

L'inconvénient est qu'en augmentant l'expressivité, prouver que ces formules sont valides devient un problème plus complexe.

Logique du second ordre [81] - La logique du second ordre est une extension de la logique des prédicats qui permet d'utiliser les opérateurs de quantification ($\forall$ et $\exists$) sur les variables mais également sur les prédicats. Par exemple, la formule $\exists P.\phi$ signifie qu'il existe un prédicat P pour lequel la formule ϕ est valide. Cette extension permet, par exemple, de spécifier des propriétés d'atteignabilité. Cette logique est donc plus expressive, mais les formules sont plus complexes à prouver.

Logiques temporelles [81] - Les logiques précédentes ne permettent pas d'exprimer simplement les notions de temps et d'évolution temporelle. D'autres logiques sont plus adaptées pour vérifier certaines propriétés liées aux systèmes réactifs et

concurrents. Pour cela, il existe des logiques temporelles (une sous-catégorie des logiques modales [81]) qui permettent de faire référence au futur (et au passé), telles que la *Linear-time Temporal Logic* (LTL), la *Computation Tree Logic* (CTL) et CTL*. Ces logiques ont l'avantage de permettre de représenter une évolution du temps par le parcours d'états. Par contre, elles sont plus complexes à vérifier que les logiques précédentes et nécessitent des méthodes et des outils de vérification pouvant considérer ces notions temporelles.

Pour pouvoir vérifier formellement qu'un système respecte la vie privée, il faut pouvoir formaliser des propriétés que le système doit respecter. Pour cela, j'ai proposé la propriété de *conformité aux finalités consenties* et la propriété de *conformité à la nécessité des données*. Ces propriétés sont exprimées en utilisant la logique des prédicats, qui s'est avérée suffisamment expressive dans mon cadre, tout en permettant une vérification totalement automatique de mes exemples.

1.2 . Contributions

Pour aider à améliorer le respect de la vie privée en informatique, plus particulièrement en proposant une solution qui permet de vérifier formellement qu'un système respecte des propriétés relatives à la vie privée, que cela soit au niveau Modèle ou au niveau Programme, j'ai réalisé les contributions suivantes, séparées en deux parties :

— partie classification :

- **représentations de la vie privée :** j'ai identifié dans la littérature différents types de représentations de la vie privée en informatique (une représentation correspondant au point de vue adopté par un article traitant de la vie privée) ;
- **GePyR, représentation générique de la vie privée :** j'ai défini une taxonomie générique fondée sur des catégories permettant de regrouper des notions similaires à travers les différentes représentations de la vie privée existantes ;
- **PyCO, ontologie du contexte :** j'ai défini une ontologie permettant de représenter les différents éléments, et leurs relations, à prendre en compte pour vérifier qu'un système respecte la vie privée. Pour cela, je me suis restreinte à un sous-ensemble des catégories définies dans GePyR ;

— partie vérification :

- **CSpeL, langage de spécification du contexte :** j'ai défini un langage permettant de formaliser un système et les éléments du contexte à prendre en compte pour vérifier le respect de la vie privée ;
- **propriétés de vie privée :** j'ai formalisé, en reprenant les notations de CSpeL, des propriétés de vie privée que doit respecter le système : une propriété de *conformité aux finalités consenties* et une propriété de *conformité à la nécessité des données* ;
- **CASTT, outil de vérification et de traduction :** j'ai implémenté un outil permettant de vérifier le respect de ces propriétés sur des traces exprimées en CSpeL. CASTT permet également de traduire le contexte spécifié en CSpeL, ainsi que les propriétés, dans des formalismes pour des outils aux niveaux Modèle et Programme.

Ces contributions et leurs relations sont illustrées sur la figure 1.7. L'identification des représentations de la vie privée en informatique se base sur des articles de la littérature proposant des façons de représenter la vie privée en informatique. À partir de ces représentations, j'ai identifié les notions communes entre elles et j'ai défini GePyR, permettant de classer les représentations identifiées dans des

catégories selon les notions de vie privée qu'elles abordent. J'ai ainsi pu classer des articles de la littérature proposant des solutions pour améliorer la vie privée. Un utilisateur peut ainsi sélectionner manuellement des articles s'intéressant à des notions similaires. Je me suis servi d'un sous-ensemble de ces catégories et des articles classés dans celui-ci pour identifier les éléments à prendre en compte et leurs relations afin de définir PyCO. J'ai classé des articles de la littérature afin d'identifier leur façon de représenter ces éléments. Un utilisateur peut ainsi sélectionner manuellement un ensemble d'articles et savoir de quelle façon ils représentent ces éléments.

Dans un second temps, pour la partie vérification, CSpeL reprend les éléments et leurs relations définies dans PyCO (en se restreignant à une seule catégorie de GePyR) pour définir un langage formel de spécification. L'utilisateur peut ainsi spécifier manuellement les éléments représentant le système (dans son contexte lié à la vie privée) dans un fichier. L'outil CASTT permet alors de vérifier des propriétés à partir de cette spécification. La formalisation de ces propriétés est fondée sur le langage CSpeL et les propriétés formalisées font référence aux éléments définis dans CSpeL.

Le lien entre les deux parties de thèse correspond au lien entre CSpeL et PyCO. La figure 1.7 ne représente pas les liens entre les contributions et l'état de l'art pour des raisons de lisibilité. Par exemple, pour définir PyCO, je me suis basée sur les catégories de GePyR mais également sur les articles de la littérature classés dans ces catégories.

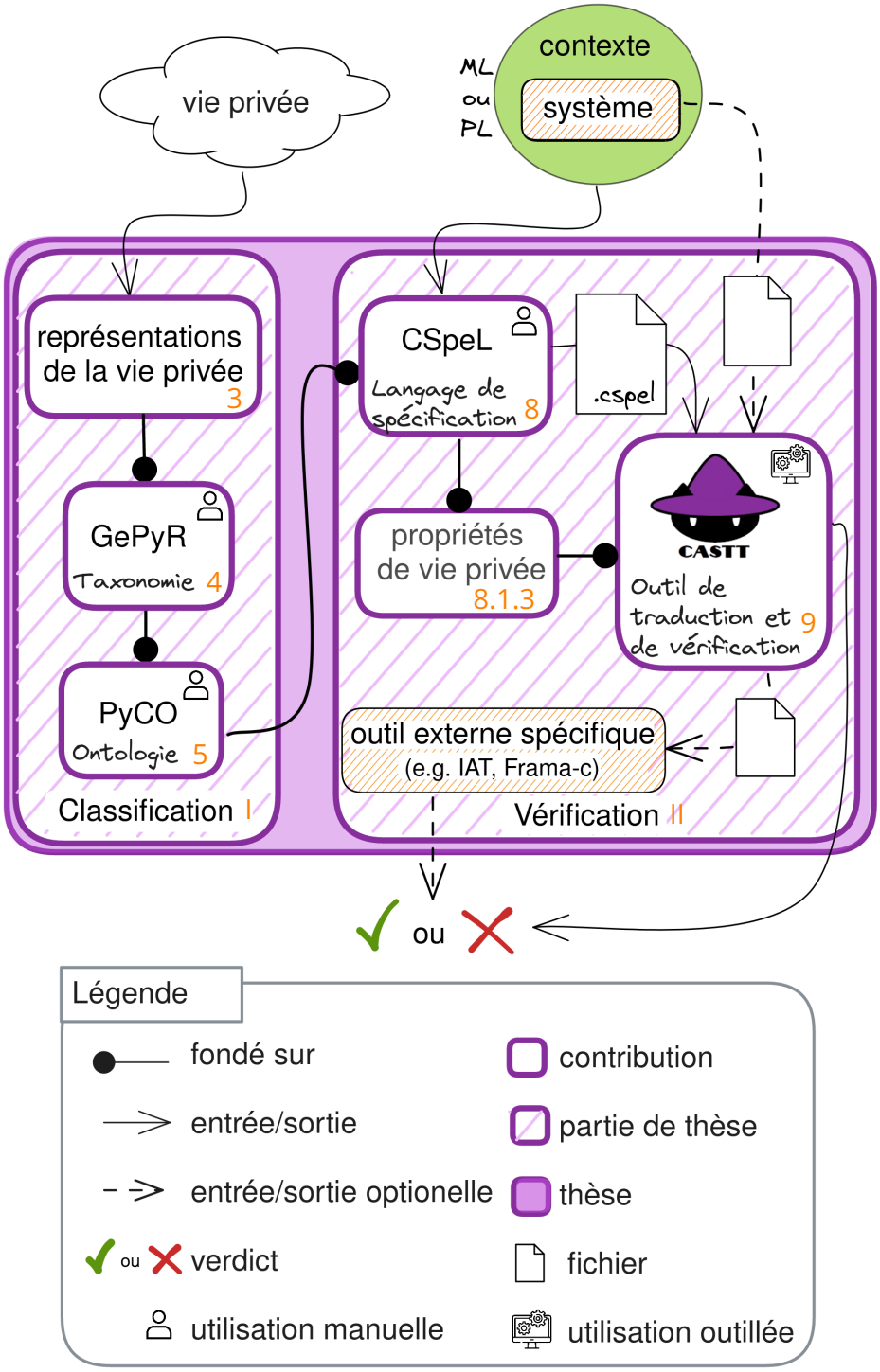


Figure 1.7 – Contributions de la thèse.

1.3 . Plan

Les explications des contributions de la partie classification se trouvent dans la Partie I. Celles-ci sont séparées en chapitres selon les contributions. Les représentations de la vie privée sont expliquées dans le chapitre 3. La taxonomie GePyR est définie et utilisée dans le chapitre 4. La définition et des exemples d'utilisation de l'ontologie PyCO sont fournis dans le chapitre 5.

Pour les contributions de la partie vérification, leur explications se trouvent dans la Partie II. De même, elles sont séparées selon les contributions. Celles pour le langage de spécification CSpeL se trouvent dans le chapitre 8. Les propriétés de vie privée que j'ai formalisées sont détaillées dans la section 8.1.3. Finalement, l'outil de vérification CASTT est présenté dans le chapitre 9, des explications sur son fonctionnement et des exemples y sont également disponibles.

1.4 . Dissemination des travaux

Les travaux concernant ces contributions ont été publiés et présentés à différentes occasions. Ils ont été aussi étendus entre ces disséminations et la rédaction du manuscrit. Ces extensions sont détaillées dans les chapitres suivants. Elle sont plus importantes pour GePyR et PyCO, car la publication présentant ces contributions est plus ancienne.

1.4.1 . Publications

— pour la partie classification, dans un atelier international (*workshop*) avec revue par les pairs :
Myriam Clouet et al. "A New Generic Representation for Modeling Privacy". In : *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 2022 ;

— pour la partie vérification, dans une conférence internationale avec revue par les pairs :
Myriam Clouet et al. "Context Specification Language for Formally Verifying Consent Properties on Models and Code". In : *TAP 2023 - 17th International Conference on Tests and Proofs*. 2023.

1.4.2 . Présentations

En plus des présentations dans les évènements ci-dessus, j'ai présentés mes travaux dans les évènements suivants :

— pour la partie classification :
- dans des séminaires internes au laboratoire :
  - ▶ Journée des thèses du Département d'Ingénierie des Logiciels et des Systèmes (DILS) au CEA LIST- 2020 et 2021.
- à l'extérieur :

  - ▶ IFIP Summer School on Privacy and Identity Management - 2021 ;
  - ▶ 11e Atelier sur la Protection de la Vie Privée (APVP'21) - 2021 ;

— pour la partie vérification :

- dans des séminaires internes au laboratoire :
  - ▶ Séminaire LSL au CEA List - 2022 ;
  - ▶ Journée des thèses du DILS au CEA List - 2022 et 2023.
- à l'extérieur :
  - ▶ Journée Génie Logiciel et Sécurité (commune aux GdR GPL et GdR Sécurité Informatique) - 2022 ;
  - ▶ Journée de séminaires commune des groupes de travail LVP et MTV2 (GdR GPL) - 2023.

1.4.3 . Code et expérimentations

Le code de CASTT et les expérimentations ont été mis à la disposition des relecteurs lors de la publication à la conférence Test and Proofs 2023 mais ils ne sont pas actuellement libres d'accès en raison de choix effectués par mon unité en terme de protection de la propriété intellectuelle.

1.5 . Guide de lecture du manuscrit

Pour aider à la lecture, une liste des abréviations est disponible au début du manuscrit. De même, la thèse est séparée en deux parties. Le lecteur peut choisir la partie qui l'intéresse le plus selon les domaines de recherche. La première partie concernant la classification correspond à de la modélisation (*via* une taxonomie, une ontologie et la réalisation d'une étude de cartographie systématique (*Systematic mapping study*)). La seconde partie concernant la vérification correspond à l'utilisation de méthodes formelles pour la vérification (*via* un langage de spécification, la formalisation de propriétés et l'utilisation d'outils).

Pour aider la lecture, les définitions sont englobées dans des blocs de définition comme celui ci-dessous. Le lecteur peut ainsi facilement les identifier.

Définition

Suivant le même principe, les exemples liés à ces définitions sont englobés dans des blocs d'exemple comme celui ci-dessous.

Exemple

De même, les théorèmes sont englobés dans des blocs comme celui ci-dessous :

Théorème

Enfin, les preuves de ces théorèmes sont englobées dans des blocs comme celui ci-dessous :

Preuve

Pour ne pas surcharger la lecture tout en permettant à ceux qui le souhaitent d'avoir des détails supplémentaires, des annexes sont disponibles en fin de manuscrit. Elles présentent des fichiers de spécification, des preuves, des scénarios de tests supplémentaires, ainsi que les fichiers de code utilisés dans les expérimentations.

Ce manuscrit contient beaucoup de figures et d'exemples, qui certes augmentent le nombre de pages du manuscrit, mais permettent d'illustrer les concepts et de faciliter la compréhension des notions.

La figure présentant les contributions et les liens entre elles (figure 1.7) est rappelée régulièrement dans le manuscrit afin de permettre au lecteur de se situer dans les chapitres des contributions qu'il aborde.

Les différences significatives entre les publications et le manuscrit sont explicitées dans des blocs comme celui ci dessous.

Première partie

CLASSIFICATION DE LA VIE PRIVÉE

Cette partie présente mes travaux concernant la classification de la vie privée en informatique. Elle présente les différentes contributions qui m'ont permis d'établir cette classification.

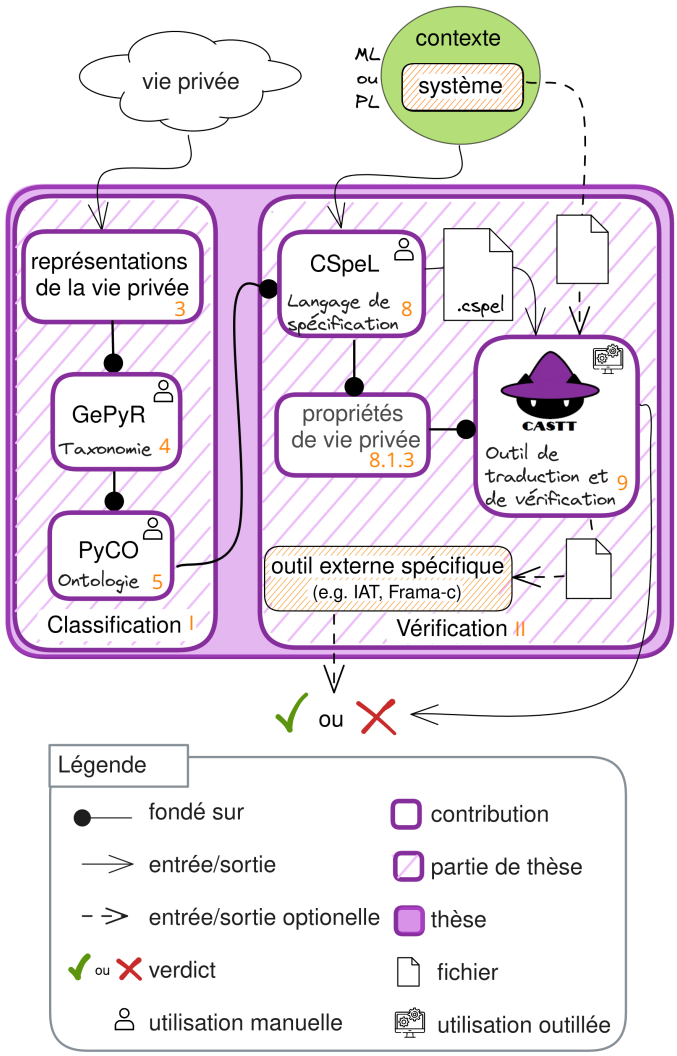


Figure 1.8 – Rappel - Lien entre les parties classification et vérification.

Comme rappelé sur la figure 1.8, mon approche est séparée en deux parties : la partie classification reposant sur les contributions GePyR et PyCO d'une part, et la partie vérification reposant sur CSpeL et CASTT d'autre part. Le but de mon approche est de vérifier formellement qu'un système respecte la vie privée au niveau Modèle et au niveau Programme. Cela pose la question de définir ce qu'est la vie privée en informatique et comment la respecter. Dans la littérature, un grand nombre d'articles traitent du respect de la vie privée, mais ils n'ont pas de structure

commune. Chacun adopte un point de vue propre ou une façon particulière de représenter la vie privée, et il n'y a pas forcément de liens évidents entre ces différentes représentations. Ainsi, il est difficile de classer les différentes approches afin d'identifier les solutions comparables, c'est-à-dire des solutions répondant à des problématiques similaires.

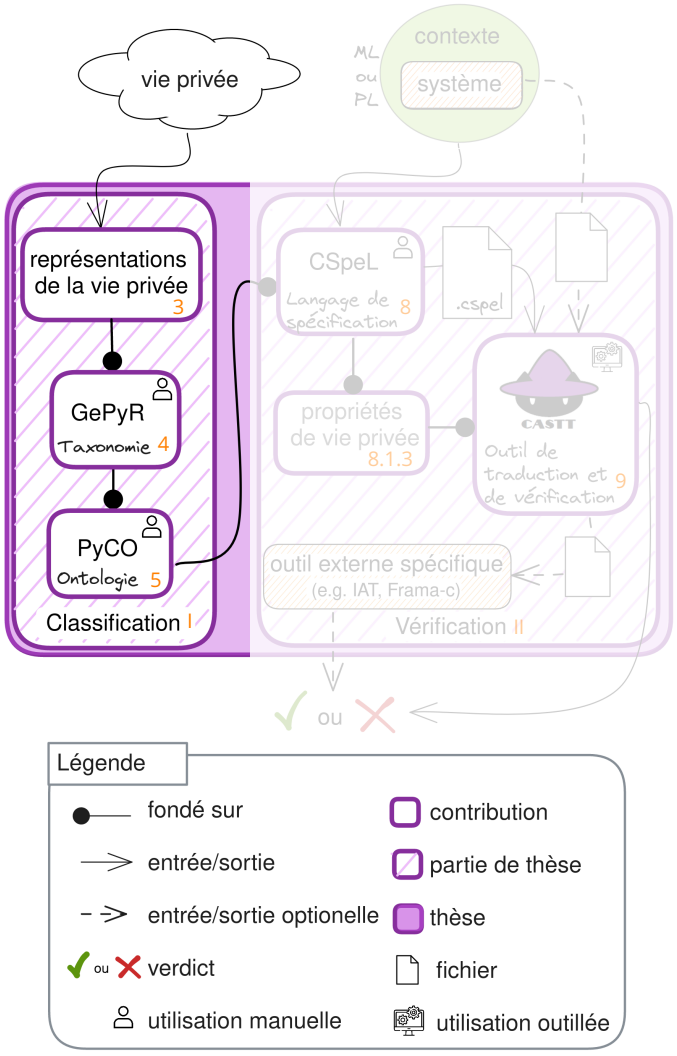


Figure 1.9 – Contributions de la partie classification.

Pour répondre à ce besoin, comme présenté sur la figure 1.9, j'ai réalisé les contributions suivantes :

— une identification des représentations de la vie privée dans la littérature.

— à partir de ces représentations, une taxonomie générique, GePyR, afin d'offrir une approche homogène face à la complexité venant de la diversité des

représentations. Par ailleurs, cette approche permet de faciliter la comparaison des solutions proposées afin d'améliorer le respect de la vie privée. GePyR permet également d'identifier les différentes notions à prendre en compte pour vérifier le respect de la vie privée en informatique.

— à partir d'un sous-ensemble de ces notions, une ontologie du contexte, PyCO, afin de représenter les éléments à prendre en compte pour vérifier le respect de la vie privée. PyCO permet ainsi de comparer la représentation de ces éléments dans différents articles de la littérature.

La section 2 présente le cas d'utilisation permettant d'illustrer mes contributions. Ensuite, la section 3 détaille les différentes représentations de la vie privée en informatique que j'ai identifiées dans la littérature. Puis, la section 4 présente GePyR, la taxonomie générique de la vie privée que j'ai définie. Finalement, la section 5 présente PyCO, mon ontologie du contexte de la vie privée.

2 - Cas d'utilisation

Cette section présente l'exemple utilisé pour illustrer mon approche de classification. Cet exemple présuppose que la classification existe déjà. Ce cas d'usage n'est pas relié à un cas réel identifié, il a seulement pour but d'illustrer comment la classification pourrait être exploitée concrètement.

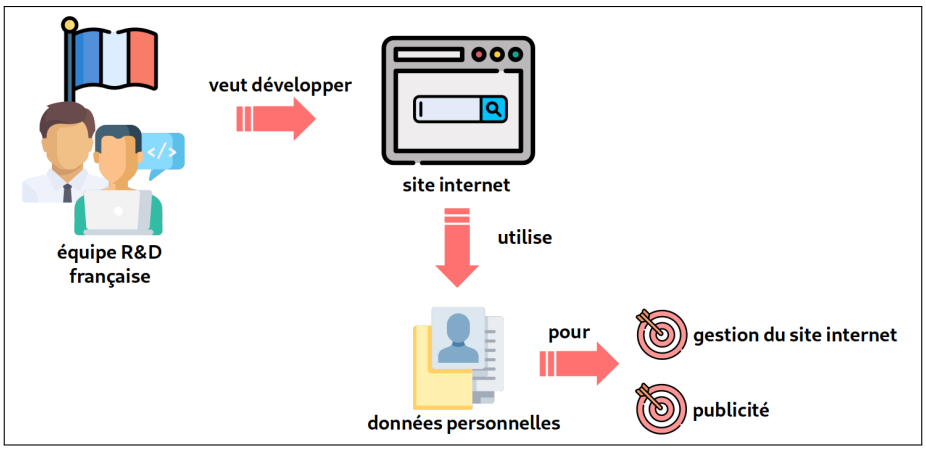


Figure 2.1 – Classification de la vie privée - Cas d'utilisation.

Comme présenté sur la figure 2.1, considérons une entreprise française qui veut développer un nouveau site internet. Ce site web a besoin de traiter des données personnelles à des fins de gestion du site (*website administration*) mais également à des fins de *marketing*. Par exemple, le site web utilise les adresses e-mail des utilisateurs pour les informer lorsque leur abonnement arrive à expiration (donc à des fins de gestion du site), mais également pour leur envoyer de la publicité ciblée. Or, le traitement de données personnelles doit respecter la vie privée de la personne concernée, ici l'utilisateur. Dans l'Union Européenne, le RGPD [55] a été mis en place pour permettre d'encadrer ces traitements. Il est obligatoire de le respecter, sous peine de sanctions, par exemple des amendes [1]. De plus, il incombe au responsable de traitement (ou à son sous-traitant s'il en a un) de mettre en place les mesures adéquates pour que les traitements des données personnelles qui sont effectués satisfassent les exigences de respect de la vie privée. Ainsi, l'équipe de développement veut pouvoir s'appuyer sur des solutions existantes dans l'état de l'art pour pouvoir mettre en place de telles mesures. Néanmoins, il existe une grande variété de articles avec des approches diverses, donc il est difficile pour

1. https://www.lemonde.fr/pixels/article/2021/09/02/donnees-personnelles-whatsapp-sanctionne-d-une-amende-record-par-le-regulateur-irlandais_6093152_4408996.html

elle d'identifier quelles solutions s'adressent à des problématiques proches et de les comparer. À ce stade du cycle de vie, le système n'est pas encore défini précisément (ni même modélisé ou implémenté) : ce travail de recherche et d'analyse de l'état de l'art est fait en amont. Ainsi, je ne détaillerai pas davantage dans cet exemple les processus et les données utilisées par le site web, ni son principe de fonctionnement. Un exemple de système du même domaine applicatif (sites internet) est détaillé plus précisément en sections 4.3 et 5.2 car il sert à évaluer mes contributions pour permettre de vérifier si un système est conforme à un ensemble de propriétés relatives au respect de la vie privée.

Même si je n'illustrerai mes travaux que sur cet exemple dans la suite du manuscrit, il est à noter que cela n'est qu'un exemple d'utilisation ; d'autres sont possibles. Il est ainsi possible, par exemple, utiliser ma classification pour positionner rapidement et de façon pertinente une nouvelle approche par rapport à l'état de l'art (*related work*). Une autre utilisation possible serait d'identifier les pistes de recherches non encore couvertes.

J'illustrerai dans la section 4.3, comment GePyR peut aider l'entreprise de cet exemple à identifier et sélectionner rapidement des solutions pertinentes En section 5.2, j'illustrerai comment PyCO peut les aider à identifier rapidement quels éléments sont à prendre en compte, et de quelle manière, pour appliquer la solution choisie.

Dans un premier temps, avant de présenter ma taxonomie (en section 4) et mon ontologie (en section 5.2) permettant de classifier la littérature, je présente dans la section suivante (section 3) les différentes représentations de la vie privée que j'ai identifiées dans l'état de l'art et sur lesquelles je me suis appuyé pour définir mes contributions.

3 - Représentations de la vie privée

Cette section présente les représentations de la vie privée qui seront utilisées pour ma classification. Ces représentations sont issues de documents de la littérature présentant des définitions de vie privée. Ces documents peuvent être par exemple des documents légaux [55], des taxonomies [159, 11], des cadres de référence (*frameworks*) [74, 50] ou encore des terminologies [133]. Le point commun entre ces documents est qu'ils fournissent une façon de représenter la vie privée par le biais d'éléments de mêmes natures (par exemple, menaces, propriétés, ...). Ces représentations peuvent être utilisées pour proposer une solution afin d'améliorer le respect de la vie privée. Par exemple la solution proposée par Sun, Guo, Zhu, et Feng suit le principe de *Data Breach Notification* (« Le responsable du traitement des données est tenu d'émettre une notification chaque fois qu'une violation de données à caractère personnel se produit ») [163], tandis que la solution proposée par Hirschi, Baelde et Delaune vérifie qu'un système respecte la propriété de *Unlinkability* (« vise à garantir qu'un utilisateur peut faire plusieurs utilisations d'un service ou d'une ressource sans que d'autres puissent relier ces utilisations entre elles ») [71].

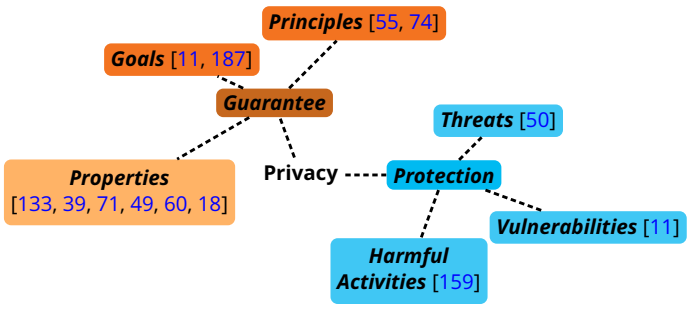


Figure 3.1 – Carte mentale des représentations de la vie privée.

Ma contribution, ici, consiste à regrouper ces représentations et à les organiser d'une façon cohérente pour ma classification. Je fournis leurs définitions telles que définies dans la littérature, des détails et des explications supplémentaires sont disponibles dans les documents cités. La plupart des définitions suivantes proviennent directement de ces documents (traduites en français), certaines ont cependant été réécrites afin de garder une certaine cohérence dans la présentation (par exemple, modifier une phrase au passif en une phrase à l'actif), tout en restant le plus proche possible de la signification de la définition originale. Celles qui ont été réécrites sont indiquées par le symbole $^{(*)}$. La liste des définitions originales et leur liens avec la version réécrite sont disponibles en Annexe A.6. Les définitions exactes peuvent également être trouvées dans les documents d'origine. Je divise

ces définitions en deux ensembles : les représentations orientées protection de la vie privée et celles orientées garantie du respect de la vie privée. Ces représentations sont résumées sur la figure 3.1. Cette figure est laissée en anglais, car elle fait référence à de la terminologie de la littérature (qui est en anglais).

D'abord, la section 3.1 présente les définitions des représentations orientées protection. Puis, la section 3.2 présente les définitions orientées garantie. Enfin, la section 3.3 consiste en une discussion concernant ces définitions.

3.1 . Représentations orientées protection (*protection*)

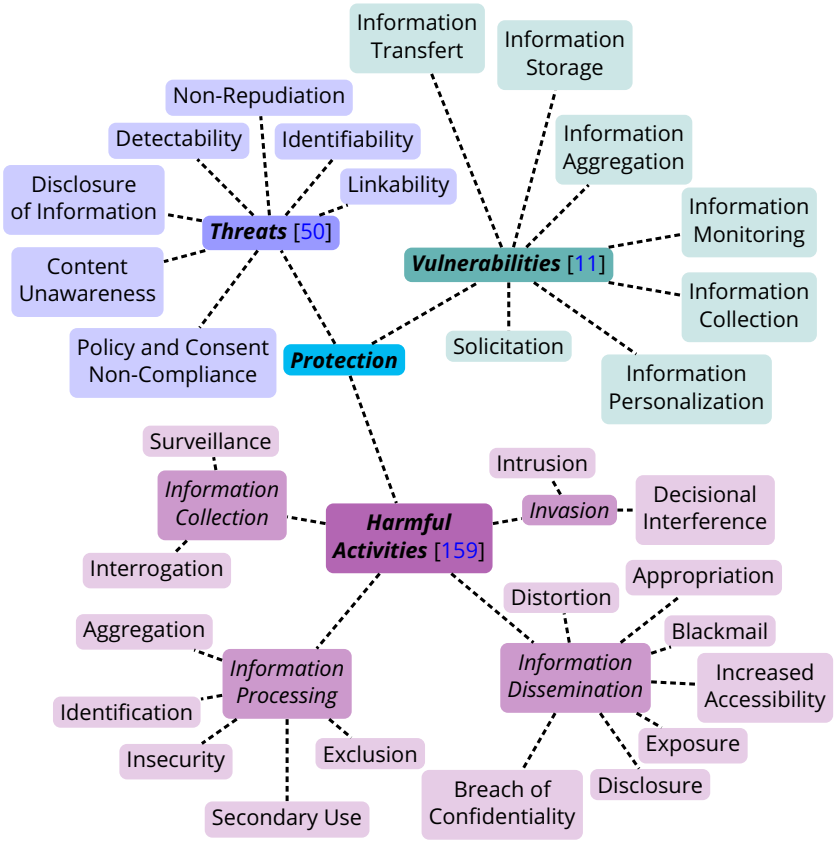


Figure 3.2 – Carte mentale des représentations orientées protection.

Certains documents proposent de représenter la vie privée par le biais d'un ensemble d'évènements contre lesquels le système a besoin d'être protégé. Selon cette vision, la vie privée peut être représentée par le biais d'un ensemble de menaces (*threats*), de vulnérabilités (*vulnerabilities*), ou d'activités nocives (*harmful activities*). Cette section décrit chacune de ces représentations en détail. Ces représentations sont résumées sur la figure 3.2. Cette figure est laissée en anglais,

car elle fait référence à de la terminologie de la littérature (qui est en anglais).

3.1.1 . Représentation par les menaces (*threats*)

Deng, Wuyts, Scandariato, Preneel et Joosen définissent un ensemble de menaces caractérisant la vie privée [50] :

- ***Linkability*** d'au moins deux éléments d'intérêt (*items of interest*, ou IOIs, par exemple des sujets, des messages, des actions, etc.), permet à un attaquant de distinguer suffisamment quand ces IOIs sont reliés ou non dans le système(*) ;
- ***Identifiability*** d'un sujet : signifie que l'attaquant peut suffisamment identifier le sujet associé à un IOI (par exemple, l'émetteur un message) ;
- ***Non-Repudiation*** : permet à un attaquant de rassembler des preuves pour contrer les affirmations partie qui répudie (*repudiating party*), et de prouver qu'un utilisateur sait, a fait, ou a dit quelque chose ;
- ***Detectability*** d'un IOI : signifie qu'un attaquant peut suffisamment distinguer si un tel élément existe ou non. Si nous considérons des messages comme IOIs, cela signifie que les messages sont suffisamment distinguables des bruits aléatoires ;
- ***Disclosure of Information*** : expose des informations personnelles des individus à ceux qui ne sont pas supposés y avoir accès ;
- ***Content Unawareness*** : indique qu'un utilisateur n'est pas conscient qu'une information a été divulguée au système ;
- ***Policy and Consent Non-Compliance*** : signifie que, même si le système affiche ses politiques de respect de la vie privée à ses utilisateurs, il n'y a aucune garantie que le système se conforme réellement aux politiques annoncées.

Pour ces menaces, les auteurs définissent la méthodologie LINDDUN pour modéliser les menaces contre la vie privée dans des logiciels.

3.1.2 . Représentation par les vulnérabilités (*vulnerabilities*)

Anton et Earp identifient un ensemble de vulnérabilités d'un système qui peut impacter le respect de la vie privée, se concentrant sur les sites internet [11] :

- ***Information Monitoring*** : reflète l'existence d'un suivi des informations par les organisations lorsque les clients visitent leur site web(*) ;
- ***Information Aggregation*** : se réfère à la combinaison d'informations personnelles identifiables (*Personally Identifiable Information*, ou PII) collectées précédemment avec des données provenant d'autres sources(*) ;

- ***Information Storage*** : reflète quels enregistrements sont stockés dans la base de données d'une organisation et comment ils le sont(*) ;
- ***Information Transfer*** : reflète la pratique consistant à autoriser la transmission d'informations, les raisons pour lesquelles elles peuvent être transférées, et à qui elles le sont(*) ;
- ***Information Collection*** : reflète quelles informations sont collectées par les sites web ;
- ***Information Personalization*** : reflète l'adaptation ou la personnalisation d'un site web à un visiteur spécifique, affectant ainsi la fonctionnalité ou le contenu offert aux visiteurs (*) ;
- ***Solicitation*** : reflète comment et dans quel but les organisations contactent les visiteurs ou d'autres personnes(*).

Les auteurs identifient ces vulnérabilités afin d'aider les développeurs de site internet à les réduire et à identifier les conflits avec les politiques de vie privée (*privacy policies*). Leur but est également de permettre aux clients d'évaluer et de comprendre ces politiques.

3.1.3 . Représentation par les activités nocives (*harmful activities*)

Solove identifie des activités nocives pour le respect de la vie privée [159] et les tries en différents groupes : *Information Collection*, *Information Processing*, *Information Dissemination* et *Invasion*. Ces activités sont :

- ***Information Collection*** :
  - ***Surveillance*** : est l'action de regarder, écouter, ou enregistrer les activités d'un individu(*) ;
  - ***Interrogation*** : consiste en des formes variées de questionnement ou de recherche d'informations ;
- ***Information Processing*** :
  - ***Aggregation*** : implique la combinaison de divers éléments d'information sur une personne ;
  - ***Identification*** : consiste à relier des informations aux individus(*) ;
  - ***Insecurity*** : implique la négligence dans la protection des informations stockées contre les fuites et l'accès inapproprié ;
  - ***Secondary Use*** : est l'utilisation d'informations collectées dans un but différent de celui prévu, sans le consentement de la personne concernée ;

  - ***Exclusion*** : concerne le fait de ne pas permettre à la personne concernée de connaître les données que d'autres détiennent à son sujet et de participer à leur traitement et à leur utilisation.

- ***Information Dissemination*** :
  - ***Breach of Confidentiality*** : consiste à rompre la promesse de garder les informations d'une personne confidentielles ;
  - ***Disclosure*** : implique la révélation d'informations véridiques sur une personne qui influencent la façon dont les autres jugent son caractère ;
  - ***Exposure*** : consiste à révéler la nudité, les problèmes ou les activités corporelles d'une autre personne ;
  - ***Increased Accessibility*** : implique d'amplifier l'accessibilité de l'information ;
  - ***Blackmail*** : est la menace de divulgation de données personnelles ;
  - ***Appropriation*** : implique l'utilisation de l'identité de la personne concernée pour servir les buts et les intérêts d'une autre ;
  - ***Distortion*** : consiste à diffuser des informations fausses ou trompeuses sur des personnes.
- ***Invasion*** :
  - ***Intrusion*** : concerne les actes invasifs qui dérangent la tranquillité ou la solitude de quelqu'un ;
  - ***Decisional Interference*** : implique l'ingérence du gouvernement dans les décisions de la personne concernée au regard de ses affaires privées.

Pour Solove, le but de sa taxonomie est de définir plus précisément les problématiques liées à la protection de la vie privée.

3.2 . Les représentations orientées garanties (*guarantee*)

Certains articles proposent de représenter la vie privée comme un ensemble de concepts qu'il est nécessaire de garantir. Ainsi, la vie privée peut être vue comme un ensemble de principes (*principles*), buts (*goals*) ou propriétés (*properties*). Ces représentations sont résumées sur la figure 3.3. Cette figure est laissée en anglais, car elle fait référence à de la terminologie de la littérature (qui est en anglais).

3.2.1 . Représentation par les principes (*principles*)

Dans le RGPD, la Commission Européenne définit des principes clés pour représenter la vie privée en la limitant à son aspect « protection des données » [55] :

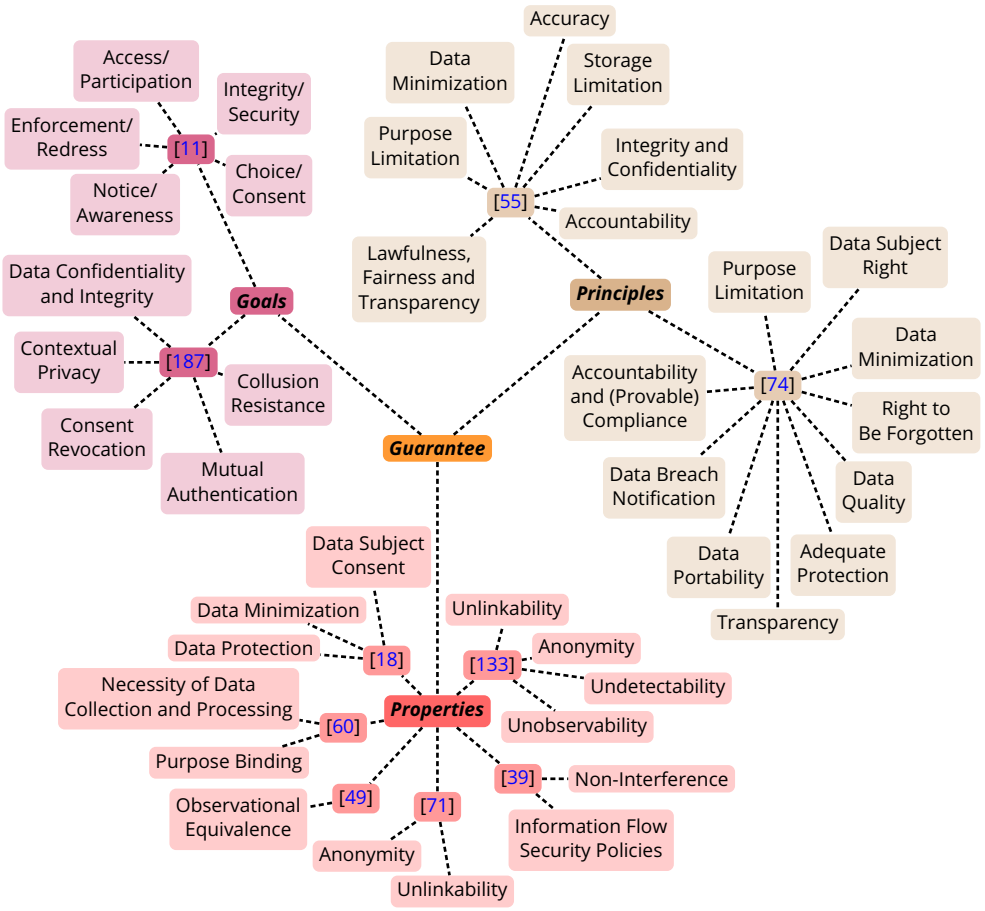


Figure 3.3 – Carte mentale des représentations orientées garanties.

— ***Lawfulness, Fairness and Transparency*** : les données à caractère personnel doivent être traitées de manière licite, loyale et transparente au regard de la personne concernée ;

— ***Purpose Limitation*** : les données à caractère personnel doivent être collectées pour des finalités déterminées, explicites et légitimes, et ne pas être traitées ultérieurement d'une manière incompatible avec ces finalités ; le traitement ultérieur à des fins archivistiques dans l'intérêt public, à des fins de recherche scientifique ou historique ou à des fins statistiques n'est pas considéré comme incompatible avec les finalités initiales ;

— ***Data Minimization*** : les données à caractère personnel doivent être adéquates, pertinentes et limitées à ce qui est nécessaire au regard des finalités pour lesquelles elles sont traitées ;

— ***Accuracy*** : les données à caractère personnel doivent être exactes et, si nécessaire, tenues à jour ; toutes les mesures raisonnables doivent être prises pour que les données à caractère personnel qui sont inexactes, eu égard aux finalités pour lesquelles elles sont traitées, soient effacées ou rectifiées sans tarder ;

— ***Storage Limitation*** : les données à caractère personnel doivent être conservées sous une forme permettant l'identification des personnes concernées pendant une durée n'excédant pas celle nécessaire au regard des finalités pour lesquelles elles sont traitées ; les données à caractère personnel peuvent être conservées pour des durées plus longues dans la mesure où elles seront traitées exclusivement à des fins archivistiques dans l'intérêt public, à des fins de recherche scientifique ou historique ou à des fins statistiques conformément à l'article 89, paragraphe 1, pour autant que soient mises en œuvre les mesures techniques et organisationnelles appropriées requises par le présent règlement afin de garantir les droits et libertés de la personne concernée ;

— ***Integrity and Confidentiality*** : les données à caractère personnel doivent être traitées de façon à garantir une sécurité appropriée des données à caractère personnel, y compris la protection contre le traitement non autorisé ou illicite et contre la perte, la destruction ou les dégâts d'origine accidentelle, à l'aide de mesures techniques ou organisationnelles appropriées ;

— ***Accountability*** : le responsable du traitement est responsable du respect des principes précédents et est en mesure de démontrer que celui-ci est respecté.

De plus, Hoepman définit un ensemble de principes légaux pour résumer les principes des lignes directrices (*guidelines*) de l'OCDE [136] et du standard ISO 29100 [82]. Cet ensemble est composé de plusieurs principes de protection des données [74] :

— ***Purpose Limitation*** : la finalité des données est précisée et l'utilisation des données est limitée à cette finalité$^{(*)}$;

— ***Data Subject Rights*** : les personnes concernées ont le droit de consulter, d'effacer et de rectifier leurs données à caractère personnel$^{(*)}$;

— ***Data Minimization*** : la collecte de données à caractère personnel doit être réduite au minimum, tant en ce qui concerne la quantité que le nombre d'acteurs qui y ont accès. Par défaut des options qui ne portent pas atteinte à la vie privée doivent être proposées et les données doivent être supprimées une fois qu'elles ne sont plus nécessaires$^{(*)}$;

— ***Right to Be Forgotten*** : les personnes concernées doivent avoir le droit de demander au responsable du traitement la rectification, l'effacement des

données à caractère personnel ou la limitation du traitement des données à caractère personnel les concernant, ou encore le droit de s'opposer à un tel traitement(*) ;

— ***Data Quality*** : les données à caractère personnel doivent être pertinentes au regard des finalités pour lesquelles elles sont utilisées, et être exactes, complètes et mises à jour ;

— ***Adequate Protection*** : les données à caractère personnel doivent être protégées de manière adéquate ;

— ***Transparency*** : la nature et l'étendue du traitement des données doivent être facilement accessibles, de même que l'identité du responsable du traitement(*) ;

— ***Data Portability*** : les personnes concernées ont le droit d'obtenir du responsable du traitement une copie des données faisant l'objet du traitement dans un format électronique et structuré, couramment utilisé et permettant leur utilisation ultérieure(*) ;

— ***Data Breach Notification*** : le responsable du traitement des données est tenu d'émettre une notification chaque fois qu'une violation de données à caractère personnel survient ;

— ***Accountability and (Provable) Compliance*** : le responsable du traitement des données est tenu de respecter ces principes et doit être en mesure de prouver qu'il respecte la règlementation(*).

À partir de ces principes, Hoepman dérive des stratégies de design pour définir des systèmes respectueux de la vie privée mais, également pour évaluer l'impact des systèmes existants.

3.2.2 . Représentation par les buts (*goals*)

En complément des vulnérabilités précédentes, Anton et Earp identifient un ensemble de buts requis pour assurer le respect de la vie privée [11] :

— ***Notice/Awareness*** : affirme que les consommateurs doivent être notifiés et/ou sensibilisés aux pratiques d'une organisation en matière d'information avant qu'une quelconque information ne soit effectivement collectée auprès d'eux ;

— ***Choice/Consent*** : garantie que les consommateurs ont la possibilité de décider de l'utilisation des informations personnelles collectées à leur sujet et si elles peuvent être utilisées à des fins secondaires ;

— ***Access/Participation*** : autorise ou restreint l'accès à un site ou à une fonctionnalité particulière selon que le consommateur fournit ou non ses informations personnelles identifiables ;

— ***Integrity/Security*** : garantit que les données sont à la fois exactes et sécurisées ;

— ***Enforcement/Redress*** : traite des mécanismes mis en place pour faire respecter la vie privée.

Zhang, Bacchus, et Lin proposent un autre ensemble de buts à atteindre pour respecter la vie privée [187] :

— ***Data Confidentiality and Integrity*** : garantit que les données ne peuvent pas être accédées par d'autres entités sans le consentement de l'utilisateur et que les données doivent être protégées des modifications(*) ;

— ***Contextual Privacy*** : permet aux demandeurs de données ("*data requesters*") de demander les informations sur la santé de l'utilisateur au centre de données ("*data center*") avec le consentement de l'utilisateur, alors que le fournisseur de données ("*data provider*") ne sait pas à qui appartiennent les informations demandées et garantit que les observateurs de la transmission des données ne peuvent pas déduire quelles informations demandées sont nécessaires au demandeur de données(*) ;

— ***Consent Revocation*** : affirme que les utilisateurs ont le privilège de mettre fin au consentement à tout moment avant son expiration(*) ;

— ***Mutual Authentication*** : affirme que le fournisseur de données et le centre de données sont en mesure de s'authentifier mutuellement, que l'utilisateur ne peut donner accès qu'à ses propres données et que le centre de données est en mesure d'authentifier le fournisseur de consentement(*) ;

— ***Collusion Resistance*** : permet au centre de données de re-chiffrer les données de l'utilisateur demandées par le demandeur après avoir reçu d'un utilisateur la clé de re-chiffrement destinée à un demandeur de données et affirme que le demandeur de données ne peut pas décrypter les autres données de l'utilisateur qui sont au-delà du consentement, même s'il est de connivence avec le centre de données(*).

Zhang, Bacchus, et Lin définissent un système qui atteint ces buts à l'aide d'un mécanisme de contrôle d'accès fondé sur le consentement.

3.2.3 . Représentation par les propriétés (*properties*)

Il existe de nombreuses définitions de propriétés pour le respect de la vie privée [133, 39, 71, 53, 49, 60, 18]. Pfitzmann et Hansen définissent un ensemble de propriétés pour représenter la vie privée, selon la perspective d'un attaquant [133] :

— ***Anonymity*** d'un sujet : signifie que l'attaquant ne peut pas suffisamment identifier le sujet dans un ensemble de sujets, l'ensemble d'*anonymity* ;

— ***Unlinkability*** de deux ou plus IOIs : signifie qu'au sein du système (comprenant ces éléments et éventuellement d'autres), l'attaquant ne peut pas suffisamment distinguer si ces IOIs sont liées entre eux ou non ;

— ***Undetectability*** d'un IOI : signifie que l'attaquant ne peut pas suffisamment distinguer si l'IOI existe ou non ;

— ***Unobservability*** d'un IOI : signifie l'*undetectability* du IOI pour tous les sujets qui ne participent pas à cet IOI et l'anonymat du ou des sujets impliqués dans l'IOI, même pour d'autres sujets impliqués dans ce IOI.

Dans plusieurs articles proposant des solutions pour assurer le respect de la vie privée, les définitions de vie privée viennent d'autres représentations de la vie privée, souvent spécifiques à certains contextes. Aussi, plusieurs articles définissent ou re-définissent des propriétés de vie privée. Clarkson et Schneider, par exemple, définissent des propriétés requérant une évaluation d'un ensemble de traces [39] :

— ***Information-Flow Security Policies*** : expriment des restrictions sur quelles informations peuvent être apprises par des utilisateurs sur un système ;

— ***Non-Interference*** : requiert que les commandes issues d'utilisateurs disposant d'une habilitation élevée puissent être retirées sans affecter les observations des utilisateurs disposant d'une habilitation faible.

Dufay, Felty et Matwin proposent également une définition alternative pour la *non-interference* [53] :

— ***Non-Interference*** : distingue les entrées (respectivement les sorties) publiques des entrées (respectivement des sorties) sensibles et empêche les fuites d'information des entrées sensibles vers les sorties publiques.

Par ailleurs, Hirschi, Baelde et Delaune [71] re-définissent les termes de Pfitzmann et Hansen [133] dans le cadre des protocoles de communication, comme :

— ***Anonymity*** : vise à garantir qu'un utilisateur puisse utiliser un service ou une ressource, sans révéler son identité. Un observateur extérieur ne devrait pas être en mesure de faire la différence entre le protocole original et une version du protocole dans laquelle l'attaquant sait que des rôles spécifiques attribués à certaines identités, connues par l'attaquant, sont présents ;

— ***Unlinkability*** : vise à garantir qu'un utilisateur peut faire plusieurs utilisations d'un service ou d'une ressource sans que d'autres puissent relier ces utilisations entre elles. Un protocole préserve l'*unlinkability* si deux sessions d'un même rôle semblent, pour une personne extérieure, avoir été exécutées avec des noms d'identité différents.

Delaune, Ryan et Smyth [49] se concentrent sur :

— ***Observational Equivalence*** : signifie que deux processus ne peuvent pas être distinguables par un quelconque attaquant.(*)

Fischer-Hubner et Ott définissent des propriétés centrées sur les données [60] :

— ***Purpose Binding*** : interdit que des données à caractère personnel obtenues pour une finalité donnée soient utilisées pour une autre finalité sans le consentement éclairé de la personne concernée(*) ;

— ***Necessity of Data Collection and Processing*** : n'autorise la collecte et le traitement de données à caractère personnel que lorsqu'ils sont nécessaires à l'accomplissement de tâches relevant de la responsabilité de l'agence de traitement des données(*).

Finalement, Barati, Rana, Petri et Theodorakopoulos définissent des propriétés centrées sur les données, fondées sur le RGPD [18] :

— ***Data Subject Consent*** : exige que toute utilisation de données confirmée par le responsable de traitement soit assortie d'une politique de confidentialité explicite et que la personne concernée y consente également ;

— ***Data Minimization*** : indique que les activités au sein des processus d'entreprise ne doivent pas recevoir ou collecter des données qui ne sont pas utilisées pour le traitement ;

— ***Data Protection*** : exige le chiffrement des données pour toute activité de traitement des données à caractère personnel dans le cadre d'un relation commerciale(*).

3.3 . Discussions

Les sections précédentes ont présenté les définitions de représentations de la vie privée que j'ai trouvé dans la littérature. Cependant, certains termes ont été introduits plusieurs fois et d'autres sont liés entre eux. Cela est discuté dans cette section.

— **Linkability :** la *Linkability* est définie comme une menace par Deng et al. [50], mais d'autres auteurs introduisent le terme opposé, l'*Unlinkability* [133, 71]. Ainsi, la définition de Pfitzmann et al. [133] est l'exacte négation de la définition de la *Linkability* introduite par Deng et al.. Même si la définition introduite par Hirschi et al. [71] reste proche de celle de Pfitzmann et al., elle est pourtant plus spécifique en termes de "services" et de "session" dans les protocoles.

— **Detectability :** la définition de l'*Undetectability* de Pfitzmann et al. [133] correspond à la négation de la définition de la *Detectability* par Deng et al. [50].

— **Disclosure :** la *Disclosure* est définie soit comme une menace [50], soit comme une activité nocive [159] même si leurs deux notions sont proches : la première est une menace qui expose des informations personnelles, tandis que la seconde est une activité nocive qui révèle des informations personnelles. Cependant, elles ne sont pas totalement équivalentes : la première prend en compte les entités qui ont le droit d'accéder aux informations, tandis que la seconde fait référence au jugement des autres personnes.

— **Information Monitoring :** la vulnérabilité *information monitoring* introduite par Anton et Earp [11] est une instantiation de l'activité nocive *Surveillance* définie par Solove [159]. En effet, l'*Information Monitoring* consiste en l'existence d'un suivi des informations par les organisations lorsque les consommateurs visitent un site web$^{(*)}$, tandis que, la *Surveillance* se réfère à l'action de regarder, écouter ou enregistrer les activités d'un individu$^{(*)}$.

— **Information Aggregation :** à la fois la vulnérabilité *Information Aggregation* [11] et l'activité nocive *Aggregation* [159] font référence à la combinaison de données personnelles.

— **Information Collection :** Anton et Earp définissent la vulnérabilité *Information Collection*, spécifique aux sites internets. Solove utilise le même terme pour désigner une catégorie d'activités nocives qui inclut la *Surveillance* et l'*Interrogation*.

— **Transparency :** Hoepman définit le principe *Transparency* [74], qui est une spécification du principe *Lawfulness, Fairness and Transparency* du RGPD [55]. La *Transparency* de Hoepman désigne ainsi le fait que la nature et l'étendue du traitement des données doivent être facilement accessibles, de même que le responsable du traitement$^{(*)}$, tandis que le principe de *Lawfulness, Fairness and Transparency* du RGPD correspond au fait que « Les données à caractère personnel doivent être traitées de manière licite, loyale et transparente au regard de la personne concernée ».

— **Purpose Limitation :** Hoepman [74] et le RGPD [55] définissent le même terme *Purpose limitation* comme principe, mais leur définitions sont légèrement différentes. Les deux concernent la limitation de l'utilisation des données pour une finalité spécifique mais le RGPD fournit une définition plus précise. Par exemple, le RGPD définit un cas spécifique pour lequel la finalité n'est pas considérée comme incompatible avec la finalité associée au traitement de la donnée.

— **Data Minimization :** la *Data minimization* est définie comme un principe [55, 74] et une autre fois comme une propriété [18]. Les points de vue des principes et des propriétés sont différents : un principe doit être suivi (par exemple, par la mise en place d'une méthodologie) et une propriété doit être vérifiée (par exemple, par de la vérification formelle). Cependant, leur définitions sont en réalité très proches : même si la définition de Hoepman [74]

est la plus précise, toutes les trois se fondent sur la notion d'une utilisation requérant le strict minimum de données. Ces représentations sont également proches de la propriété *Necessity of Data Collection and Processing* définie par Fischer-Hubner et Ott [60], comme le fait que la collecte et le traitement de données à caractère personnel ne sont autorisés que lorsqu'ils sont nécessaires à l'accomplissement de tâches relevant de la responsabilité de l'agence de traitement des données$^{(*)}$.

— **Data Protection :** la propriété *Data Protection* définie par Barati et al. [18] est liée au principe *Privacy by Design* du RGPD [55]. Cependant, Hoepman utilise le même terme, *Data Protection*, pour représenter l'ensemble de ses principes (*Purpose Limitation*, *Data Subject Rights*, *Data Minimization*, *Right to Be Forgotten*, *Data Quality*, *Adequate Protection*, *Transparency*, *Data Portability*, *Data Breach Notification* et *Accountability and (Provable) Compliance*) [74]. La *data minimization* est donc, pour Hoepman, un principe de *data protection*, tandis que Barati et al. les considèrent comme deux propriétés différentes.

— **Integrity and Confidentiality :** le principe *Integrity and Confidentiality* du RGPD [55] est lié au but *Integrity/Security* d'Anton et Earp [11] et au but *Data Confidentiality and Integrity* de Zhang et al. [187]. Cependant, la définition du RGPD est plus précise. En effet, elle précise certains traitements pour lesquelles il faut protéger les données : « Les données à caractère personnel doivent être traitées de façon à garantir une sécurité appropriée des données à caractère personnel, y compris la protection contre le traitement non autorisé ou illicite et contre la perte, la destruction ou les dégâts d'origine accidentelle, à l'aide de mesures techniques ou organisationnelles appropriées ». *Integrity/Security* étant « garantit que les données sont à la fois exactes et sécurisées », et *Data Confidentiality and Integrity* garantit que les données ne peuvent pas être accédées par d'autres entités sans le consentement de l'utilisateur et que les données doivent être protégées des modifications$^{(*)}$.

— **Accountability :** le principe d'*Accountability* du RGPD [55] est proche du principe *Accountability and (Provable) Compliance* de Hoepman [74]. Les termes utilisés diffèrent légèrement mais leur signification est similaire : le responsable de traitement doit démontrer qu'il respecte les principes de la régulation (RGPD pour le RGPD, l'OCDE pour Hoepman).

— **Notice/Awareness :** étant à première vue similaires, le but *Notice/Awareness* d'Anton et Earp [11] et la menace *Content Unawareness* de Deng et al. [50] ne font pas référence à des concepts similaires. En effet, le premier fait référence aux informations sur les pratiques d'une organisation, tandis que la seconde fait référence à la méconnaissance liée à la sujet de la divulgation de l'information.

— **Consent :** trois représentations comportent le terme *Consent*, à savoir le but *Choice/Consent* d'Anton et Earp [11], le but *Consent Revocation* de Zhang et al. [187], et la propriété *Data Subject Consent* de Barati et al. [18]. Même si tout les trois se fondent sur une notion de consentement, ils sont quelque peu différents : le but *Choice/Consent* fait référence à la collection d'informations et aux finalités secondaires, le but *Consent Revocation* goal fait référence à la fin du consentement, tandis que la propriété *Data Subject Consent* fait référence au responsable de traitement, à la politique de respect de la vie privée définie et à la personne concernée.

— **Anonymity :** les deux définitions pour la propriété *Anonymity* par Pfitzmann et al. [133], et par Hirschi et al. [71] sont similaires, même si la seconde est plus spécifique en faisant référence à l'« utilisateur », au « service » et au « protocole ».

— **Non-Interference :** Clarkson et Schneider [39] et Dufay et al. [53] définissent chacun une propriété *Non-Interference*. Pourtant elles sont différentes : celle de Clarkson et Schneider fait référence aux « commandes », à l'« habilitation » et aux « observations », tandis que celle de Dufay et al. fait référence aux entrées/sorties publiques, aux entrées/sorties sensibles et aux « fuites ».

Pour permettre la classification d'articles de la littérature, en plus de ces différentes représentations spécifiques, dans la suite de la thèse, je considère les définitions suivante pour les types de représentation identifiés précédemment :

Définition 1: Types de représentations

— **Attaque** : action qui fait apparaître intentionnellement une mauvaise chose ;

— **Principe** (*Principle*) : ligne de conduite ou règle à suivre ;

— **But** (*Goal*) : objectif à atteindre ;

— **Propriété** (*Property*) : caractérisation du système à respecter ;

— **Vulnérabilité** (*Vulnerability*) : faiblesse du système qui permet une attaque ;

— **Menace** (*Threat*) : potentielle action ou évènement facilité par une vulnérabilité qui a un impact indésirable sur le système ;

— **Activité nocive** (*Harmful activity*) : action qui peut constituer ou mener à une menace.

Bien que l'attaque ne soit pas un type représenté précédemment, dans la définition 1, sa définition est donnée car elle est utilisée dans d'autres. À noter que ces définitions peuvent être appliquées à des domaines autres que la vie privée en

informatique, comme le montre l'exemple suivant.

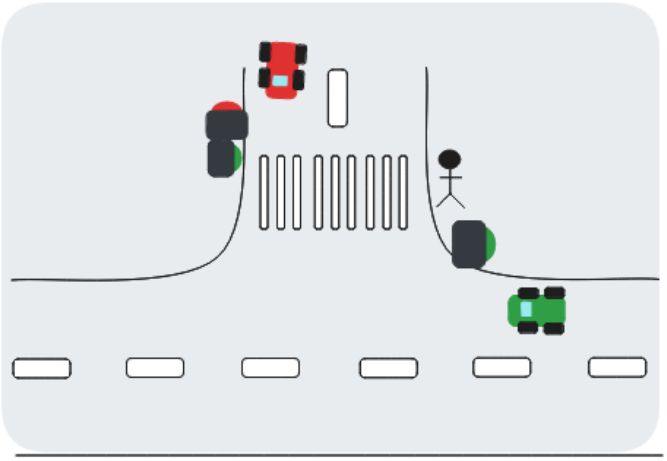


Figure 3.4 – Schéma de la traversée d'un piéton à un carrefour.

Exemple 1: Instanciation des types de représentation

Cet exemple instancie les types de représentations de la définition 1 sur un exemple d'un domaine différent de la vie privée en informatique. Le choix d'un domaine qui ne soit pas en lien avec la vie privée est volontaire, cela permet d'illustrer sur une situation courante les différences parfois subtiles entre les différentes représentations. Le système considéré est la traversée d'un piéton à un carrefour. Ce système est illustré sur la figure 3.4.
L'instanciation des représentations dans ce contexte est la suivante :

- **Principe** : le piéton est prudent pour traverser ;
- **But** : le piéton est sain et sauf de l'autre côté de la rue ;
- **Propriété** : le piéton traverse quand le feu piéton est vert ;
- **Vulnérabilité** : le feu piéton n'est pas coordonné avec un feu pour les voitures ;
- **Menace** : une voiture renverse le piéton ;
- **Activité nocive** : une voiture traverse le passage piéton.

Les types de représentations présentées ici correspondent à celles identifiées dans [40], leurs définitions et une discussion ont été ajoutées. La représentation *via* les dimensions a été retirée car après analyse, elle n'était pas pertinente. En effet, comme expliqué dans [40], cette représentation n'est pas composée d'éléments homogènes, ces dimensions correspondant à des axes composés de points discrets (*discrete points*).

Cette section a présenté les représentations de la vie privée qui seront utilisées pour ma classification. Ces représentations sont issues de divers documents de la littérature présentant des définitions de vie privée. D'abord, j'ai donné les définitions des représentations orientées protection (section 3.1). Ensuite, celles des représentations orientées garantie (section 3.2). Enfin, la section 3.3 présente quelques réflexions sur les similarités et les différences entre certaines de ces représentations. On peut ainsi remarquer qu'il existe une grande variété de représentations qui, pourtant, peuvent faire référence à des notions similaires, mais considérées sous différents points de vues.

Le chapitre suivant présente la représentation générique GePyR que j'ai définie afin de regrouper au sein de mêmes catégories des représentations faisant référence à des notions similaires. Un tel travail d'homogénéisation est nécessaire pour permettre la comparaison d'articles proposant des solutions pour améliorer le respect de la vie privée.

4 - Représentation Générique de la Vie Privée - GePyR

Cette section présente GePyR, une nouvelle représentation de la vie privée, construite comme un système de classification reposant sur des catégories liées à la vie privée. Le but de GePyR est d'offrir une approche homogène face à la complexité venant de la diversité des représentations et de faciliter la comparaison de solutions proposées afin d'améliorer le respect de la vie privée.

Comme expliqué dans la section précédente (section 3), j'ai identifié dans la littérature six représentations liées à la vie privée, qui la considèrent comme un ensemble de : menaces, vulnérabilités, activités nocives, principes, buts, et propriétés. GePyR englobe les différentes notions utilisées dans ces représentations.

Dans un premier temps, la section 4.1 définit les catégories de GePyR, puis la section 4.2 explique comment ces catégories peuvent être spécialisées afin de retrouver les concepts des représentations existantes. Ensuite la section 4.3 illustre l'utilisation de GePyR sur le cas d'utilisation défini dans la section 2, et finalement la section 4.4 présente une utilisation plus générale, pour classer des articles de la littérature.

4.1 . Définition des catégories

GePyR est un système de classification reposant sur des catégories permettant de regrouper des solutions permettant d'améliorer le respect de la vie privée afin d'identifier des solutions comparables.

J'ai défini cinq catégories pour GePyR, qui sont les **caractérisations** :

— ***du Traitement*** (*Processing characterization*) ;

— ***de la Visibilité*** (*Visibility characterization*) ;

— ***de la Transparence*** (*Transparency characterisation*) ;

— ***de l'Exactitude des données*** (*Data Accuracy characterization*) ; et

— ***de la Responsabilité*** (*Accountability characterization*).

Ces catégories étant génériques, elles permettent de couvrir les notions présentes dans les représentations de la vie privée en informatique, de la section 3. Lorsque cela est nécessaire (par exemple, lorsque le cas d'utilisation considère une représentation précise), ces catégories peuvent être spécialisées selon ces représentations. Par exemple, si l'on considère une représentation sous forme de principe, la catégorie *caractérisation du Traitement* pourra être spécialisée sous la forme du principe *Purpose Limitation* [55], tandis que si l'on considère une représentation sous forme de propriétés, elle pourra être spécialisée comme la propriété *Necessity of Data Collection and Processing* [60]. Ces spécialisations sont détaillées en section 4.2. Je

considère que les catégories *caractérisation du Traitement* et *caractérisation de la Visibilité* sont les catégories principales de GePyR, car elles permettent de couvrir le plus de représentations de la vie privée, comme cela sera détaillé en section 4.2.

Les définitions de ces catégories sont les suivantes :

Définition 2: *Caractérisation du Traitement*

La *caractérisation du Traitement* fait référence à toute notion liée au traitement autorisé des données personnelles.

Cela signifie que l'on va classer dans cette catégorie toutes les représentations qui ont un lien avec le traitement des données personnelles. La signification que je considère pour le terme « traitement » est large : cela peut correspondre aussi bien à l'utilisation qu'au stockage de données. À noter que les notions liées aux autres catégories (visibilité, transparence, exactitude et responsabilité) ne sont pas incluses dans celle-ci.

Dans cette catégorie, j'ai pu également définir 4 sous-catégories, liées :

— aux finalités du traitement (*Purpose*) ;

— à la nécessité des données (pour le traitement) (*Necessity*) ;

— à la durée du traitement (*Retention Duration*) ;

— à l'évolution de l'accord pour le traitement (*Consent Evolution*).

Ainsi, par exemple, le principe de *Purpose limitation* du RGPD [55] est classée dans cette catégorie car elle fait référence aux finalités des **traitements**. De même pour la propriété de *Necessity of Data Collection and Processing* de Fischer-Hubner et Ott [60] car elle fait référence au fait que le **traitement** des données personnelles (et leur collection) est nécessaire.

Je n'ai identifié des sous-catégories que pour cette catégorie, car c'est celle qui a fait l'objet des travaux de la seconde partie de ma thèse, la partie vérification de propriétés de respect de la vie privée, notamment dans le but de définir des propriétés formelles liées au traitement (dans le cadre de la vie privée). Ces propriétés de traitement (*processing-related properties*) sont détaillées en section 8.1.3.

Définition 3: *Caractérisation de la Visibilité*

La *caractérisation de la Visibilité* fait référence à toute notion liée à la visibilité des données personnelles restreinte à un ensemble de personnes ou d'organisations autorisées.

Cela signifie que l'on va classer dans cette catégorie toutes les représentations qui ont un lien avec la lecture des données personnelles mais également avec leur accès, leur divulgation, leur connaissance, etc.

Ainsi, par exemple, la menace *Disclosure of Information* de Deng et al. [50] est classée dans cette catégorie car elle fait référence à l'accès des données personnelles (et à leur divulgation) par ceux qui n'y sont pas autorisés.

Définition 4: *Caractérisation de la Transparence*

La *caractérisation de la Transparence* fait référence à toute notion liée à la conscience qu'a la personne concernée par les données personnelles de l'utilisation qui en est faite.

Cela signifie que l'on va classer dans cette catégorie toutes les représentations qui ont un lien avec la présentation d'informations à la personne concernée, tels que les documents concernant les notices explicatives, les langages tels que P3P (https://www.w3.org/TR/P3P/), ou même les bannières de cookies.

Définition 5: *Caractérisation de l'Exactitude des données*

La *caractérisation de l'Exactitude des données* fait référence à toute notion liée à l'exactitude des données personnelles traitées.

Par exemple, on va classer dans cette catégorie les représentations qui ont un lien avec la possibilité de corriger les données personnelles, mais également celles qui interdisent les modifications non autorisées.

Définition 6: *Caractérisation de la Responsabilité*

La *caractérisation de la Responsabilité* fait référence à toute notion liée à l'obligation qu'ont les responsables du traitement et leurs sous-traitants de démontrer qu'ils respectent la vie privée.

Par exemple, le principe de *Accountability* du RGPD [55] est classé dans cette catégorie mais également le but de *Enforcement/Redress* de Anton et Earp [11]. En effet ce but fait référence aux mécanismes mis en place pour faire **respecter la vie privée**.

Cette notion dépend fortement de la représentation de la vie privée considérée, car la démonstration du respect de la vie privée dépend de comment celle-ci est représentée. Dans GePyR, comme illustré sur la figure 4.1, la représentation de la vie privée correspond à des catégories. Donc pour démontrer le respect de la vie privée, il faut démontrer le respect des catégories (la *caractérisation du Traitement*, la *caractérisation de la Visibilité*, la *caractérisation de la Transparence* et la *caractérisation de l'Exactitude des données*). Donc la *caractérisation de la*

Responsabilité dépend de mes autres categories.

GePyR permet de regrouper plusieurs représentations de la vie privée dans cinq catégories. J'ai fait mon possible pour être exhaustive, cependant il est possible que d'autres représentations ne puissent pas rentrer dans ces catégories. GePyR est néanmoins suffisamment flexible pour être étendue facilement afin de les prendre en compte. Je présente les spécialisations dans la section suivante.

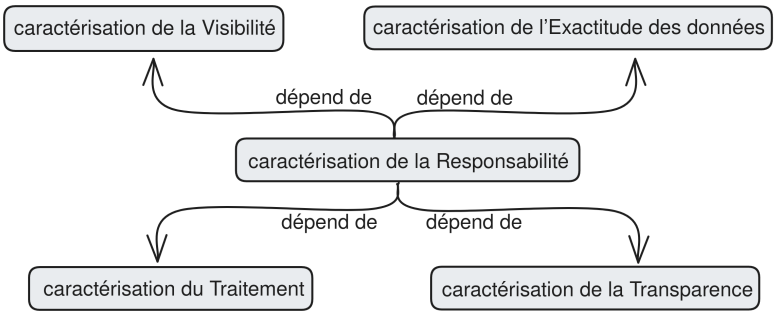


Figure 4.1 – Dependences entre les catégories de GePyR.

Par rapport à l'article [40], j'ai ajouté le terme « caractérisation » pour mettre l'accent sur le fait que l'on s'intéresse à ce qui caractérise les notions des différentes catégories. J'ai renommé la catégorie *Confidentiality* (confidentialité) en « visibilité » pour qu'elle reflète mieux sa définition. J'ai également renommé la catégorie *Consent* (consentement) en « traitement » et j'ai changé sa définition pour mieux correspondre aux notions des représentations spécifiques classées dans cette catégorie (par exemple, le principe de *Data minimisation*). J'ai ajouté une nouvelle catégorie, la *caractérisation de l'Exactitude des données*, pour regrouper certaines notions qui étaient présentes dans les représentations spécifiques mais qui n'étaient pas explicitées dans les catégories de GePyR dans l'article [40].

4.2 . Spécialisations des catégories

Les catégories de GePyR peuvent être spécialisées de différentes façons. Ces spécialisations dépendent du contexte dans lequel les catégories sont utilisées et du point de vue à partir duquel elles sont considérées.

Le tableau 4.1 présente les spécialisations de mes catégories selon les représentations de la vie privée présentées en section 3. Ce tableau est laissé en anglais, car il fait référence à de la terminologie de la littérature (qui est en anglais), ainsi les catégories de GePyR sont également mises en anglais pour la cohérence. Les cellules dans le tableau sont différemment remplies selon les spécialisations. Chaque

Table 4.1 – Spécialisations des catégories de GePyR.

	Processing characterisation	Visibility characterisation	Transparency characterisation	Data Accuracy characterisation	Accountability characterisation
Threats	Policy and Consent Non-Compliance [50]	Linkability [50]	Content Unawareness [50]		
		Identifiability [50]			
		Detectability [50]			
		Disclosure of Information [50]			
Vulnerabilities	Information Monitoring [11]	Information Aggregation [11]	Solicitation [11]		
	Information Collection [11]	Information Transfer [11]			
	Information Storage [11]				
	Information Personalization [11]				
Harmful Activities	Interrogation [159]	Identification [159]	Exclusion [159]	Distortion [159]	
	Secondary Use [159]	Breach of Confidentiality [159]			
		Disclosure [159]			
		Exposure [159]			
		Increased Accessibility [159]			
		Blackmail [159]			
Principles	**Lawfulness, Fairness and** Transparency [55]	Integrity and **Confidentiality** [55]	Lawfulness, Fairness and **Transparency** [55]	**Integrity** and Confidentiality [55]	Accountability [55]
	Purpose Limitation [55, 181, 74]	Adequate protection [74]	Transparency [74]	Accuracy [55]	Accountability and (Provable) Compliance [74]
	Storage Limitation [55]	Data Breach Notification [74]		Data Subject Rights [74]	
	Data Minimization [55, 74]			Right to Be Forgotten [74]	
				Data Quality [74]	
Goals	Choice/Consent [11]	Integrity/**Security** [11]	Notice/ Awareness [11]	**Integrity**/ Security [11]	Enforcement/ Redress [11]
	Consent Revocation [187]	**Data Confidentiality** and Integrity [187]		Data Confidentiality and **Integrity** [187]	
	Mutual Authentication [187]				
	Collusion Resistance [187]				
Properties	Purpose Binding [60]	Anonymity [133, 71]			
	Necessity of Data Collection and Processing [60]	Unlinkability [133, 71]			
	Data Subject Consent [18]	Undetectability [133]			
	Data Minimization [18]	Unobservability [133]			
		Information-Flow Security Policies [39]			
		Non-Interference [39, 53]			
		Observational Equivalence [49]			
		Data Protection [18]			

catégorie de GePyR peut être spécialisée dans au moins une représentation de l'état de l'art, et chaque représentation est couverte par au moins une de mes catégories. De plus, ce tableau permet d'illustrer le fait que les catégories principales de GePyR, *caractérisation du Traitement* et *caractérisation de la Visibilité*, peuvent être spécialisées dans toutes les représentations de l'état de l'art. Pour spécialiser mes catégories dans les représentations de l'état de l'art, je me suis basée sur leur définitions, détaillées en section 3. Le but est de trouver les notions en lien avec ces définitions qui correspondent aux définitions de mes catégories, définies en section 4.1.

Exemple 2: Spécialisation en *Purpose Limitation*

Par exemple, je considère que le principe de limitation des finalités (*Purpose Limitation*) du RGPD défini comme "Les données à caractère personnel doivent être collectées pour des finalités déterminées, explicites et légitimes, et ne pas être traitées ultérieurement d'une manière incompatible avec ces finalités ;[. . .]" [55] spécialise la catégorie de GePyR ***caractérisation du Traitement***, définie comme "fait référence à toute notion liée au traitement autorisé des données personnelles", car leurs définitions font référence à des notions liées au traitement des données personnelles afin qu'il soit autorisé.

La table de spécialisation diffère légèrement de celle publiée dans [40], pour suivre l'évolution des catégories de GePyR. De même, l'analyse des définitions des représentations spécifiques a été réévaluée pour consolider cette classification. Il peut arriver que la définition d'une représentation spécifique contient plusieurs parties et que ces parties correspondent à des catégories différentes de GePyR. Le cas échéant, je dédouble la représentation spécifique et je mets en évidence la partie correspondant à la catégorie en la mettant en gras dans le tableau.

Cette section a présenté les différentes spécialisations possibles pour les catégories de GePyR selon les représentations de la vie privée, détaillées en section 3, en suivant les définitions des catégories faites en section 4.1. La section suivante illustre l'utilisation de GePyR sur l'exemple de cas d'utilisation données en section 2.

4.3 . Illustration sur le cas d'usage

Reprenons le cas d'utilisation, présenté en section 2, d'une entreprise française qui veut développer un nouveau site internet traitant les données personnelles des utilisateurs, pour différentes finalités. L'entreprise veut mettre en place des mesures pour assurer que son traitement respecte la vie privée, en s'appuyant sur des solutions existantes dans la littérature. Pour cela, elle suit le processus illustré sur

la figure 4.2. À l'aide de GePyR, les articles de la littérature sont classés dans des sous-ensembles, selon les notions de vie privée ciblées par leurs approches. Cette classification pourrait, par exemple, être faite par un fournisseur spécialisé de bases de connaissance de telle sorte que l'entreprise utilisatrice se contente de chercher ce qui l'intéresse dans le corpus déjà construit. Considérons dans un premier temps, que l'entreprise veut étudier les solutions concernant le traitement des données personnelles. De plus, étant française, elle doit respecter le RGPD, donc elle s'intéresse plus particulièrement aux solutions pour respecter les principes de vie privée. Elle sélectionne ainsi le sous-ensemble d'articles correspondant. Cela lui permet de n'étudier que des solutions concernant la problématique qu'elle souhaite cibler.

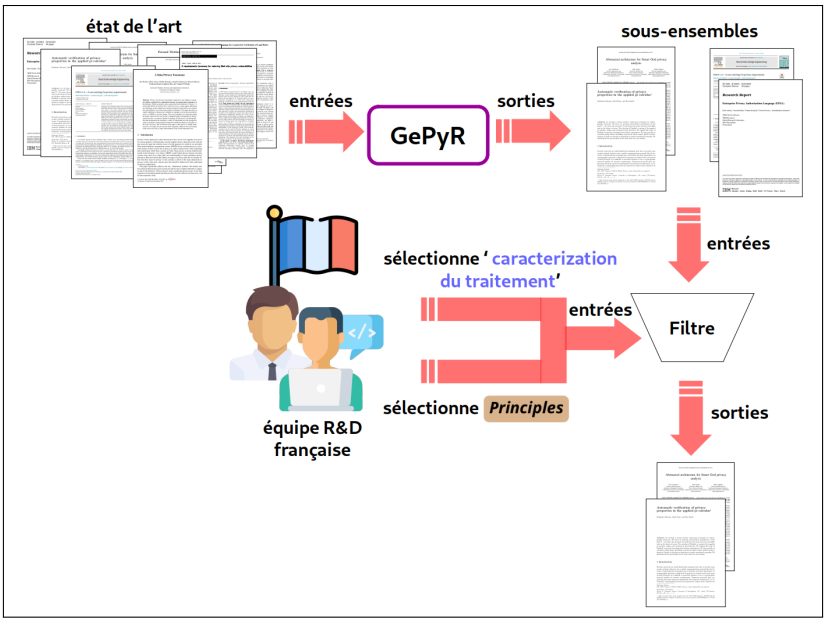


Figure 4.2 – Illustration d'utilisation de GePyR sur le cas d'usage.

4.4 . Classification d'articles de la littérature à l'aide de GePyR

Cette section présente la classification de la littérature que j'ai réalisée à l'aide de GePyR, et des représentations de la vie privée. Cette classification prend la forme d'une étude de cartographie systématique (*Systematic Mapping Study* ou SMS). Une telle étude est une revue systématique (*Systematic Review* - SR) ayant un sujet d'étude très large, ce qui implique également des questions de recherche plus larges et plus abstraites [91]. Elle a pour but de structurer un domaine de recherche [131] à l'aide d'une d'un schéma de classification (*classification scheme*), *via* des catégories [129]. Elle permet ainsi de mettre en évidence la couverture de champs de recherche selon la fréquence de publication dans les catégories de classification [129], ce qui permet de découvrir les tendances et les manques (*gap*)

dans un domaine de recherche [131].

Pour ma méthodologie j'ai suivi les recommandations définies dans la littérature [91, 131]. Ainsi, ma SMS est composée de trois phases : la planification (*planning*), la réalisation (*conducting*) et le rapport (*reporting*). Ces recommandations sont largement appliqués dans la littérature dès lors qu'il s'agit d'effectuer une SMS ou bien une SR [131, 115, 48, 66, 54, 5, 116, 83].

La phase de planification est détaillée en section 4.4.1, la phase de réalisation en section 4.4.2, et finalement la section 4.4.3 détaille la phase de rapport.

4.4.1 . Planification de la *Systematic Mapping Study*

Lors de la phase de planification, le besoin de la SMS est identifié, en précisant son champ d'application et son but principal. Les questions de recherche (*Research Questions*, RQs) sont définies, ainsi que le protocole de recherche. À la fin de cette phase, le protocole est évalué avant de passer à la phase suivante, qui est la phase de réalisation [91, 131].

a) Besoin d'une SMS sur le respect de la vie privée

Comme l'ont observé Alnajran et al. [5], dans le domaine du *mobile cloud computing*, il y a un nombre considérable d'articles publiés concernant des solutions pour améliorer le respect de la vie privée, en conséquence de quoi les résultats de ces articles ne sont pas suffisamment agrégés. Une SMS s'avère donc nécessaire pour identifier les tendances et les manques dans les solutions proposées. Cela est également vrai lorsque l'on s'intéresse au domaine plus large du respect de la vie privée en informatique.

Ainsi, le domaine d'application de cette SMS est le respect de la vie privée en informatique. Ce domaine concerne des notions très diverses et varie selon les pays et les populations [159]. De plus, il s'applique à des domaines d'applications variés (par exemple, les assistants vocaux [43], les sites internets [94] ou bien encore les systèmes informatiques des hôpitaux [132]). Cette SMS n'est donc pas restreinte à un domaine d'application unique et est appliquée à une notion de respect de la vie privée assez générique.

Le but de cette SMS est d'avoir une vue d'ensemble des articles existants proposant des solutions pour améliorer le respect de la vie privée en informatique en les structurant à l'aide d'une classification pour identifier des solutions comparables, et ainsi découvrir les tendances et manques de ce domaine [131].

b) Questions de recherche

Les SMS ont des questions de recherches avec des sujets plus larges que celles des *systematic reviews* [91], et permettent d'atteindre le but défini précédemment, à savoir identifier des solutions comparables, et ainsi découvrir les tendances et manques de ce domaine. Les questions de recherche que j'ai sélectionnées sont :

— **RQ1 :** Quelle est la couverture des notions liées à la vie privée ?

— **RQ2 :** Quelles sont les représentations de la vie privée qui sont les plus considérées et les moins considérées ?
— **RQ3 :** Quel type de représentation est le plus utilisé selon les notions considérées ?
— **RQ4 :** Existe-t-il beaucoup d'outils pour les informaticiens pour les aider à respecter la vie privée ?
— **RQ5 :** Quelle est l'évolution des publications sur l'amélioration du respect de la vie privée ?

c) Protocole de recherche

L'établissement du protocole de recherche nécessite la définition d'une stratégie de recherche, de critères de sélection et d'un schéma de classification.

Stratégie de recherche – La stratégie de recherche correspond au choix des mots clés, de la requête pour la recherche d'articles, et de la base de donnée à interroger. Comme recommandé par Kitchenham et Charters [92], j'utilise la méthodologie PICO (*Population*, *Intervention*, *Comparison*, *Outcome*), pour réaliser une sélection de mot clés (*keywords*) pertinents.

D'après Kitchenham et al.[91] les éléments composant cette méthodologie sont les suivants, pour l'informatique :

— *Population* : cela peut être un rôle spécifique (par exemple, testeurs), une catégorie de développeur (par exemple, novice ou expérimenté), un domaine d'application (par exemple, les systèmes de contrôles) ou un groupe industriel (par exemple, des compagnies de télécommunications) ;
— *Intervention* : c'est une méthodologie, un outil, une technologie,une procédure qui adresse un problème spécifique (par exemple, une technique de spécification des exigences) ;
— *Comparison* : cela correspond à la comparaison de ce qui a été sélectionné pour l'*Intervention* (*i.e.* ce que l'on compare. Par exemple, les performances des outils sélectionnés) ;
— *Outcomes* : cela regroupe des facteurs d'importance tels que la réduction des coûts de production ou l'amélioration de la fiabilité (*reliability*).

Dans mon approche l'instanciation correspondante est la suivante :

— *Population* : dans mon contexte, la population est constituée d'articles présentant des solutions améliorant le respect de la vie privée et la protection des données ;
— *Intervention* : comme pour Petersen et al. [131], dans le contexte de mon étude, aucune intervention spécifique n'est investiguée ;
— *Comparison* : comme le domaine de recherche étudié est large, et qu'aucune intervention spécifique n'est étudiée, la comparaison concerne les notions liées à la vie privée, et non pas un outil ou une technique en particulier ;

— *Outcome* : comme Petersen et al., je ne considère pas de résultats mesurables [131], car je ne me concentre pas sur des études empiriques (*empirical studies*).

À partir de l'instanciation de ces éléments, les termes de recherche de la SMS sont définis. Les termes de recherche pour les SMS sont plus génériques que les *systematic reviews* [91]. Ils sont dirigés par les questions de recherche [129]. Comme Petersen et al. [131], je regroupe les mots clés, avec leur synonymes, dans des ensembles. À noter que les termes de recherche sont en anglais.

Ce qui donne :

— Ensemble 1 : termes de recherche en lien avec le domaine de recherche, par exemple : « *software engineering* » (comme [131]) ;

— Ensemble 2 : termes de recherche directement en lien avec la population. Il comprend deux sous-ensembles :

  — Sous-ensemble 2.1 : les termes généralement utilisés pour parler du respect de la vie privée (par exemple : « *privacy* » ou « *data protection* ») ;

  — Sous-ensemble 2.2 : les termes utilisés dans les différents types de représentations de la vie privée, détaillés en section 3 (par exemple, « *Purpose Limitation* » pour la représentation *via* les principes, ou « *Linkability* » pour les menaces) ;

— Ensemble 3 : termes de recherche liés à l'amélioration du respect de la vie privée (par exemple : « *enhancing* » ou « *ensuring* »).

La table 4.2 présente l'ensemble des mots clés sélectionnés selon les ensembles et sous-ensembles identifiés. À partir de ces mots clés, j'ai défini la requête structurée disponible en Annexe A.5. Ainsi les ensembles sont connectés par un « et » logique (*AND*), de la forme `(1) AND (2) AND (3)` (l'ensemble 2 étant composé des sous-ensembles 2.1 et 2.2). Les termes à l'intérieur des ensembles, et sous-ensembles, sont connectés par un « ou » logique (*OR*), de la forme `("software engineering" OR systems OR ...)`.

Brereton et al. [28] recommandent d'interroger les sources électroniques suivantes : IEEExplore, ACM Digital library, Google scholar, Citeseer library, Inspec, ScienceDirect et EI Compendex.

Dans un premier temps, j'ai sélectionné ACM Digital library car mon affiliation me permettait d'avoir un accès aux documents et qu'il est possible d'interroger la base à l'aide d'une requête structurée et de sélectionner des filtres.

Pour avoir un contenu uniforme dans les articles étudiés (*i.e.* de même type), les filtres que j'ai choisis sont les suivants :

— Type de contenu : RESEARCH ARTICLE ;

— Format du média : PDF ;

— Type de publication : PROCEEDINGS ;

Table 4.2 – Mots-clés sélectionnés pour la recherche d'articles.

Ensemble	Mots-clés
1	"software engineering", systems, software, applications, apps, database, architecture, devices, network, programs, code
2.1	privacy, "data protection", GDPR, "Privacy Act", privacy-related, privacy-type, "privacy by design", "personal data"
2.2	Linkability, Identifiability, Non-Repudiation, Detectability, "Disclosure of Information", "Content Unawareness", "Policy and Consent Non-Compliance", "Information Monitoring", "Information Aggregation", "Information Storage", "Information Collection", "Information Transfer", "Information Personalization", Solicitation, Surveillance, Interrogation, Aggregation, Identification, Insecurity, "Secondary Use", Exclusion, "Breach of Confidentiality", Disclosure, Exposure, "Increased Accessibility", Blackmail, Appropriation, Distortion, Intrusion, "Decisional Interference", "Lawfulness, Fairness and Transparency", "Purpose Limitation", "Data Minimization", Accuracy, "Storage Limitation", "Integrity and Confidentiality", Accountability, "Data Subject Rights", "Right to Be Forgotten", "Data Quality", "Adequate protection", "Data Portability",Transparency, "Data Breach Notification", "Accountability and (Provable) Compliance", "Notice/Awareness", "Enforcement/Redress", "Access/Participation", "Integrity/Security", "Choice/Consent", "Collusion Resistance", "Mutual Authentication", "Consent Revocation", "Contextual Privacy", "Data Confidentiality and Integrity", Anonymity, Unlinkability, Undetectability, Unobservability, "Information-Flow Security Policies", Non-Interference, "Purpose Binding", "Data Subject Consent" "Necessity of Data Collection and Processing" "Observational Equivalence"
3	enhancing, ensuring, verification, formalisation, implementation, specification, protection, compliance

Afin de permettre d'identifier l'influence du RGPD (règlement central en Europe pour la protection des données personnelles et également très médiatisé), tout en permettant d'avoir une période de temps suffisamment étendue pour permettre l'identification de l'évolution des tendances, j'ai également choisi le filtre suivant :

— Année de publication : de 2011 à 2021 (+/- 5 ans par rapport à l'entrée en application du RGPD [55]).

Critères de sélection – Kitchenham et al. fournissent un ensemble de recommendations pour la sélection d'articles à classifier [91]. Les critères de sélection sont de deux types : inclusion et exclusion. Ces critères servent à ne garder que les articles permettant de répondre aux RQs. Il est également recommandé de garder

la liste des articles qui ont été exclus. De même, lorsqu'une seule personne a procédé à la selection des articles, elle doit discuter avec des relecteurs des articles exclus et de ceux inclus.

En se fondant sur les critères de Petersen et al. [131] et en les adaptant à mon sujet de SMS, les critères d'inclusion choisis sont les suivants :

— $\boldsymbol{i_1}$:Article qui a pour but de proposer des solutions pour améliorer le respect de la vie privée (méthodologie, outils, modélisation, ...) ;

— $\boldsymbol{i_2}$:Article qui appartient au domaine de l'informatique ;

De même, en se fondant sur les critères de Petersen et al. [131], les critères d'exclusion choisis sont :

— $\boldsymbol{e_1}$:Article écrit en une autre langue que l'anglais ;

— $\boldsymbol{e_2}$:Article non accessible gratuitement *via* mon affiliation ;

— $\boldsymbol{e_3}$:Article pour lequel je n'ai pas accès à l'intégralité du texte ;

— $\boldsymbol{e_4}$:Article qui ne propose pas de solution mais une définition de la vie privée (par exemple, les taxonomies) ;

À noter qu'il n'est pas rare d'avoir plus de critères d'exclusion que d'inclusion [131, 48, 66, 91].

J'ai ainsi défini un processus de sélection d'articles de façon similaire à celle de Guaman et al. [66]. L'article est inclus (*included*) quand tous les critères d'inclusion sont remplis ; exclu (*excluded*) quand l'article remplit au moins un critère d'exclusion (même s'il remplit les critères d'inclusion) ; et non clair (*unclear*) quand il y a certains doutes. Je suis ce processus, puis mes trois encadrants vérifient ma sélection pour valider les articles retenus. Ce processus est représenté sur la figure 4.3.

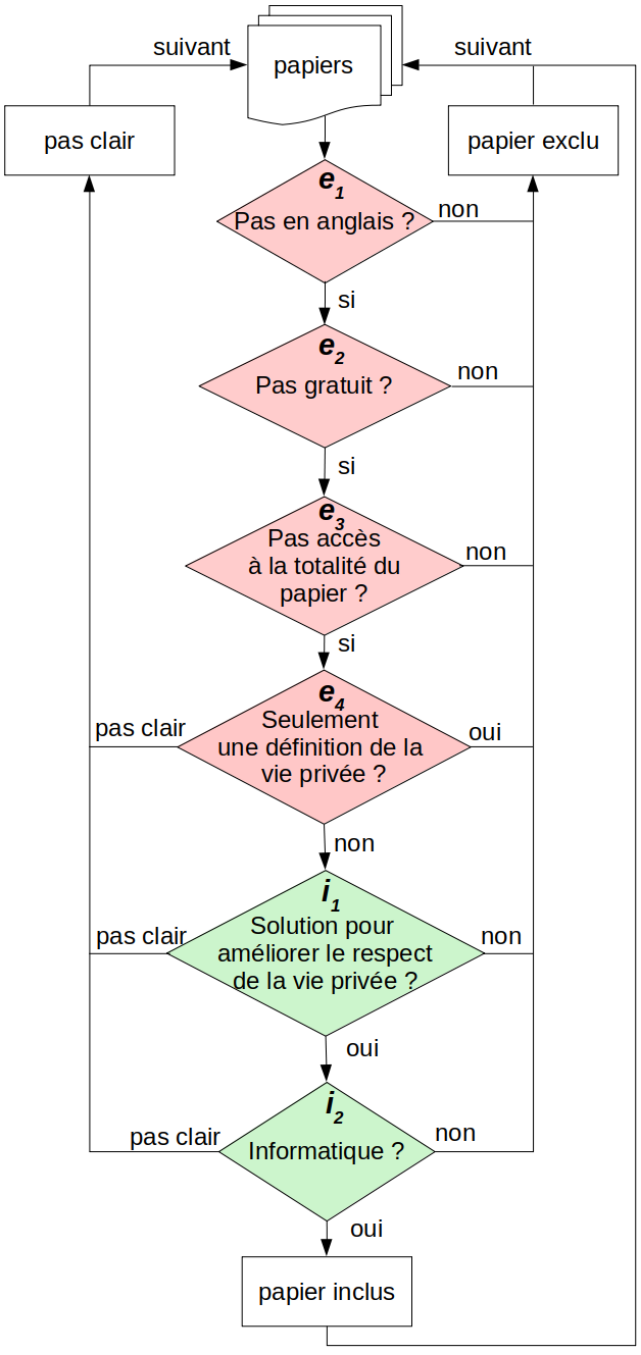


Figure 4.3 – Processus de sélection des articles à partir des résultats de la requête.

Schéma de classification – Mon schéma de classification suit le même principe que celui de Guaman et al. [66], comme illustré sur la figure 4.4. Le but de la classification est de pouvoir répondre aux questions de recherche définies précédemment.

Pour répondre aux questions de recherche **RQ1** et **RQ3**, j'identifie les catégories de GePyR, définies en section 4.1, liées aux notions considérées dans l'article étudié.

Pour répondre à **RQ2** et **RQ3**, j'identifie les types de représentation, détaillés en section 3. Si l'article mentionne explicitement le type de représentation considérée (par exemple, les vulnérabilités), alors ce type est indiqué dans la classification. Sinon, si l'article mentionne explicitement une représentation spécialisée, alors le type indiqué dans la classification sera sélectionné comme le type de représentation lié (par exemple, si l'article propose une solution pour suivre la *purpose limitation* du GDPR, alors le type de représentation sera celle des principes). Enfin, si aucune mention n'est faite du type de représentation ou de représentation spécialisée, alors le type de représentation pour la classification sera choisi selon la définition de représentation, donnée en section 3, qui se rapproche le plus de ce qui est expliqué dans l'article à classer (par exemple, si l'approche propose une solution prenant en compte une activité qui peut être qualifiée de nocive, alors l'article sera classé avec comme type de représentation : activité nocive).

Pour répondre à la question de recherche **RQ4**, j'identifie si un outil est précisé dans l'article, ou bien si un outil, non précisé mais que je considère comme très bien établi (par exemple, Coq [78] pour un assistant de preuves), pourrait être utilisable dans l'approche proposée.

Finalement, pour répondre à la question **RQ5**, j'identifie l'année de publication de l'article.

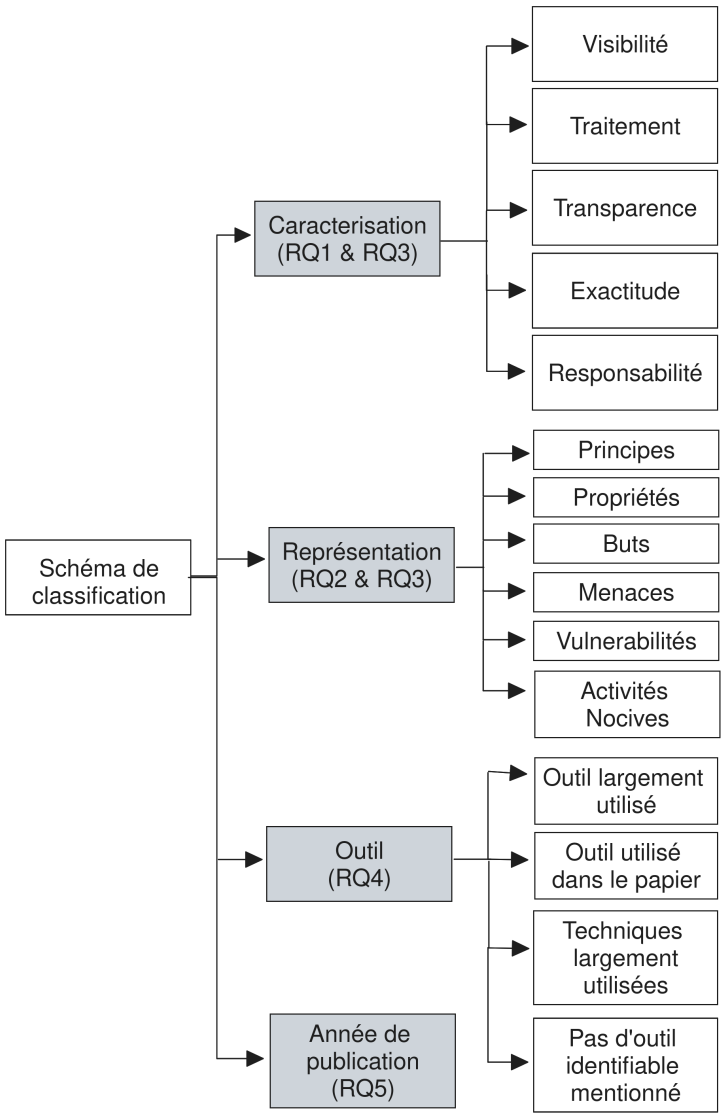


Figure 4.4 – Schéma de classification pour répondre aux questions de recherche.

d) Évaluation du protocole

J'ai établi le protocole de recherche, qui a été évalué, discuté et validé par mes trois encadrants, comme le recommande Brereton et al. [28]. Ainsi, pour l'extraction des données et la validation de ma classification, j'ai appliqué la même méthode que Petersen et al. [131], c'est à dire que je les ai réalisées et que mes encadrants les ont vérifiées et validées. Ce processus permet de vérifier la cohérence du protocole [91], en particulier lorsque le nombre d'articles traités est important [28].

e) Validité de l'évaluation

D'après Petersen et Gencel [130], différents types de validités doivent être pris en compte :

- validité de description (*descriptive validity*) : « pouvons-nous décrire la vérité objective/subjective avec précision ? »
- validité de théorie (*theoretical validity*) : « saisissons-nous ce que nous avons l'intention de saisir ? »
- généralisation (*generalizability*) : « dans quelle mesure pouvons-nous généraliser les résultats ? »
- validité d'interprétation (*interpretive validity*) : « les conclusions/inférences tirées sont-elles raisonnables compte tenu des données représentant une vérité objective/subjective ? »

Pour leur type de validité de généralisation, ils se fondent sur celles de Maxwel [111] :

- interne (*internal*) : « généralisation au sein de la communauté, du groupe ou de l'institution étudiée à des personnes, des événements et des contextes qui n'ont pas été directement observés ou interrogés. »
- externe (*external*) : « se réfère à la mesure dans laquelle il est possible d'étendre le compte rendu d'une situation ou d'une population particulière à d'autres communautés, groupes ou institutions. »

Ma façon de prendre en compte ces validités est semblable à celle de Petersen et al. [131]. Je prends également en compte les éléments suivants :

- validité de description : j'utilise un formulaire de collecte de données pour enregistrer les données traitées. Il est ainsi possible de revisiter le processus d'extraction des données.
- validité de théorie : en suivant les recommandations définies par Kitchenham et al. [90] et par Brereton et al. [28], je fais l'extraction des données et mes encadrants la vérifient. Or, comme Petersen et al. [131] le font remarquer, même si ce processus permet de réduire la menace d'un biais, cette dernière ne peut pas être totalement écartée car l'extraction est soumise à un jugement humain.
- généralisation : pour adresser la menace d'un biais dans cette validité, je me réfère aux catégories de GePyR et à ses spécialisations dans les autres représentations. Ainsi, pour la validité interne, je peux généraliser les observations faites pour une catégorie de GePyR dans une de ses spécialisations (par exemple, une observation faite pour la catégorie de visibilité peut être étendue à ses spécialisations telle que la vulnérabilités d'*Identification*). Réciproquement, pour la validité externe, je peux généraliser les observations faites pour une représentation spécifique au sein de la catégorie de GePyR associée. Grâce aux catégories génériques de GePyR et à ses spécialisations

identifiées, ce biais peut être contrôlé car ces généralisations sont traçables (par exemple, une observation faite pour le principe de *Purpose limitation* peut être étendue à la catégorie de traitement). De plus, la séparation des notions liées à la vie privée dans des catégories différentes de GePyR permet de limiter la généralisation d'observations à des notions comparables.

— validité d'interprétation : « une menace pour l'interprétation des données est le biais du chercheur. » selon Petersen et al. [131]. Pour limiter ce biais, j'ai utilisés les recommandations et les recommandations pour les *Systematic Mapping Studies* et les *Systematic reviews* [129, 131, 91] qui sont largement utilisées [131, 115, 48, 66, 54, 5, 116, 83] (et citées plus de quatorze mille fois d'après Google Scholar). De même, afin de limiter ce biais, j'ai réalisé l'interprétation et mes encadrants l'ont vérifiée.

4.4.2 . Réalisation de la *Systematic Mapping Study*

La phase de réalisation correspond à la recherche d'articles à l'aide d'une base de données, au filtrage des résultats de la requête selon des critères prédéfinis, et au classement des articles.

a) Recherche d'articles

Pour la recherche d'articles, comme dit précédemment, j'ai interrogé la base de données de ACM Digital library au moyen de la requête disponible en Annexe A.5, avec les filtres suivants : RESEARCH ARTICLE, PDF et PROCEEDINGS, et en restreignant la recherche aux années de publication de 2011 à 2021. Cette requête a retourné 197 articles.

b) Filtrage des résultats selon les critères de sélection

J'ai ensuite appliqué les critères de sélection précédents et le processus de sélection illustré sur la figure 4.3. Je n'ai ainsi gardé que les articles rédigés entièrement en anglais, auxquels j'ai eu accès en totalité et gratuitement (*via* mon affiliation) et qui proposent une solution, et pas seulement une définition, pour améliorer le respect de la vie privée, en informatique. Cela correspond au final à **79** articles (soit environ 40% des articles retournés par la base de donnée).

c) Classement des articles selon le schéma de classification

J'ai appliqué le schéma de classification présenté précédemment, illustré sur la figure 4.4. Ainsi, pour sélectionner :

— la catégorie de GePyR :
  — si une représentation spécifique (par exemple, la propriété *Anonymity* par Kita et al. [89]) est considérée dans l'article, alors la catégorie se spécialisant en cette représentation est sélectionnée (donc, pour cet article, *Visibility*) ;
  — sinon, je compare les problématiques considérées dans l'article et je les compare avec les définitions des catégories de GePyR (détaillées en section 4.1) pour sélectionner la catégorie la plus pertinente. Ceci implique un biais d'analyse, car ce classement dépend de mon inter-

prétation de l'approche, mais la validation par la relecture de mes encadrants permet de limiter ce biais ;

— le type de représentation :
  — si le(s) type(s) de représentation est/sont mentionné(s) directement dans l'article (par exemple, l'article de Squicciarini définit des *properties* [162]) alors il est classé selon ce type de représentation (donc, dans ce cas-la, *Property*) ;
  — si une/des représentation(s) spécialisée(s) est/sont mentionné(s) directement dans l'article (comme dans l'exemple précédent avec la propiété *Anonymity* [89]), alors il est classé selon ce type de représentation contenant cette représentation spécialisée (donc, dans ce cas-là *Propriété*) ;
  — sinon, j'analyse l'article pour trouver quel type de représentation correspond le mieux, selon les définitions des type de représentations et des représentations spécialisées (détaillés en section 3). De même cela implique un biais d'analyse, qui est limité par la relecture de mes encadrants ;
— la présence d'un outil :
  — si un outil est spécifié et que c'est un outil utilisé par ailleurs (au moins une citation) ou bien que l'outil proposé est une extension d'un outil utilisé par ailleurs, l'article est associé au code ● ;
  — si un outil est spécifié mais que cela correspond à un prototype ou à une implémentation pour l'évaluation de l'approche, l'article est associé au code ◐ ;
  — si aucun outil n'est spécifié mais que l'approche repose sur des techniques suffisamment connues dans la littérature pour qu'il semble possible qu'un outil existant puisse être utilisé, l'article est associé au code ○ ;
  — sinon, l'article est associé au code -.

Les tableaux 4.3 et 4.4 correspondent à la classification que j'ai réalisée pour les articles sélectionnés. Pour chaque article sélectionné, elles présentent la catégorie issue de GePyR associée à l'article, le type de représentation de la vie privée considéré, la présence d'un outil ainsi que l'année de publication.

4.4.3 . Analyse de la *Systematic Mapping Study*

La phase de rapport correspond à l'analyse de la classification afin de répondre aux questions de recherche, en présentant des métriques, tels que des totaux, et non des analyses plus profondes [91]. Cette analyse permet de découvrir les tendances et les manques du domaine de recherche étudié [131].

Pour répondre à la question de recherche **RQ1** : « quelle est la couverture des notions liées à la vie privée ? », j'analyse les articles selon les catégories de GePyR

Table 4.3 – Classification suite à la SMS - I.

REF	GePyR	REPRESENTATION	TOOL	YEAR
[107]	Visibilité	Menace & Propriété	◐	2011
[185]	Visibilité	Menace	◐	2011
[156]	Visibilité	But	-	2011
[47]	Visibilité & Transparence	Vulnérabilité	◐	2012
[184]	Visibilité	Vulnérabilité	◐	2012
[101]	Visibilité	But & Propriété	●	2012
[110]	Visibilité	Vulnérabilité	◐	2013
[151]	Exactitude	Menace	-	2013
[162]	Traitement & Exactitude	Propriété	◐	2013
[179]	Visibilité	Menace	◐	2013
[145]	Visibilité & Transparence	Menace & Propriété	◐	2013
[120]	Visibilité	Menace	◐	2013
[150]	Visibilité & Transparence	Menace & Vulnérabilité	-	2013
[72]	Visibilité	Vulnérabilité	◐	2014
[138]	Visibilité	Vulnérabilité	-	2014
[164]	Visibilité	Vulnérabilité	◐	2014
[99]	Visibilité	Activité nocive & Menace	◐	2015
[139]	Visibilité & Exactitude	Propriété & But	●	2015
[113]	Visibilité	Activité nocive & Menace	◐	2016
[77]	Visibilité	Vulnérabilité	◐	2016
[93]	Visibilité	Menace	◐	2016
[161]	Exactitude	Menace	◐	2016
[73]	Responsabilité	Principe	-	2017
[33]	Visibilité	Propriété	●	2017
[84]	Visibilité	Vulnérabilité & Propriété	●	2017
[86]	Visibilité & Traitement	Propriété	●	2017
[52]	Visibilité	Vulnérabilité	◐	2017
[146]	Visibilité	Propriété	◐	2017
[147]	Visibilité	Menace	◐	2017
[6]	Visibilité	Menace	◐	2017
[15]	Visibilité	Menace & Vulnérabilité	-	2017
[103]	Visibilité	Vulnérabilité	-	2017
[1]	Visibilité	But	◐	2017
[7]	Responsabilité	Principe	-	2018
[29]	Traitement & Visibilité	Activité nocive	◐	2018
[23]	Traitement	Activité nocive	-	2018
[190]	Visibilité	Menace	-	2018
[89]	Visibilité	Propriété	-	2018
[104]	Responsabilité	Principe	-	2018
[123]	Exactitude & Transparence	Principe	-	2018
[61]	Visibilité	Menace	-	2018
[189]	Visibilité	Menace & Propriété	◐	2018
[88]	Visibilité	Menace	◐	2018
[176]	Visibilité	Principe	-	2018
[56]	Responsabilité	Principe	-	2018
[160]	Visibilité	But	◐	2019

Table 4.4 – Classification suite à la SMS - II.

REF	GePyR	REPRÉSENTATION	OUTIL	ANNÉE
[3]	Responsabilité	Principe	-	2019
[155]	Visibilité & Responsabilité	Principe	◐	2019
[178]	Visibilité	Vulnérabilité	◐	2019
[87]	Transparence & Visibilité	Principe & Propriété	◐	2019
[183]	Visibilité	Propriété	◐	2019
[75]	Visibilité & Responsabilité	Principe	-	2019
[16]	Responsabilité	Principe	-	2019
[122]	Responsabilité	Principe	-	2019
[76]	Visibilité	Menace	-	2019
[67]	Visibilité	Menace	-	2019
[85]	Visibilité	Menace	◐	2020
[149]	Visibilité	Vulnérabilité & Menace	◐	2020
[32]	Responsabilité	Principe	-	2020
[186]	Visibilité	Menace	◐	2020
[112]	Visibilité	Vulnérabilité	-	2020
[140]	Visibilité	Propriété	◐	2020
[105]	Visibilité	Vulnérabilité	◐	2020
[182]	Visibilité	Menace	◐	2020
[137]	Visibilité	Propriété	◐	2020
[34]	Visibilité & Exactitude	Menace	●	2020
[31]	Visibilité	Menace	-	2020
[166]	Visibilité	Menace	-	2020
[10]	Exactitude	Menace	◐	2020
[100]	Visibilité	Vulnérabilité & Propriété	-	2020
[80]	Transparence	Menace	◐	2021
[173]	Visibilité & Transparence	Menace	◐	2021
[8]	Visibilité	Vulnérabilité & Menace	-	2021
[167]	Responsabilité	Principe	-	2021
[108]	Visibilité	Menace	◐	2021
[169]	Responsabilité	Principe	-	2021
[51]	Visibilité	Menace	◐	2021
[102]	Visibilité	But	●	2021
[9]	Responsabilité	Principe	◐	2021

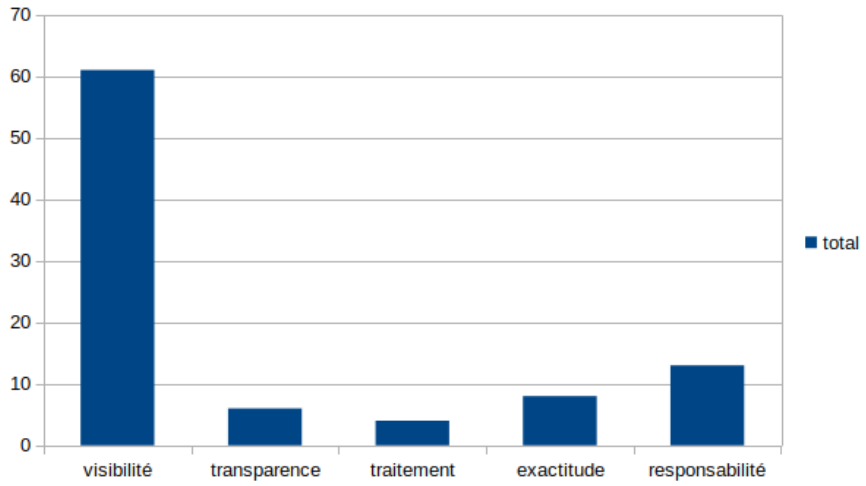


Figure 4.5 – Nombre d'articles selon leur catégorie de GePyR.

couvertes. Le nombre d'articles selon leur catégorie de GePyR est présenté sur la figure 4.5. On peut ainsi remarquer que les notions les plus considérées, représentant plus de 66% des articles, sont liées à la visibilité des données personnelles (*visibility*), suivies par les notions liées à la responsabilité de traitement (*accountability*), 14%. À l'autre bout du spectre, les notions les moins considérées, avec seulement près de 4% des articles, sont liées au traitement des données personnelles (*processing*). Il est à noter que certains articles proposent des solutions concernant plusieurs catégories de GePyR, tel que Ulybyshev et al. [173], dans lequel les auteurs proposent une solution concernant à la fois la visibilité des données et leur exactitude (*data accuracy*).

Pour répondre à la question de recherche **RQ2** : « quelles sont les représentations de la vie privée qui sont les plus considérées et les moins considérées ? », j'analyse les articles selon les types de représentations considérés dans ceux-ci. Le nombre d'articles selon chaque type de représentation est illustré sur la figure 4.6. Ainsi, on peut remarquer que la représentation la plus considérée, avec près de 35,5% des articles, est la représentation *via* les menaces. Les représentations *via* les principes, *via* les vulnérabilités et *via* les propriétés sont considérées plutôt de façons similaires (entre 17 et 19% des articles). Par contre, les buts et activités nocives sont assez peu considérés, comptant entre 4 et 6% des articles. Ici aussi, il est à noter que certains articles couvrent plusieurs types de représentation, tel que celui écrit par Lu et al. [107] qui propose de vérifier qu'un système respecte certaines propriétés afin de le protéger face à des menaces.

Pour répondre à la question de recherche **RQ3** : « quel type de représentation est le plus utilisé selon les notions considérées ? » j'analyse les notions considérées

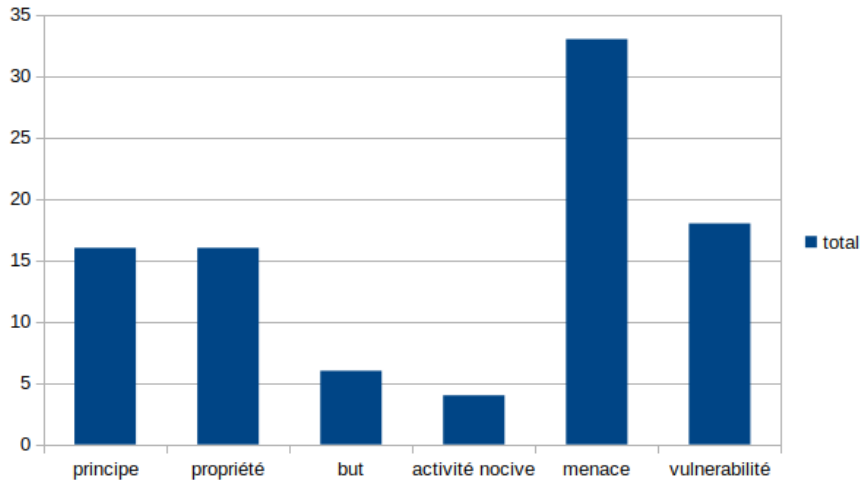


Figure 4.6 – Nombre d'articles selon leur type de représentation de la vie privée.

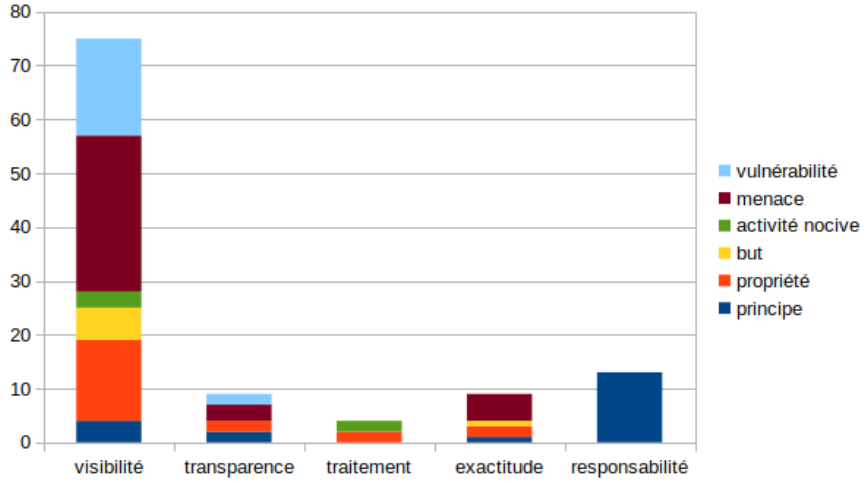


Figure 4.7 – Notions de vie privées considérées selon le type de représentation de vie privée.

selon les types de représentation dans les articles. Le résultat est présenté sur la figure 4.7. On peut ainsi peut remarquer que les notions liées à la visibilité peuvent être considérées selon tous les types de représentation identifiés, mais que le type de représentations le plus utilisé est celui *via* les menaces (47%), suivi des représentations *via* les vulnérabilités (29.5%) puis les propriétés (24.5%). Les notions liées à la transparence sont également considérées selon plusieurs types de représentation en proportions similaires, sauf la représentation *via* les buts ou

les activités nocives. De même, les notions liées à l'exactitude des données sont couvertes par différents types de représentations, mais avec une prépondérance pour la représentation *via* les menaces. En revanche, les notions liées au traitement des données ne sont couvertes que par les représentations *via* les propriétés ou les activités nocives (en proportion égale). Cela est encore plus marqué en ce qui concerne les notions liées à la responsabilité qui ne sont couvertes que par la représentation *via* les principes.

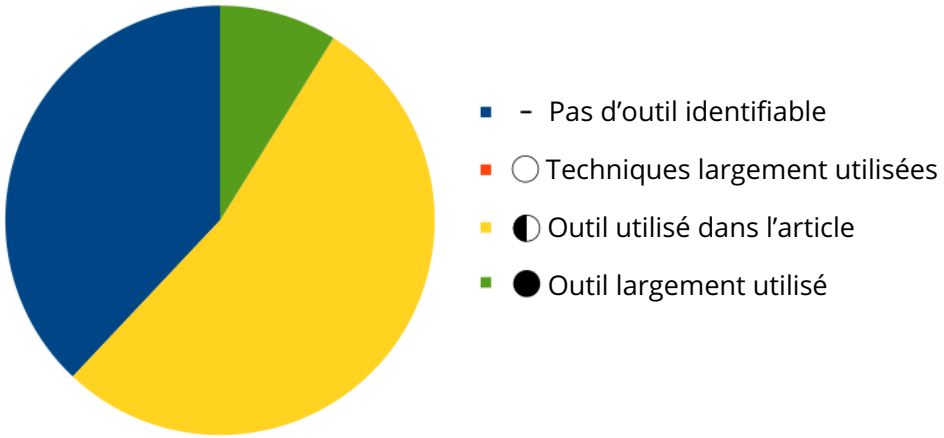


Figure 4.8 – Proportion des types d'outils utilisés dans les articles.

Pour répondre à la question de recherche **RQ4** : « existe-t-il beaucoup d'outils pour les informaticiens pour les aider à respecter la vie privée ? » j'analyse le type d'outil utilisé dans les articles, en reprenant le code précédemment défini dans cette section (-, ○, ◐, ●). Cette proportion est illustrée sur la figure 4.8. Ainsi, on peut remarquer que la majorité des articles, soit 53%, proposent un prototype ou bien une implémentation pour évaluer leur approche (◐). La plupart des articles restants, soit 37% des articles, ne proposent aucun outil (-). Un petit nombre, 8,8%, fondent leur approche sur des outils déjà existants (●). Par contre, aucun article n'utilise des techniques ou théories bien connues sans proposer un outil ou une implémentation. Il semble ainsi qu'il n'y ait pas tellement d'outils corrects (*sound*) pour aider les développeurs à respecter la vie privée mais que des outils minimaux (prototype, implémentation pour évaluation) soient réalisés pour des études spécifiques.

Pour répondre à la question de recherche **RQ5** : « quelle est l'évolution des publications sur l'amélioration du respect de la vie privée ? » j'analyse le nombre d'articles publiés selon les années, comme cela est présenté sur la figure 4.9. On peut remarquer qu'il y a un premier pic de publication en 2013, ce qui correspond

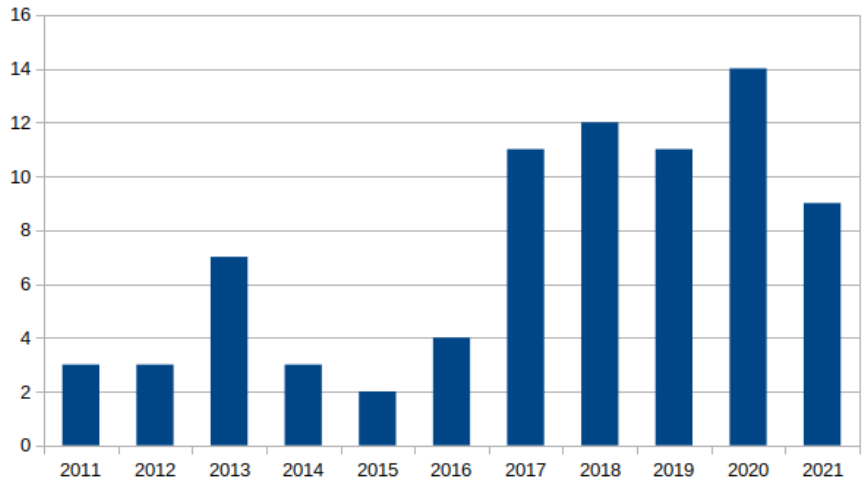


Figure 4.9 – Nombre de publications par année.

à l'année suivant la proposition du RGPD [42](en 2012). Ensuite, à partir de l'année 2017, année suivant la publication du RGPD [55], le nombre de publications augmente fortement, même si on peut remarquer que le nombre de publications baisse de nouveau en 2021, ce qui correspond à la période post-COVID-19.

Cette SMS a remplacé la classification réalisée dans [40], qui était basée sur un ensemble d'articles sélectionnés de façon non systématique. La méthode ce classification utilisée ici est plus rigoureuse que celle réalisée précédemment.

Il existe une grande variété de représentations qui, pourtant, peuvent faire références à des notions similaires mais sous différents points de vues. Le but de GePyR est d'offrir une approche homogène face à la complexité venant de cette diversité et de faciliter la comparaison de solutions proposées afin d'améliorer le respect de la vie privée. Pour cela, GePyR repose sur des catégories génériques qui reprennent les notions considérées dans ces représentations : les catégories de *caractérisation du Traitement*, de *caractérisation de la Visibilité*, de *caractérisation de la Transparence*, de *caractérisation de l'Exactitude des données* et de *caractérisation de la Responsabilité*. Ces catégories sont spécialisables dans les représentations de l'état de l'art identifiées dans le chapitre précédent. J'ai appliqué GePyR sur un cas d'utilisation qui permet d'illustrer comment et pourquoi une entreprise pourrait l'utiliser. J'ai également utilisé GePyR dans une *Systematic Mapping Study* et ana-

lysé 79 articles de la littérature proposant des solutions pour améliorer le respect de la vie privée. Pourtant, ces catégories ne permettent pas d'identifier facilement quels sont les éléments à prendre en compte pour vérifier qu'un système respecte la vie privée, car elles ne les modélisent pas.

5 - Ontologie du contexte lié à la vie privée - PyCO

Ce chapitre présente PyCO, une nouvelle ontologie du contexte lié à la vie privée. « Une ontologie est un modèle de données contenant des concepts et relations permettant de modéliser un ensemble de connaissances dans un domaine donné » [1]. Le but de cette ontologie est d'identifier les éléments à prendre en compte pour vérifier le respect de la vie privée. En effet, la validité de propriétés liées à la vie privée dépend du contexte dans lequel est inscrit le système étudié.

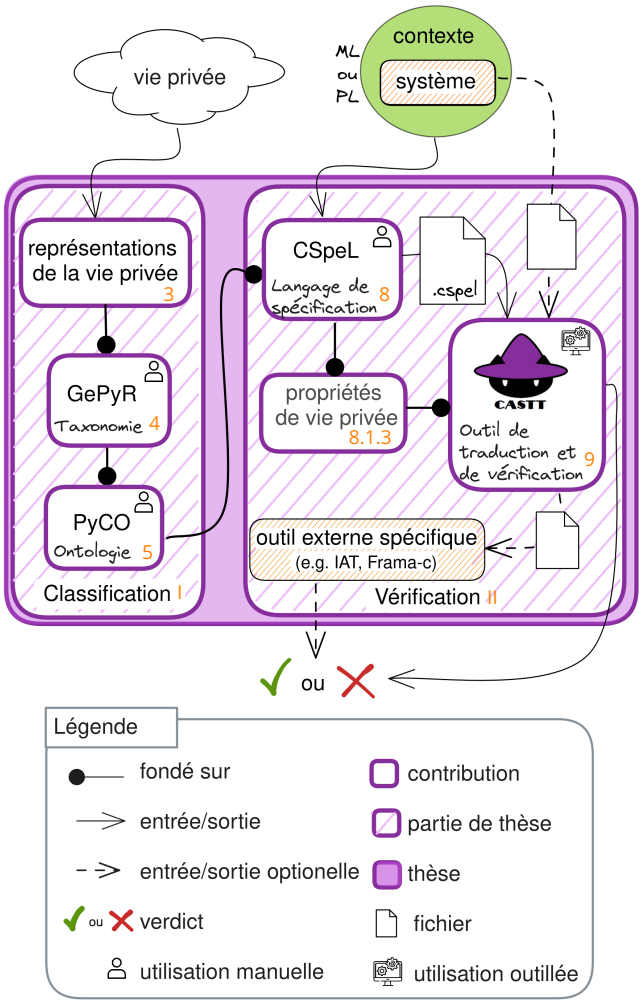


Figure 5.1 – Rappel - Lien entre les parties classification et vérification.

1. https://fr.wikipedia.org/wiki/Ontologie_(informatique)

Comme illustré sur la figure 5.1, PyCO est fondé sur GePyR, définie en section 4, et sert de support à la partie vérification développée dans la Partie II. Ainsi, cette ontologie participe au but plus général de ma thèse qui est de vérifier formellement qu'un système respecte des propriétés de vie privée. La section 5.1 présente PyCO, ses éléments clés et leur relations. Ensuite la section 5.2 illustre l'utilisation de cette ontologie sur le cas d'utilisation, défini en section 2. Finalement la section 5.3 présente une utilisation plus générale de PyCO pour classer la littérature, avec une illustration portant sur six articles.

5.1 . Définition

Pour définir PyCO, je me suis basée sur GePyR en me limitant aux notions liées aux catégories de *caractérisation du Traitement*, de *caractérisation de la Transparence* et de *caractérisation de la Responsabilité*, définies par GePyR. Cette démarche a pour but d'identifier les éléments à prendre en compte pour pouvoir vérifier formellement le respect de la vie privée.

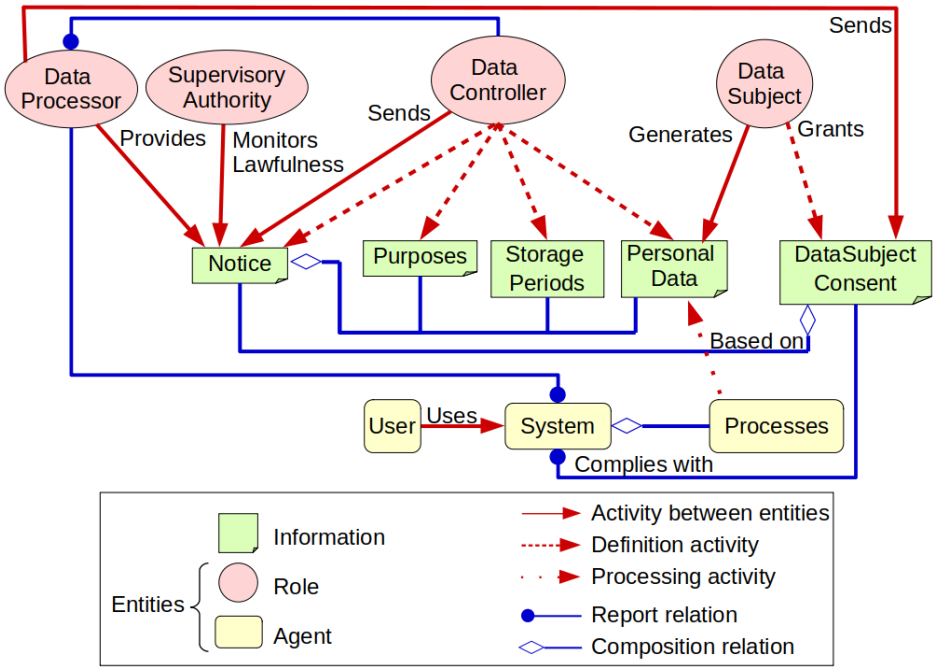


Figure 5.2 – Ontologie du contexte lié à la vie privée - PyCO.

Pour identifier ces éléments et leurs relations, je me suis basée sur les éléments pris en compte dans des solutions existantes liées aux notions de ces catégories [96, 143, 188, 60, 170, 168, 12, 124, 70, 132, 53, 18, 4]. Pour la terminologie je me suis largement inspirée des définitions du RGPD [55] et de celles du *Working*

Party 29 [181][2]. En effet, ces documents définissent en détail les éléments du contexte tels que les parties prenantes (*stakeholders*), l'environnement et le système. Il est donc important de noter que PyCO a été construit en s'appuyant sur des sources variées, incluant le RGPD et des articles scientifiques. Certains éléments et relations peuvent donc légèrement différer des modèles existants pour des questions de généricité. Malgré cela, cette ontologie peut être vue comme un dénominateur commun entre ces sources.

Ces éléments et leurs relations sont représentés dans mon ontologie, PyCO, présentée sur la figure 5.2. La figure est laissée en anglais car elle correspond à la figure dans notre article publié [40] (même s'il y a quelques différences, qui seront expliquées par la suite). Ils sont définis dans la définition 7.

Définition 7: Éléments de PyCO

- **Données personnelles** (***Personal Data***) : données liées à une personne identifiable ;
- **Personne concernée** (***Data Subject***) : personne à laquelle se rapportent les données personnelles ;
- **Responsable du traitement** (***Data Controller***) : entité qui a l'intention de traiter les données et qui définit donc les finalités de leur traitement ;
- **Sous-traitant** (***Data Processor***) : entité qui traite les données pour le compte du responsable du traitement et qui fournit les garanties d'implémentation des mesures appropriées pour la protection de la vie privée ;
- **Système et processus** (***System and Processes***) : entités qui utilisent et traitent les données personnelles ;
- **Utilisateur** (***User***) : individu qui utilise le système ;
- **Finalités de traitement** (***Purposes***) : finalités pour lesquelles les données personnelles sont traitées ;
- **Durées de conservation** (***Storage Duration***) : durées pendant lesquelles les données sont conservées et donc pouvant être traitées ;
- **Consentement de la personne concernée** (***Data Subject Consent***) : consentement accordé par la personne concernée ;
- **Notice** (***Notice***) : ensemble d'informations nécessaires pour établir le consentement ;
- **Autorité de contrôle** (***Supervisory Authority***) : entité surveillant la conformité avec la loi.

Ces éléments peuvent être divisés en deux sous-ensembles : les entités et les informations. Parmi les entités, certaines correspondent aux rôles légaux de certains

2. Nouvellement dénommé « Comité Européen de la Protection des Données ».

individus, entreprises ou organisations, selon le contexte, tandis que les autres sont des agents, c'est à dire des individus ou composants informatiques réalisant des actions liées au système ou interagissant avec celui-ci. Dans certains contextes, la même personne, entreprise ou organisation, peut à la fois avoir un rôle et être un agent. Par exemple, un utilisateur de réseau social est à la fois l'utilisateur du système et la personne concernée par les données personnelles (qu'il fournit au réseau social). De même, plusieurs rôles peuvent être instanciés par la même personne, entreprise ou organisation. Par exemple, lorsque le responsable du traitement ne délègue pas la mise en œuvre à un sous-traitant, les responsabilités liées à ce rôle sont également attribuées au responsable du traitement (je considère dans ce cas, à des fins d'analyse, que le responsable du traitement prend également le rôle de sous-traitant). Par exemple, c'est ainsi au responsable du traitement de mettre en œuvre les mesures appropriées pour le traitement des données. Cependant, dans certains contextes, ces entités peuvent représenter différentes personnes, entreprises ou organisations, qui ne sont pas liées. Par exemple, dans le cadre d'un logiciel d'hôpital qui enregistre les données des patients, l'utilisateur (*i.e.* le personnel médical) est différent de la personne concernée (*i.e.* un patient), car c'est le personnel médical qui interagit avec le logiciel et non le patient. J'ai également séparé les notions de finalités de traitement et de processus, car ces éléments peuvent être différents selon les contextes. En effet, un processus peut avoir été défini pour plusieurs finalités (par exemple, un processus permettant d'accéder à une donnée), et une finalité peut être réalisée par plusieurs processus (par exemple, lors du raffinement d'un processus, défini pour une finalité, en plusieurs processus permettant de réaliser cette finalité). Pour ces raisons, j'ai choisi de laisser séparées :

— les entités **Personne concernée** et **Utilisateur** ;
— les entités **Responsable du traitement** et **Sous-traitant** ;
— les entités **Processus** et **Finalités de traitement**,

tout en laissant la possibilité que leurs instanciations soient semblables (même si ces séparations ne sont généralement pas considérées dans les ontologies liées à la vie privée disponibles dans la littérature [126, 63, 125]). L'exemple 3 illustre l'instanciation de ces éléments dans le contexte de mon exemple de cas d'usage, défini en section 2.

Exemple 3: Instanciation des éléments de PyCO

— **Données personnelles** : l'adresse e-mail ;
— **Personne concernée** : l'utilisateur du site internet ;
— **Responsable du traitement** : le propriétaire du site internet ;
— **Sous-traitant** : le propriétaire du site internet (car il n'y a pas de délégation à un tiers) ;
— **Système et processus** : les fonctions dans le code du site internet,

qui envoient les notifications d'expiration d'inscription ;

- **Utilisateur** : l'utilisateur du site internet ;
- **Finalités de traitement** : les finalités de « *website administration* » et de « *marketing* » ;
- **Durées de conservation** : les types de durées « *no-retention* » et « *business-practices* » (types de durées de rétentions définis par Hayati et al. [70]) ;
- **Consentement de la personne concernée** : l'accord de traiter l'adresse e-mail pour l'administration du site internet (« *website administration* ») ;
- **Notice** : la composition de l'adresse e-mail, des finalités « *website administration* » et « *marketing* », et des durées « *no-retention* » et « *business-practices* » ;
- **Autorité de contrôle** : la CNIL (Commission nationale de l'informatique et des libertés).

En plus de ces éléments, comme PyCO est une ontologie, elle inclut aussi les liens entre ces éléments. Ces liens sont de deux sortes : les activités et les relations. Les activités représentent des actions entre des entités (*activity between entities*) et peuvent définir (*definition activity*) ou traiter (*processing activity*) des informations. Les relations, quant à elles, représentent soit des liens de composition (*composition relation*), soit des liens d'autorité (*report relation*). Ces relations sont également représentées sur la figure 5.2.

Voici la liste de ces relations (le pictogramme devant chaque élément écrit en gras correspond à la légende de la figure 5.2) :

→ **Activité entre entités** (*activity between entities*) :

- le Responsable du traitement envoie (*Sends*) la Notice au sous-traitant ;
- le Sous-traitant fournit (*Provides*) la Notice à la personne concernée ;
- le Sous-traitant envoie (*Sends*) le Consentement de la personne concernée au responsable du traitement ;
- l'Autorité de contrôle vérifie la légalité (*Monitors Lawfulness*) de la Notice ;
- la Personne concernée génère (*Generates*) des Données personnelles ;
- l'Utilisateur utilise (*Uses*) le Système ;

⇢ **Activité de définition** (*definition activity*) :

- le Responsable du traitement définit la Notice ;
- le Responsable du traitement définit les Finalités de traitement ;
- le Responsable du traitement définit les Durées de conservation ;

- le Responsable du traitement définit les Données personnelles utilisées ;
- la Personne concernée accorde (*Grants*), ou retire son Consentement (ce qui « définit » ce qui est accordé ou non) ;

Activité de traitement (*processing activity*) :

- les Processus traitent les Données personnelles ;

Relation de composition (*composition relation*) :

- la Notice contient les Finalités de traitement (elle est donc composée, entre autres, des Finalités de traitement) ;
- la Notice contient les Durées de conservation (elle est donc composée, entre autres, des Durées de conservation) ;
- la Notice contient les Données personnelles utilisées (elle est donc composée, entre autres, des Données personnelles utilisées) ;
- le Système est composé des Processus ;
- le Consentement de la Personne concernée est basé sur (*Based on*) la Notice (elle est donc composée, entre autres de la Notice) ;

Relation d'autorité (*report relation*) :

- le Sous-traitant traite les données pour le compte du Responsable du traitement (*i.e.* il est « sous ses ordres ») ;
- le Système est contrôlé par le Sous-traitant ;
- le Système doit être conforme (*Complies with*) au Consentement de la personne concernée (*i.e.* il doit le respecter et ne pas aller au-delà de ce qui a été accordé).

La Figure 5.2 diffère légèrement de celle publiée dans notre article [40]. Le type « activité de consommation » (*Consumption activity*) a été renommé en « activité de traitement » (*Processing activity*) car cela correspond mieux à la sémantique du lien et est plus compréhensible. L'activité « envoi » (*Sends*) entre le Système et les Données personnelles a été retirée, car elle est incluse dans l'activité de traitement entre les Processus et les Données personnelles.

5.2 . Illustration sur le cas d'usage

Cette section illustre l'utilisation de PyCO sur le cas d'usage défini en section 2, celui des sites internets. L'équipe R&D utilise PyCO durant l'étape d'implémentation du cycle de vie du site Internet. Dans l'ensemble des articles identifiés avec GePyR, l'équipe en choisit un, par exemple l'article de Hayati et Abadi [70], car leur domaine applicatif correspond également à celui des sites internets. L'équipe utilise PyCO pour identifier les éléments considérés dans l'article et comment ils sont représentés.

Table 5.1 – Instanciation des éléments de PyCO sur le cas d'usage.

PAGE	LIGNE	ÉLÉMENT	REPRÉSENTATION
4	2	Données personnelles	Objet Java
5	39	Processus	Fonctions Java
5	20	Finalités	Mots-clés & *Principal* en JIF [118]
8	26	Durées de conservation	Mots-clés & Labels en JIF [118]
-	-	Notice	Non spécifié
-	-	Consentement de la personne concernée	Non spécifié
4	38	Responsable du traitement	Description en langue naturelle
-	-	Sous-traitant	Non spécifié
5	1	Personne concernée	Description en langue naturelle
5	1	Utilisateur	Description en langue naturelle
-	-	Autorité de contrôle	Non spécifié

L'instanciation des éléments de PyCO selon les représentations des éléments considérés pour l'approche de Hayati et Abadi [70] est représentée dans le tableau 5.1. Pour la traçabilité, le tableau présente également la page et la ligne de la première instanciation de ces éléments. Ainsi, à l'aide de cette ontologie, l'équipe de R&D peut identifier rapidement les éléments qui sont pris en compte dans la solution proposée par Hayati et Abadi, et ceux qui ne le sont pas. Par exemple, on peut remarquer que les Données personnelles et les Finalités de traitement sont considérées mais que le Consentement de la personne concernée, lui, n'est pas considéré. De même, même si la Personne concernée (qui est également l'Utilisateur) et le Responsable du traitement sont spécifiés, ils le sont seulement en langage naturel (l'utilisateur du site internet et l'entreprise à qui appartient le site, *online bookseller*) et n'ont donc pas d'incidence sur le système. L'Autorité de contrôle, quand à elle, n'est pas spécifiée, tout comme le sous-traitant. Ainsi, grâce à ces informations utilisées pendant les étapes de validation et de vérification du cycle de vie du site internet, l'équipe de R&D peut facilement comparer sa solution avec le cadre (*framework*) théorique de Hayati et Abadi, dans le but de vérifier si les éléments et leurs relations sont correctement pris en compte.

5.3 . Méthodologie d'utilisation

Cette section montre comment mon ontologie, PyCO, peut être instanciée sur des articles existants qui proposent déjà des solutions pour améliorer le respect de la vie privée. Le tableau 5.2 montre comment certains éléments de PyCO peuvent être instanciés depuis des cas d'usage sur lesquels des solutions de la littérature ont été appliquées. Afin de faciliter la comparaison des approches, cette table présente :

— le niveau d'abstraction du système ;

— le domaine d'application ;

— le type de système ;

Table 5.2 – PyCO - Instanciation sur des exemples concrets.

LVL	REF	DOMAINE	SYSTÈME	PROCESSUS	FINALITÉS	DONNÉES PERSON-NELLES	DURÉES DE CONSER-VATION
ML	[132]	Médical	IT**	BPM*	Mots-clés	Mots-clés	∅
	[170]	Médical	Processus de diagnostique	Processus de décision markovien	Fonction de décision	Mots-clés	∅
	[18]	Domotique	Appareil intelligent (IOT)	BPM*	Mots-clés	Mots-clés	∅
PL	[168]	Médical	IT	Fonctions	Mots-clés	Mots-clés	∅
	[70]	Services internet	Site internet	Fonctions	Mots-clés	« Objets »	Mots-clés
	[53]	Ressources humaines	Bases de données	Fonctions	Mots-clés	« Objets »	∅

* Business Process Models

** Technologies de l'information

— les éléments clés modélisés : les processus, les finalités de traitement, les données personnelles et les durées de conservation.

Elle n'inclut pas les autres éléments tels que l'Utilisateur, la Personne concernée, le Responsable des données, le Sous-traitant ou encore l'Autorité de contrôle car ils ne constituent pas des facteurs distinctifs : quand ils sont considérés, ils sont toujours représentés de la même façon (en langage naturel), et ce quel que soit le domaine d'application ou le niveau d'abstraction. La table ne présente pas non plus la Notice ou le Consentement de la personne concernée car, si différence il y a, elle peut se déduire en s'appuyant sur les éléments déjà présentés dans le tableau 5.2 (Données personnelles, Finalités de traitement, Durée de conservation composant la Notice et le Consentement de la personne concernée).

Pour illustrer comment PyCO peut être instancié sur des exemples variés, j'ai choisi trois références pour chaque niveau d'abstraction : [132, 170, 18] pour le niveau modèle (graphique ou formel - **ML**) et [168, 70, 53] pour le niveau programme (**PL**). Ce choix permet d'identifier plus facilement les différences et les similarités que ce soit au sein d'un même niveau d'abstraction ou entre différents niveaux. J'ai sélectionné ces articles car ils présentaient un exemple de système dont le niveau d'abstraction était facilement identifiable.

Grâce à cette classification, on peut remarquer des similarités entre les approches au sein d'un même niveau d'abstraction. Par exemple, au niveau programme (PL), les processus sont toujours représentés par des fonctions (dans les exemples considérés). De même des similarités indépendantes du niveau d'abstraction peuvent être identifiées. Par exemple, les Finalités de traitement sont représentées la majorité du temps à l'aide de mots clés. De telles observations peuvent être

utilisées de plusieurs façons. Cela peut aider à comparer les différentes solutions issues de la littérature. Cela peut également aider à identifier les formalismes déjà utilisés afin de proposer de nouvelles solutions compatibles et/ou comparables avec l'état de l'art. À l'opposé, cela peut aussi être utilisé pour identifier des directions de recherches qui n'ont pas encore été étudiées (ou plus exactement, qui n'ont pas encore donné lieu à des travaux publiés). Cela peut aussi permettre d'identifier les solutions proposant des représentations inhabituelles, telle que la solution proposée par Tschantz et al. [170], qui représente les Finalités de traitement non pas comme des mots clés mais en tant que fonctions de décision.

Cette classification permet aussi d'identifier les éléments qui sont rarement pris en compte. Par exemple, dans les travaux sélectionnés, seul Hayati et Abadi [70] prennent en compte les durées de conservation dans leur solution pour améliorer le respect de la vie privée.


L'instanciation présentée ci-dessus diffère légèrement de celle publiée dans [40]. Le niveau HL (correspondant au niveau RL dans le manuscrit) a été supprimé du tableau, car l'approche a finalement été restreinte aux niveaux ML et PL pour pouvoir proposer une approche formelle.

À partir de ma taxonomie GePyR (présentée en section 4), j'ai défini une ontologie du contexte lié à la vie privée, PyCO. Le but de cette ontologie est d'identifier les éléments à prendre en compte pour vérifier le respect de la vie privée, ainsi que leur relations. Pour cela, je me suis fondée sur des sources variées de la littérature. Pour la définir je me suis restreinte à un sous-ensemble des catégories de GePyR pour me concentrer sur des notions propres à la vie privée. J'ai utilisé cette ontologie sur un premier sous-ensemble d'articles pour identifier quels étaient les éléments pris en compte et comment ils étaient représentés dans leur solution. Cette ontologie me sert de base pour identifier les éléments à prendre en compte dans ma partie suivante, la partie vérification.

6 - Conclusion et perspectives de la partie classification

Ce chapitre conclut la partie classification. Pour cela, les limites des contributions de cette partie sont discutées en section 6.1. Puis, la section 6.2 présente les travaux connexes à ces contributions. Ensuite, la section 6.3 propose différents travaux futurs possibles. Finalement, la section 6.4 conclut cette partie en récapitulant ses points majeurs.

6.1 . Discussion et limites

Cette section discute des choix faits pour mon approche et de leurs limites. Tout d'abord, la section 6.1.1 discute de la sélection des articles que j'ai réalisée pour identifier les différentes représentations de la vie privée. Dans un second temps, la section 6.1.2 explique les choix effectués pour la constitution de la base de données pour réaliser la SMS. Puis, la section 6.1.3 présente les limites des catégories que j'ai définies dans GePyR. Ensuite, la section 6.1.4 rappelle les raisons de la sélection d'un sous-ensemble de catégories de GePyR pour définir PyCO. La section 6.1.5 rappelle les raisons de la différentiation entre processus et finalités de traitement dans PyCO. Finalement, la section 6.1.6 rappelle les raisons de la différentiation entre la personne concernée et l'utilisateur dans PyCO.

6.1.1 . Sélection des articles pour les représentations de la vie privée

Pour identifier les types de représentations de la vie privée dans la littérature, j'ai considéré le RGPD [55], car c'est la source de droit principale en Union européenne pour le traitement des données personnelles tout en respectant la vie privée : tous les traitements de données personnelles en union européenne doivent le respecter [1]. D'autres lois autour du monde concernent les mêmes problématiques, telles que l'*Australian's Privacy Act* [13], ou le *Japan's Act on the Protection of Personal Information* [128] (que je n'ai pas étudiées en détail, cela est laissé en travaux futurs comme détaillés en section 6.3.1). Dans un premier temps, ne garder que le règlement concernant l'Union européenne était suffisant pour identifier un type de représentation supplémentaire : la représentation *via* les Principes.

Je me suis également fondée sur un premier ensemble d'articles sélectionnés par mes encadrants, proposant des solutions pour améliorer le respect de la vie privée. Dans cet ensemble, je n'ai retenu que les articles proposant des définitions et des points de vue pour représenter la vie privée. Par exemple, j'ai retenu l'article de Delaune et al [49] qui définit une propriété d'*Observational Equivalence*, ou

1. https://www.economie.gouv.fr/entreprises/reglement-general-protection-donnees-rgpd

encore l'article de Deng et al. [50] qui identifie un ensemble de menaces envers la vie privée.

Pour les autres articles de cet ensemble, j'ai appliqué la méthode de *snowball search*. Cette méthode correspond à l'utilisation des références et des citations d'une liste d'articles pour identifier d'autres articles pertinents[2]. Il existe deux types de boule de neige (*snowballing*) [180] :

— boule de neige à l'envers (*Backward Snowballing*) : correspond à l'utilisation des références dans un article pour identifier de nouveaux articles à inclure ;

— boule de neige vers l'avant (*Forward Snowballing*) : correspond à l'identification de nouveaux articles à inclure fondée sur ceux citant l'article considéré.

J'ai utilisé les deux techniques : la première pour obtenir les définitions des représentations utilisées (et donc référencées) dans les articles étudiés, par exemple, pour les articles de Solove [159] ou de Pfitzmann et Hansen [133], et la seconde pour identifier les redéfinitions de représentations existantes ou de propositions de nouvelles représentations se comparant à celles existantes. Par exemple, j'ai considéré l'article de Hirschi et al. [71] qui redéfinit une propriété d'*Anonymity* fondée sur celle de Pfitzmann et Hansen. Ces techniques peuvent avoir un biais dû au premier ensemble d'articles sur lequel elles sont appliquées. Pour contrecarrer ce biais, j'ai effectué une SMS en utilisant ces représentations. Cela a ainsi permis de vérifier si d'autres types de représentations étaient nécessaires.

6.1.2 . Source électronique sélectionnée pour la *Systematic Mapping Study*

Lors de ma réalisation de la SMS, j'ai sélectionné une des sources électroniques recommandées par Brereton et al. [28], ce qui pourrait limiter l'analyse réalisée avec la SMS (par le manque d'articles). Je n'ai pas interrogé les autres car cette source fournissait suffisamment d'articles (197 dont 79 après avoir appliqué les critères de sélection) pour pouvoir appliquer la SMS et réaliser une première classification. J'ai gardé cette source, parmi les autres proposées, car tous les articles retournés par ma requête étaient totalement et gratuitement accessibles depuis mon affiliation. J'avais de plus la possibilité d'utiliser une requête structurée et d'ajouter des filtres pour obtenir facilement des documents uniformes (pdf, *research paper*, *proccedings*), comme cela est détaillé en section 4.4.

6.1.3 . Définitions des catégories de GePyR

Les définitions des catégories que j'ai données en section 4.1 peuvent être sujettes à discussion. Tout comme cela est expliqué en section 3.3 pour les représentations, mes définitions étant en langage naturel et s'appuyant sur plusieurs sources, certains peuvent les comprendre différemment de ce que je veux signifier. Par exemple, la catégorie de *caractérisation du Traitement* regroupe des notions liées au traitement des données personnelles, et la catégorie de *caractérisation*

2. https://onderzoektips.ugent.be/en/tips/00002030/

de la Visibilité regroupe des notions liées à la visibilité des données personnelles. Elles sont pourtant disjointes. En effet, la visibilité des données restreintes à des personnes autorisées n'est pas considérée comme un traitement en tant que tel.

Ce genre d'interprétations dû à l'utilisation du langage naturel, se retrouve aussi dans la littérature, comme cela est expliqué en section 3.3. Par exemple, Barati et al. [18] ainsi que Hoepman [74] considèrent la notion de *Data Protection*, mais en interprétant cette notion de façon différente. Pour limiter ces mauvaises compréhensions et aider le lecteur à comprendre leur signification, j'ai fourni des explications et des exemples détaillés dans les sections 4.1, 4.2, 4.3 et 4.4.

6.1.4 . Sélection d'un sous-ensemble de catégories pour PyCO

Comme expliqué en section 5.1, pour définir PyCO, je me suis limitée aux catégories de *caractérisations du Traitement*, *de la Transparence* et *de la Responsabilité*. Je n'ai pas considéré, dans cette ontologie, les éléments nécessaires pour vérifier des propriétés liées aux catégories de *caractérisation de la Visibilité* et *de l'Exactitude des données*. Or, les notions liées à ces catégories ne sont pas propres au domaine de la vie privée : elles existent également dans le domaine de la sécurité, ce qui entraîne que d'autres travaux de ce domaine ont déjà considéré ces notions. Au contraire, les notions liées aux catégories de *caractérisation du Traitement*, de *de la Transparence* et *de la Responsabilité* sont propres à la vie privée.

6.1.5 . Différentiation entre processus et finalités de traitement dans PyCO

Comme expliqué en section 5.1, j'ai modélisé séparément les processus du système et les finalités de traitement, contrairement à ce que l'on peut retrouver dans d'autres ontologies de la littérature (c'est par exemple le cas dans l'ontologie définie par Pandit et al. [126]). J'ai fait le choix de séparer ces deux notions car ces éléments peuvent être différents selon les contextes. En effet, un processus peut avoir été défini pour plusieurs finalités (par exemple, un processus permettant d'accéder à une donnée), et une finalité peut être réalisée à l'aide de plusieurs processus (par exemple, lors du raffinement d'un processus abstrait, défini pour une finalité, en plusieurs processus permettant de réaliser cette finalité). De plus, d'après la sanction de la CNIL contre Google [3], il semble bien nécessaire de différencier les finalités de traitement des processus. En effet, la CNIL a jugé que le manque de transparence de Google sur le traitement des données personnelles lors de la demande de consentement auprès des utilisateurs était dû au fait que le consentement était demandé pour chaque service (i.e. processus) et non pour des finalités spécifiques. Une telle séparation est notamment utile pour garantir que le consentement est « spécifique » et « non ambigu ».

3. https://www.lemonde.fr/pixels/article/2019/01/21/donnees-personnelles-la-cnil-condamne-google-a-une-amende-record-de-50-millions-d-euros_5412337_4408996.html

6.1.6 . Différentiation entre la personne concernée et l'utilisateur dans PyCO

De même, comme expliqué en section 5.1, j'ai modélisé séparément la personne concernée et l'utilisateur du système, contrairement à ce que l'on peut retrouver dans d'autres ontologies de la littérature (comme par exemple dans celle d'Oltramari et al. [125] et Pandit et al. [126]). J'ai fait ce choix car, dans certains cas, la personne concernée utilise directement le système (tel que dans les sites internet) mais, dans d'autres contextes, ce n'est pas elle qui interagit avec le système mais une autre personne (tel que dans les systèmes informatiques des hôpitaux).

6.2 . Travaux connexes

Cette section présente les travaux existants par rapports à mes contributions relatives à la partie classification. La section 6.2.1 présente les travaux concernant les études de l'état de l'art et la section 6.2.2 ceux concernant les ontologies relatives à la vie privée.

6.2.1 . Études de la littérature

Plusieurs enquêtes (*reviews*) se sont déjà intéressés à des problématiques de vie privée [22, 144, 127, 119, 171, 172]. Malheureusement, elles ne couvrent généralement pas le domaine de la vie privée en informatique de manière générale, mais elles ciblent un domaine d'application spécifique [22, 144, 127, 119] ou une technique spécifique [171, 172]. De même, la plupart d'entre elles sont orientées sur des problématiques liées à la visibilité des données, laissant ainsi d'autres notions de vie privée, telle que le consentement, non couvertes. À ma connaissance, seule la récente enquête de Verreydt et al. [175] et la mienne prennent en compte à la fois des notions liées à la visibilité des données et des notions liées au consentement. Cependant, celle de Verreydt et al. se concentre sur le consentement électronique, ses représentations et ses implémentations dans les systèmes de partage de données. Au contraire, mon approche n'adopte pas un point de vue spécifique et propose de s'appuyer sur des représentations variées de la vie privée, détaillées en section 3, pour classer les articles.

De même, plusieurs *systematic mapping studies* (SMS) se sont déjà intéressées à des problématiques de vie privée. Morales et al. [116] réalisent une SMS centrée sur l'étude du respect de la vie privée par conception (*Privacy by Design* - PbD) dans l'ingénierie logicielle. Ils concluent qu'il y a un manque de recherches robustes (*sound*) pour le PbD et que les bonnes pratiques associées ne sont ni totalement développées ni validées. Ils réalisent une autre SMS en 2019 [115] sur le même sujet mais plus précise. Dans la première étude, les catégories de classification utilisées étaient : exigences de logiciel (« *software requirements* ») et conception de logiciel (« *software design* ») alors que dans la seconde les catégories sont plus précises : définition du respect de la vie privée par conception (« *PbD definition* »), principes

ou but du respect de la vie privée par conception (« *PbD principles or goals* ») et pratiques ou techniques du respect de la vie privée par conception (« *PbD practices or techniques* »). Ainsi, grâce à cette seconde étude, ils concluent que la plupart des articles classifiés s'intéressent à la minimisation des données (*data minimization*), mais que ce domaine de recherche est encore très immature et, donc, qu'il y a un fort besoin de contributions supplémentaires dans celui-ci.

Del Alamo et al. [48] présentent également une SMS sur des problématiques de vie privée, mais centrée sur l'étude des politiques de vie privée. Ils concluent que les solutions existantes ont de bonnes performances et que ce domaine de recherche est prometteur, car l'intérêt pour ce domaine est en augmentation dans les articles publiés durant les cinq années précédant la publication de leur article (donc entre les années 2017 et 2022).

Guaman et al. [66] réalisent une SMS sur des techniques de qualité logicielle (*software quality*) appliquées à la détection d'anomalies de vie privée (*privacy related anomalies*). Ils en déduisent que c'est un domaine de recherche dont l'intérêt va grandissant, sous l'influence du RGPD.

Concernant des applications spécifiques, Ebrahimi et al. [54] analysent les solutions concernant la vie privée dans les applications mobiles à l'aide d'une SMS. Ils concluent qu'il y a un grand écart dans les solutions proposées : la plupart proposent des solutions concernant la fuite de données, mais très peu en proposent en lien avec les exigences de la vie privée ou encore les politiques de vie privée et presque aucune ne propose de solution du point de vue utilisateur.

Dans un autre domaine, Alnajrani et al.[5] proposent une SMS sur la protection des données et de la vie privée dans le *mobile cloud computing*. Ils concluent que l'intérêt pour ce domaine de recherche est croissant et que beaucoup de champs ne sont pas couverts, et donc qu'il reste beaucoup de questions de recherche ouvertes.

Enfin, en ciblant un domaine d'application sensible, Iwaya et al. [83] réalisent une SMS sur *mobile health & ubiquitous health*. Ils concluent qu'il y a de fortes disparités dans les domaines de recherche considérés. Par exemple, les techniques de contrôle d'accès et d'authentification sont sur-représentées, tandis que les techniques concernant les politiques de vie privée et la gestion de programmes (*program management*) sont peu considérées.

À contrario, la SMS que j'ai réalisée considère un domaine de recherche plus large, afin d'avoir une vue d'ensemble des notions liées à la vie privée en ingénierie logicielle couvertes dans la littérature.

6.2.2 . Ontologies

Plusieurs ontologies ont été proposées dans la littérature pour représenter la vie privée. Gharib et al. [63] fournissent une ontologie implémentée centrée sur les exigences légales relatives à la vie privée, basée sur une *systematic literature review*. Elle a pour but de fournir un ensemble de concepts pour analyser ces exigences dans leur contexte social et organisationnel.

Par ailleurs, Pandit et al. [126] définissent une ontologie de modèles de concep-

tion (« *design pattern* ») modélisant les données personnelles dans les politiques de vie privée. Leur but est de construire une représentation commune pour des activités utilisant des informations contenues dans des politiques de vie privée, en particulier pour partager ces informations.

Aussi, Loukil [106] propose une approche pour assurer le respect de la vie privée dans l'IoT (*Internet of Things*) selon les lois et règlements. Dans cette approche, une ontologie est définie pour décrire l'environment IoT et les exigences liées au respect de la vie privée dans ce domaine particulier.

Enfin, Oltramari et al. [125] définissent un cadre (*framework*) sémantique pour analyser des politiques de vie privée. Leur but est d'améliorer la compréhension des politiques de vie privée pour les personnes concernées et d'aider les chercheurs et les régulateurs à les analyser. Ils implémentent un outil basé sur ce cadre (*framework*). Même s'ils ne définissent pas l'ontologie dans le seul but de modéliser tous les concepts liés à la vie privée, ils s'appuient sur des concepts liés tels que les durées de rétention des données par exemple.

Trois de ces ontologies [63, 106, 125] ne suivent pas la même méthode que moi. Ils partent des exigences, ou des politiques de vie privées, pour en déduire une ontologie, car leur but est différent de du mien. En effet, à travers mon ontologie, mon but est de modéliser les éléments à prendre en compte pour vérifier le respect de la vie privée, donc avec une orientation vers leur vérification. De manière différente, le but d'Oltramari et al. est orienté utilisateur, tandis que celui de Gharib et al. est centré sur les exigences de la vie privée dans leurs contextes sociaux et organisationnel et que celui de Loukil est ciblé sur les exigences spécifiques de l'IoT.

Parmi les ontologies présentées précédemment, celle de Pandit et al. [126] est celle qui est la plus proche de la mienne en raison de sa méthode, qui est similaire : partir du système et des solutions existantes dans la littérature pour définir l'ontologie. Nous partageons le même but et les deux ontologies ont plusieurs éléments en commun. Il subsiste cependant des différences. Tout d'abord, leur approche ne différencie pas les processus et les finalités de traitement, comme discuté en section 6.1. Une autre différence est qu'ils prédéfinissent des types de processus (par exemple, la collecte de données), ce qui n'est pas le cas de PyCO. Cependant, grâce à sa généricité, mon ontologie peut aussi représenter ces processus spécifiques. L'approche de Pandit et al. se concentre sur ce qui est nécessaire pour les politiques de vie privée, tandis que je considère un contexte plus large, incluant par exemple, les acteurs intervenant dans le respect de la vie privée.

Certaines approches proposent des études (*surveys*) d'ontologies légales. Ainsi, Kurteva et al. proposent une étude sur les ontologies centrées sur le consentement, tel que défini dans le RGPD [95]. Ils fournissent une vue d'ensemble (*overview*) détaillée présentant les spécificités du consentement et ses aspect légaux. En complément de cette vue d'ensemble, ils définissent des pratiques pour assurer la confor-

mité envers le consentement. De leur côté, Leone et al. fournissent une analyse comparative d'ontologies légales récentes [98]. Ils comparent en détail ces ontologies sur des critères spécifiques tels que les langages utilisés ou les modèles de norme adoptés. Cependant, les deux se concentrent seulement sur l'aspect légal. En particulier, aucune n'identifie les éléments communs de vie privée représentés dans ces ontologies. Dans mon approche, je propose une représentation plus générique de la vie privée qui, même si elle est moins détaillée et moins spécifique, permet d'identifier les éléments principaux à prendre en compte pour vérifier le respect de la vie privée ainsi que leurs relations.

Ainsi, les approches proposées dans la littérature sont spécifiques : soit selon un point de vue ou une représentation particulière, soit selon un domaine de recherche spécifique, soit encore selon des techniques précises. À ma connaissance, aucune autre approche que la mienne ne propose une solution aussi générique et avec une considération aussi large de la vie privée en informatique. Cependant, en adoptant des points de vue spécifiques, ces approches sont probablement plus précises et détaillées que mon approche générique, car elles considèrent des éléments qui n'existeraient pas dans d'autres contextes.

6.3 . Travaux futurs

Cette section présente la suite des travaux envisagée. Actuellement, ceux envisagés sont :

- l'analyse d'autres lois et règlements concernant la vie privée, présentée en section 6.3.1 ;
- l'extension à la SMS réalisée utilisant GePyR pour considérer d'autres bases de données recommandées pour réaliser de telles études, détaillée en section 6.3.2 ;
- la réalisation d'une SMS en utilisant PyCO, détaillée en section 6.3.3 ;
- une implémentation pour GePyR, détaillée en section 6.3.4 ;
- une implémentation pour PyCO, présentée en section 6.3.5 ;
- l'ajout des catégories de GePyR dans un outil existant lié à la gestion des exigences, présenté en section 6.3.6.

Pour chacune de ces implémentations (pour GePyR et pour PyCO), un schéma d'utilisation et une proposition d'interface graphique sont fournis. Le niveau de détails, donné plus bas, est influencé par l'état d'avancement de ces idées. Pour chacune de ces perspectives, j'identifie la limite actuelle, l'idée pour dépasser celle-ci, puis je détaille les pistes de mise en œuvre pour réaliser cette idée.

6.3.1 . Études d'autres lois et règlements

Limite : actuellement, j'ai utilisé seulement le RGPD comme source légale pour identifier les types de représentations de la vie privée en informatique, ce qui pourrait impacter GePyR et PyCO qui ne seraient pas applicables à d'autres textes légaux.

Idée : l'idée serait d'étudier plus en détail d'autres lois et règlements pour vérifier si GePyR couvre les notions de vie privée considérées dans ceux-ci, et si des éléments sont à ajouter dans PyCO pour prendre en compte ces sources.

Pistes de mise en œuvre : le principe serait de sélectionner d'autres textes légaux établis dans d'autres parties du monde, tels que l'*Australian's Privacy Act* [13] ou le *Act on the Protection of Personal Information* japonais [128], d'identifier si toutes les notions définies dans ces textes sont pris en compte dans les catégories de GePyR, de voir si cela ajoute un nouveau type de représentation (autre qu'une représentation *via* les principes), et si des éléments sont à ajouter dans PyCO.

6.3.2 . Extension de la SMS utilisant GePyR

Limite : actuellement, j'ai réalisé une SMS avec GePyR à partir d'une seule base de donnée sélectionnée, l'ACM Digital library.

Idée : l'idée serait d'étendre la SMS déjà réalisée avec GePyR et détaillée en section 4.4 en interrogeant des bases de données supplémentaires afin d'analyser un plus grand nombre d'articles. Ainsi, cela permettrait de voir si cela modifierait les observations faites sur la classification dans la SMS déjà réalisée.

Pistes de mise en œuvre : pour cela, le principe serait de suivre les recommandations de Brereton et al. [28], qui conseillent d'utiliser également les bases suivantes : IEEExplore, Google scholar, Citeseer library, Inspec, ScienceDirect et EI Compendex.

6.3.3 . Réalisation d'une SMS utilisant PyCO

Limite : actuellement, j'ai réalisé une SMS avec GePyR, ce qui permet d'utiliser cette contribution pour analyser la littérature de façon structurée, en s'appuyant sur les catégories de GePyR. Néanmoins, je n'ai pas utilisé, pour le moment, PyCO, qui permet également de modéliser des éléments à prendre en compte pour vérifier le respect de la vie privée en informatique. Actuellement, je n'ai appliqué mon ontologie que sur six articles de la littérature. Même si cela permet d'illustrer le mécanisme d'utilisation de PyCO, cela ne permet pas de réaliser des observations robustes. Pour cela, une SMS pourrait être une solution pertinente.

Idée : l'idée serait de suivre une méthode similaire à celle utilisée pour la SMS utilisant GePyR, afin d'identifier quels sont les éléments modélisés par PyCO dans les différentes approches de la literature, et comment ils sont représentés. Cela permettrait ainsi d'étendre et d'affiner les observations faites sur l'exemple d'utilisation de PyCO.

Pistes de mise en œuvre : le principe serait de reprendre la méthode détaillée en section 4.4 mais en utilisant PyCO, détaillée en section 5.1.

6.3.4 . Implémentation pour GePyR

Limite : actuellement l'utilisation de GePyR pour identifier un ensemble d'articles considérant des représentations ou des notions liées à une même catégorie est manuelle.

Idée : l'idée serait d'implémenter un outil permettant d'automatiser l'utilisation de GePyR.

Pistes de mise en œuvre : le principe d'utilisation de cet outil est représenté sur la figure 6.1. Un prototype d'interface graphique possible pour cet outil est illustré sur la figure 6.2. Ainsi, l'utilisateur à la possibilité de sélectionner un ensemble de catégories (une ou plusieurs) correspondant aux notions qui l'intéressent en fonction de sa problématique, ces catégories étant définies en section 4.1. Il peut, sinon, sélectionner un ensemble de représentations de la vie privée qui l'intéressent en fonction de sa problématique, ces dernières étant présentées en section 3. L'utilisateur peut également sélectionner à la fois des catégories et des représentations. À partir de cette sélection, l'outil interroge alors une base de données contenant l'ensemble des articles associés à des labels correspondant aux catégories et aux représentations. Ainsi, l'utilisateur peut facilement avoir accès à des articles proposant des solutions ciblant des problématiques similaires à celle qui l'intéresse.

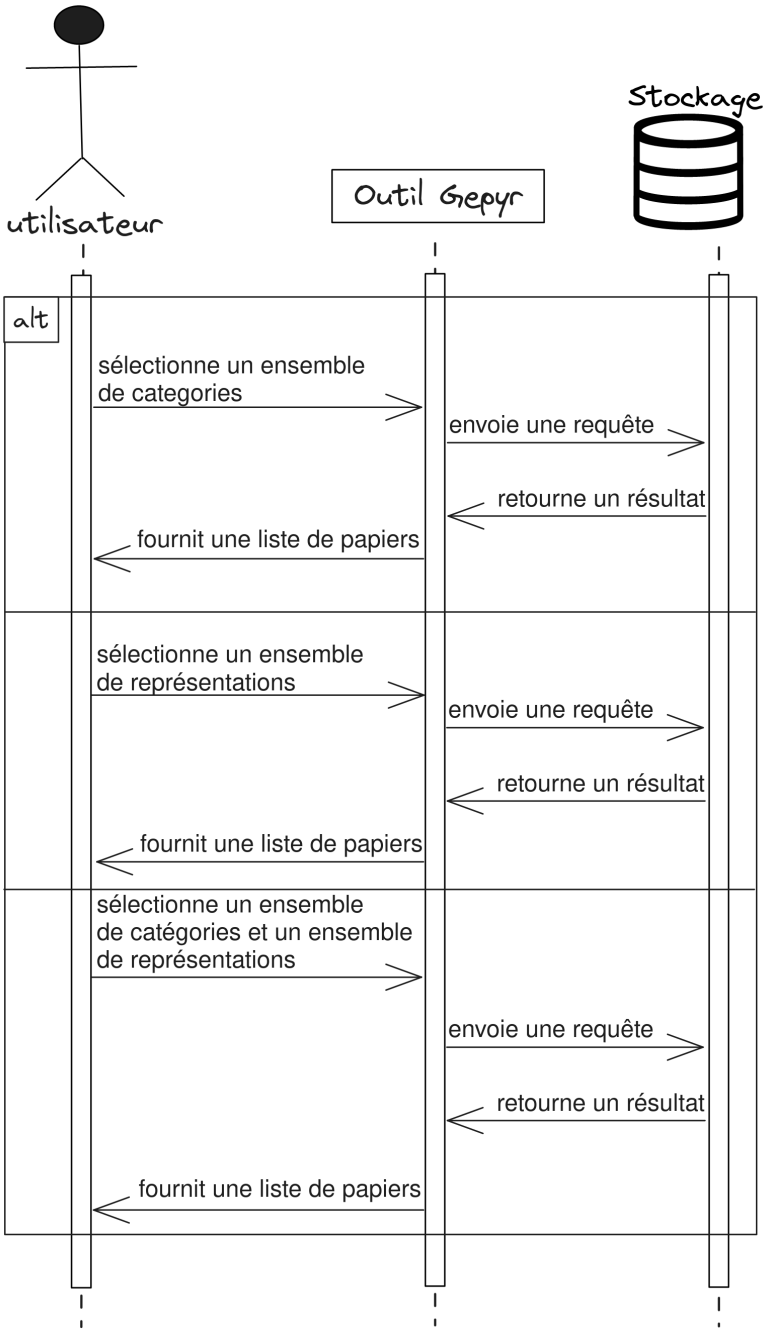


Figure 6.1 – Exemple de diagramme de séquence fondé sur un outil futur pour GePyR.

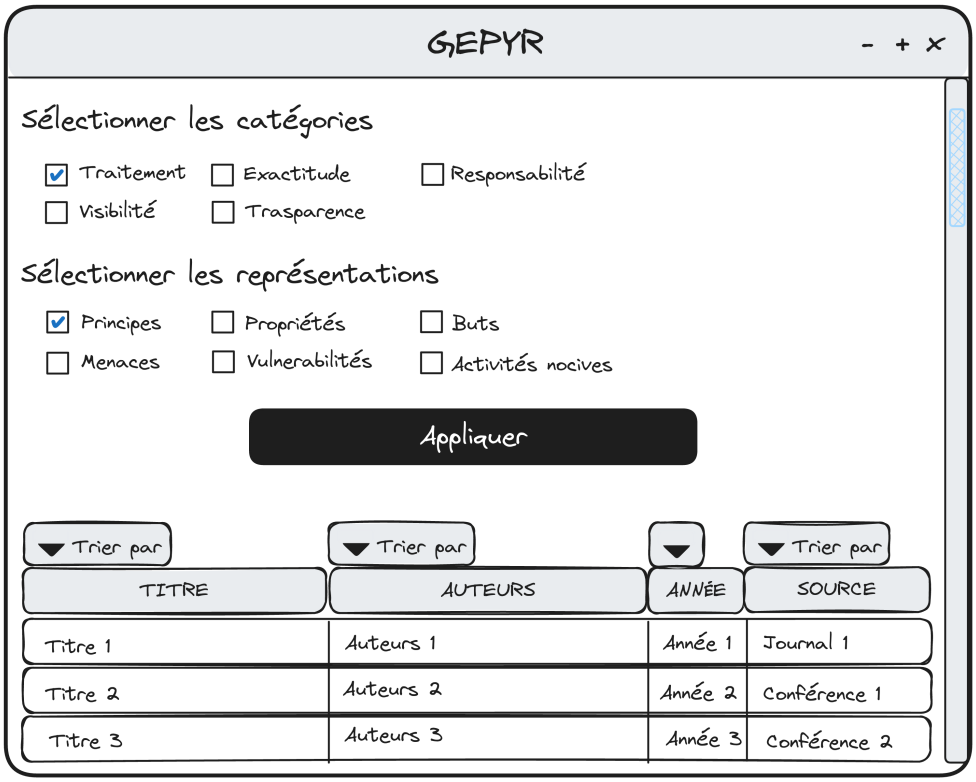


Figure 6.2 – Maquette d'interface graphique d'un outil futur pour GePyR.

6.3.5 . Implémentation pour PyCO

Limite : actuellement, l'utilisation de PyCO pour identifier l'instanciation de ses éléments à partir d'articles de la littérature se fait manuellement.

Idée : l'idée serait similaire à GePyR, c'est-à-dire implémenter un outil permettant d'automatiser l'utilisation de PyCO.

Pistes de mise en œuvre : le principe d'utilisation de cet outil est représenté sur la figure 6.3 et la maquette sur la figure 6.4. L'utilisateur sélectionne un ensemble d'articles qu'il souhaite analyser parmi les articles proposés par l'outil. L'outil récupère ensuite les informations liées aux articles sélectionnés depuis une base de données. Ces informations correspondent alors à l'instanciation des éléments de PyCO dans ces articles (*i.e.* lesquels sont considérés et comment ils sont représentés).

Une possibilité serait de combiner les deux outils en un seul. Ainsi, après avoir recherché un ensemble d'articles à l'aide de GePyR, l'utilisateur pourrait utiliser PyCO directement dessus.

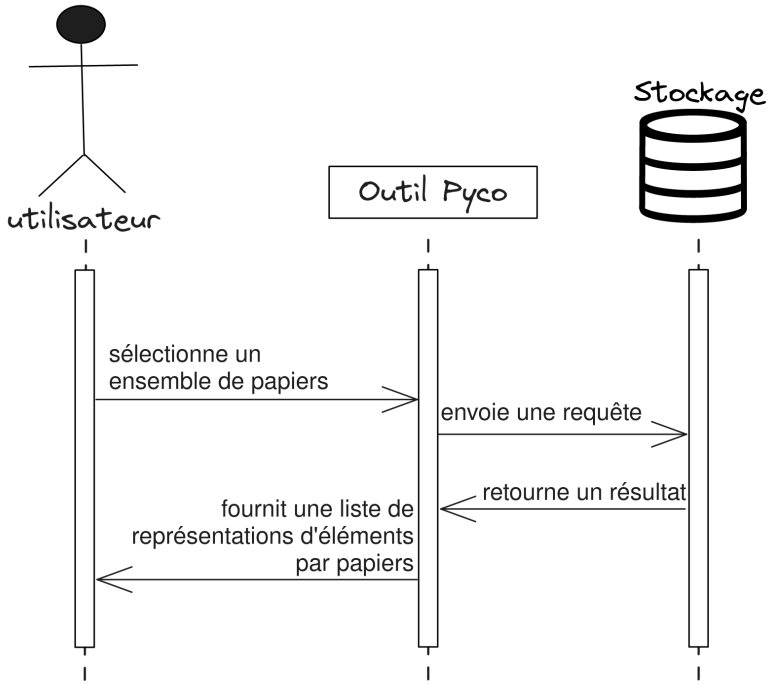


Figure 6.3 – Exemple de diagramme de séquence fondé sur un outil futur pour PyCO.

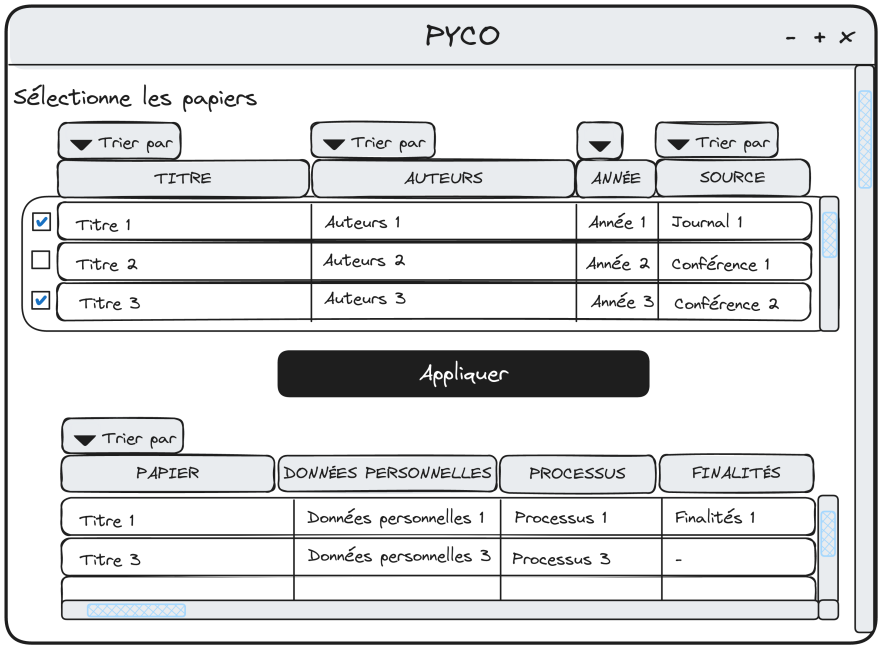


Figure 6.4 – Maquette d'interface graphique d'un outil futur pour PyCO.

6.3.6 . Ajout des catégories de GePyR dans un outil existant de gestion des exigences

Limite : Actuellement, GePyR est n'est utilisé que pour classer des articles afin de les comparer, mais les catégories de GePyR pourraient être utilisées dans un autre but.

Idée : L'idée serait d'ajouter les catégories définies dans GePyR dans l'outil MaatREQ [17], un outil de formalisation d'exigences, afin de spécifier les exigences qui sont en rapport avec le respect de la vie privée.

Pistes de mise en œuvre : Cet outil permet de formaliser les exigences de façon structurée comme illustré sur la figure 6.5 (provenant de [17]). MaatREQ permet de transformer de façon automatique ces exigences structurées en algèbre de processus [14] pour les valider formellement par simulation. Cette validation a pour but d'aider l'utilisateur à confirmer la cohérence des exigences spécifiées et à les réviser si besoin.

R1 : **when** set button is pressed, **the** system **shall** activate the alarm **immediately after** 60s

R2 : **after** the alarm activation, **when** motion is detected, **the** system **shall** emit a tone **immediately**

Figure 6.5 – Exemple d'exigences structurées dans MaatREQ [17].

L'ajout des catégories de GePyR dans cet outil permettrait de spécifier les exigences qui sont en rapport avec le respect de la vie privée, avec une séparation suffisamment fine pour identifier des contraintes. Par exemple, une exigence spécifiée avec la catégorie *caractérisation de la Visibilité* aurait comme contrainte de devoir spécifier qui a le droit d'accéder aux données personnelles, tandis qu'une exigence spécifiée avec la catégorie *caractérisation de l'Exactitude des données* devrait indiquer quelles sont les mesures prises pour garantir la justesse des données personnelles utilisées. Cet exemple ne présente pas de façon exhaustive les contraintes que doivent respecter les exigences. Identifier l'ensemble de ces contraintes selon les catégories de GePyR est toutefois inclus dans cette perspective.

6.4 . Conclusion

Cette section conclut la partie classification de cette thèse. Le but de mes travaux est d'aider à améliorer le respect de la vie privée. Plus particulièrement, mes contributions liées à la partie classification de la vie privée, permettent d'identifier ce que représente la vie privée en informatique, dans la littérature, et ce qu'il faut prendre en compte pour vérifier qu'elle est respectée.

Pour cela, j'ai identifié des types de représentation de la vie privée dans la littérature, détaillés et discutés en section 3. Ainsi, la vie privée peut être considérée via des Principes, des Propriétés, des Buts, mais également via des Menaces, des Vulnérabilités ou bien encore via des Activités Nocives. Pour identifier ces représentations, je me suis fondée sur des sources variées, telles que des textes légaux, des taxonomies, des terminologies, ou bien encore simplement des articles proposant des solutions qui pour cela définissent, ou redéfinissent, leur façon d'appréhender la vie privée. Ces représentations variées permettent de considérer la vie privée sous différents aspects, et donc de considérer la grande variété de problématiques qui lui sont liées. Par contre, certaines représentations, bien que différentes (par exemple, des menaces contre lesquelles il faut protéger le système et des propriétés que le système doit respecter), font référence à des notions similaires. Des articles qui considèrent des représentations différentes peuvent donc en réalité répondre à des problèmes similaires, ce qui rend difficile la comparaison d'articles dans la littérature.

Pour permettre une telle comparaison, j'ai définie une taxonomie générique, GePyR, détaillée en section 4. Cette taxonomie est fondée sur des catégories permettant de regrouper des représentations s'appuyant sur des notions similaires. Ces catégories sont au nombre de cinq : les *caractérisations du Traitement*, *de la Visibilité*, *de l'Exactitude des données*, *de la Transparence* et du *de la Responsabilité*. Selon les points de vue et les représentations considérées, ces catégories génériques peuvent être spécialisées dans ces représentations spécifiques (par exemple, la *caractérisation du Traitement* peut être spécialisée comme le principe de *limitation des finalités*). Cette taxonomie permet de classer les articles de la littérature. Ainsi,

j'ai utilisé GePyR dans une SMS afin d'identifier des tendances et des manques du domaine de recherche concernant la vie privée en informatique, en m'appuyant sur des métriques pré-identifiées. Pour cela, je me suis intéressée aux questions de recherches suivantes (plus de détails sur leurs réponses à l'aide de la SMS sont disponibles en Section 4.4) :

- **RQ1** : Quelle est la couverture des notions liées à la vie privée ?
  Les notions liées à la visibilité des données personnelles sont – de loin – les plus couvertes ; les moins couvertes sont celles liées au traitement des données personnelles.
- **RQ2** : Quelles sont les représentations de la vie privée qui sont les plus considérées et les moins considérées ?
  La représentation la plus considérée est celle *via* les menaces. Les représentations *via* les buts et *via* les activités nocives sont, quant à elles, assez peu considérées.
- **RQ3** : Quel type de représentation est le plus utilisé selon les notions considérées ?
  Les types de représentations utilisés varient selon les notions considérées. Par exemple, les notions liées à la visibilité sont considérées selon tous les types de représentation, avec une prédominance de la représentation *via* les menaces, tandis que les notions liées à la responsabilité ne sont couvertes que par la représentation *via* les principes.
- **RQ4** : Existe-t-il beaucoup d'outils pour les informaticiens pour les aider à respecter la vie privée ?
  Au vu de la SMS, il existerait peu d'outils corrects (*sound*) pour cela. Dans la majorité des cas, les solutions ne reposent que sur des outils ad-hoc ou bien sur aucun outil.
- **RQ5** : Quelle est l'évolution des publications sur l'amélioration du respect de la vie privée ?
  Le nombre de publications semble suivre les arrivées de nouvelles lois et règlements. Leurs arrivées semblent augmenter l'intérêt pour ce domaine de recherche, qui retombe légèrement avec le temps.

À partir de cette taxonomie, j'ai défini une ontologie du contexte lié à la vie privée, PyCO, détaillée en section 5. Le but de cette ontologie est d'identifier les éléments à prendre en compte pour vérifier le respect de la vie privée, ainsi que leurs relations. Pour cela, je me suis fondée sur des sources variées de la littérature, sur le RGPD et également sur des articles proposant des solutions pour améliorer le respect de la vie privée en informatique. Pour la définir, je me suis restreinte à un sous-ensemble des catégories de GePyR afin de pouvoir me concentrer sur des notions propres à la vie privée. Je me suis ainsi appuyée sur des articles classés dans ces catégories et j'ai pu identifier les éléments suivants à prendre en compte :

- les données personnelles ;

— la personne concernée ;
— le responsable du traitement ;
— le sous-traitant ;
— le système et ses processus ;
— l'utilisateur ;
— les finalités de traitement ;
— les durées de conservation ;
— le consentement de la personne concernée ;
— la notice ;
— l'autorité de contrôle.

Même si je n'ai pas réalisé de SMS à partir de cette ontologie, je l'ai utilisée sur un premier sous-ensemble d'articles pour identifier quels étaient les éléments pris en compte et comment ils étaient représentés dans leur solution. Cette ontologie m'a servi de base pour identifier les éléments à prendre en compte dans ma partie vérification.

Ces contributions ont certaines limites. Ces limites peuvent être dûes à des choix que j'ai faits, par exemple la sélection des articles pour définir ma taxonomie ou bien encore les catégories sélectionnées pour définir mon ontologie. Elles sont discutées dans la section 6.1 ainsi que certains choix que j'ai faits dans la définition de mes contributions, comme par exemple le fait de séparer, dans PyCO, les processus des finalités de traitement.

Bien que plusieurs approches aient été proposées pour modéliser les notions liées à la vie privée et d'autres pour classifier la littérature, comme présenté en section 6.2, celles-ci sont spécifiques : soit selon un point de vue ou une représentation (par exemple les menaces), soit selon un domaine de recherche spécifique (tel que les politiques de vie privée ou bien encore l'IoT), soit encore selon des techniques (par exemple de qualité logicielle). À ma connaissance, aucune autre approche ne propose une solution aussi générique avec une considération aussi large de la vie privée en informatique.

Enfin, j'ai identifié plusieurs suites de travaux possibles, détaillées en section 6.3 : une analyse d'autres lois et règlements concernant la vie privée, une extension à la SMS réalisée pour prendre en compte d'autres bases de données recommandées pour réaliser des SMS, la réalisation d'une SMS en utilisant PyCO, une implémentation pour GePyR, une autre pour PyCO, et l'ajout des catégories de GePyR dans un outil existant de gestion des exigences.

Ces travaux pour la classification de la vie privée constituent une première étape qui permet d'avoir une vue globale de la littérature concernant la vie privée en informatique. Ils forment une base suffisante pour aider à réaliser les travaux pour la vérification de systèmes comme détaillé dans la partie suivante.

Deuxième partie

VÉRIFICATION DE PROPRIÉTÉS DE RESPECT DE LA VIE PRIVÉE

Cette partie présente mes travaux concernant la vérification de propriétés liées au respect de la vie privée. Elle présente mes différentes contributions ainsi que des cas d'utilisation me permettant d'illustrer et d'évaluer ces contributions.

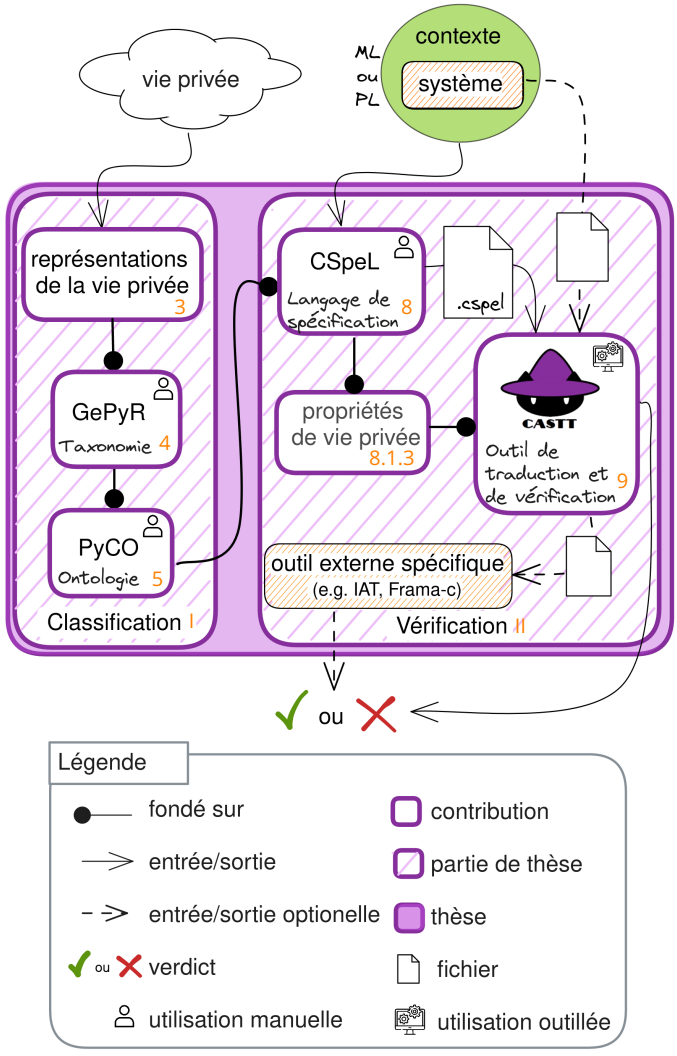


Figure 6.6 – Rappel - Lien entre les parties classification et vérification.

Comme rappelée sur la figure 6.6, mon approche est séparée en deux parties : la partie classification reposant sur les contributions GePyR et PyCO d'une part, et la partie vérification reposant sur CSpeL et CASTT d'autre part.

Grâce à GePyR j'ai pu identifier plusieurs catégories permettant de regrouper des représentations de la vie privée existantes dans la littérature. La catégorie qui m'intéresse pour les travaux concernant cette nouvelle partie est la *caractérisation du Traitement* (des données personnelles) et plus particulièrement sa représentation

sous forme de propriétés. Comme vu en section 4.1, la *caractérisation du Traitement* regroupe un ensemble de notions différentes qui peuvent varier selon le type de représentation de la vie privée choisi (par exemple, Propriétés ou Principes). De plus, ces notions peuvent être encore séparées en sous-catégories : celles concernant les finalités (*purpose*), la nécessité des données (*necessity*), la durée d'utilisation des données (*duration*) et l'évolution du consentement (*consent evolution*). Les propriétés formelles liées au traitement que je définis correspondent à ces sous-catégories.

L'ontologie PyCO, présentée en section 5, modélise les différents éléments liés au traitement des données personnelles ainsi que leurs relations afin d'identifier les éléments à prendre en compte pour vérifier qu'un système respecte la vie privée. Le langage CSpeL se fonde sur ces éléments pour définir un modèle formel.

Comme vu en section 1.1.2, je considère que le système peut être représenté à différents niveaux d'abstraction pour la vérification. La figure 6.7 illustre un cycle de vie en V avec différents niveaux. Les différents niveaux qui ont été pris en compte pour ces travaux sont le niveau Modèle (ML) et le niveau Programme (PL). Le niveau Requirement (RL) n'a pas été retenu dans la version actuelle des travaux, car il ne permettait pas de proposer une approche formelle. Pour le niveau Modèle, deux sous-niveaux ont été définis : le niveau Modèle Graphique (ML Graphique), qui implique l'utilisation d'un langage de modélisation graphique, et le niveau Modèle Formel (ML Formel), qui implique l'utilisation d'un langage de modélisation formel.

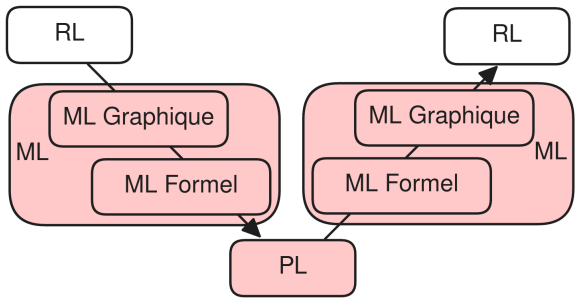


Figure 6.7 – Exemple de cycle de vie avec différents niveaux d'abstraction.

Pour résumer, le but des contributions présentées dans cette partie est de vérifier des propriétés de vie privée à différentes étapes du cycle de vie d'un système et cela à l'aide d'un seul et même langage. Pour atteindre ce but, comme présenté sur la figure 6.8, j'ai réalisé les contributions suivantes :

— CSpeL, un langage formel permettant de spécifier un système dans son contexte lié à la vie privée, en s'appuyant sur un modèle formel composé :
  - de l'ensemble des processus du système ;

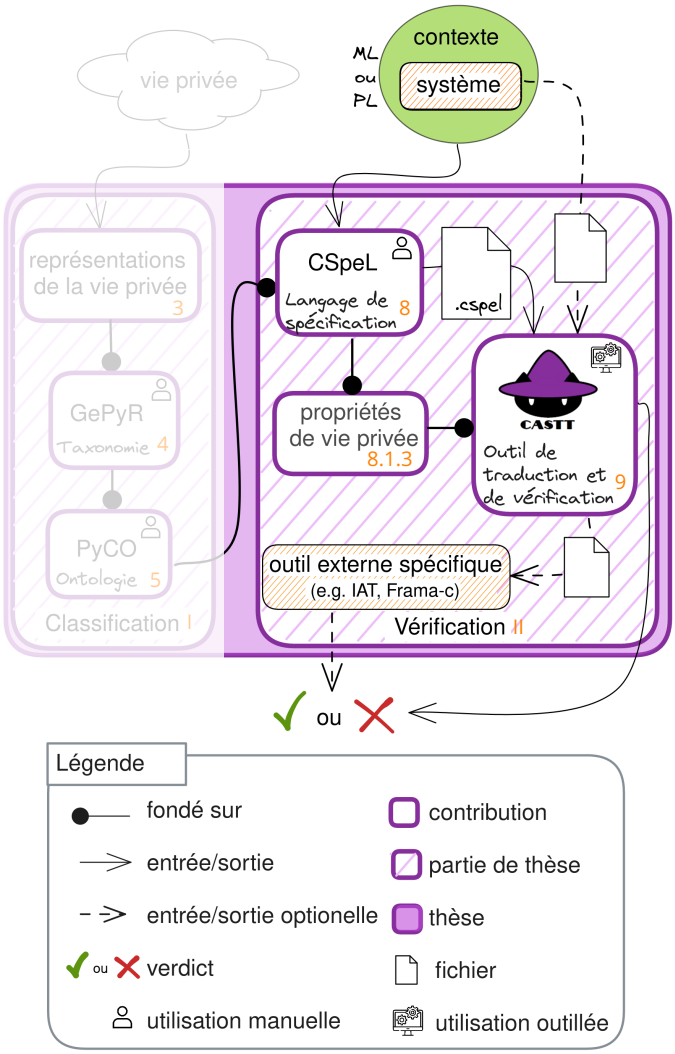


Figure 6.8 – Vérification outillée du respect de propriétés de vie privée.

- de l'ensemble des données personnelles traitées ;
- de l'ensemble des finalités de traitement ;
- de la relation de consentement (quelle finalité de traitement a été consentie pour des données personnelles) ;
- de la relation d'association des finalités de traitement aux processus ;
- de la relation représentant les données personnelles nécessaires aux processus ;

et permettant de spécifier des traces d'exécutions de ce système ;

— la formalisation de propriétés liées au respect de la vie privée, se fondant sur

le formalisme de CSpeL. La propriété de *conformité aux finalités consenties* et celle de *conformité à la nécessité des données* ;

— CASTT, un outil permettant de vérifier ces propriétés sur un système à partir de la spécification fournie en CSpeL et pouvant être utilisé conjointement avec un outil spécifique de vérification (dépendant du niveau d'abstraction considéré). Cette vérification peut être réalisée sur des traces spécifiées en CSpeL ou directement sur le système au niveau Modèle Formel ou au niveau Programme à l'aide de mécanismes de traduction vers des outils de vérification formelle spécifiques à ces niveaux.

La section 7 présente les cas d'utilisation permettant d'illustrer et d'évaluer ces travaux. Dans un second temps, la section 8 détaille le langage de spécification CSpeL et les propriétés que j'ai définies pour pouvoir vérifier qu'un système respecte la vie privée, selon la spécification fournie en CSpeL. Ensuite, la section 9 présente l'outil CASTT qui permet de vérifier de façon outillée que le système respecte les propriétés précédentes, ainsi que son évaluation. Enfin, la section 10 analyse ces travaux et leurs limites, fournit des pistes et des idées de travaux futurs, compare ces travaux à l'état de l'art et conclut cette partie.

7 - Cas d'utilisation

Cette section présente les exemples utilisés pour illustrer et évaluer mon approche de vérification. J'ai choisi deux domaines d'applications dans lesquelles des problématiques de vie privée sont à prendre en compte. Le premier domaine d'application concerne la santé et est détaillé en section 7.3. Le deuxième concerne les sites web et est détaillé en section 7.4. Les exemples peuvent être décrits à trois niveaux d'abstraction : au niveau modèle graphique, au niveau modèle formel ou au niveau programme en utilisant le langage C. Dans un premier temps, faisons une rapide présentation en section 7.1 des langages de modélisation utilisés, ainsi que du principe de raffinement en section 7.2.

7.1 . Langages de modélisation utilisés

Cette section présente les langages utilisés pour mes exemples au niveau modèle. Pour le niveau modèle graphique, la notation BPMN [30] est utilisée et est présentée en section 7.1.1, tandis que pour le niveau modèle formel, celle utilisée, présentée en section 7.1.2, est fondée sur le langage d'interaction formel utilisé par l'outil IAT [109].

7.1.1 . Langage BPMN

BPMN [30] est un langage de modélisation graphique pour spécifier des processus et leur déroulement (*workflow* en anglais). Chaque processus BPMN P_i modélisé contient un évènement initial S_i (*Event Start*), un évènement terminal E_i (*Event End*) et différentes tâches T_{ij} (*Activity*). Ces éléments sont connectés les uns aux autres par un flux de séquence (*Sequence Flow*). Certains utilisent des données, ce qui est représenté par une association sur le flux de séquence (*Data Association*). Les représentations graphiques de ces éléments sont disponibles dans le tableau 7.1.

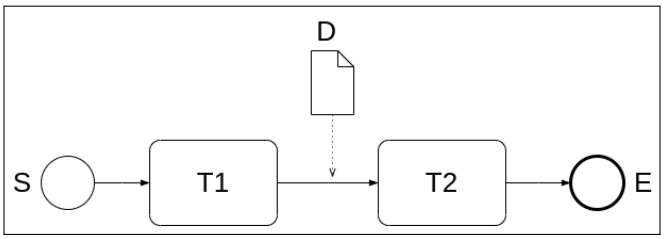
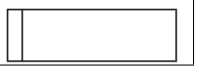


Figure 7.1 – Exemple simple d'utilisation de BPMN.

La figure 7.1 illustre un exemple simple reprenant ces différents éléments. Cette figure modélise le processus déclenché par l'évènement initial S. Une fois déclenché,

le processus effectue la tâche $T1$, puis la tâche $T2$, ce qui déclenche l'évènement terminal E. De plus, la tâche $T1$ transfère la donnée D à la tâche $T2$, la donnée étant donc utilisée par les deux tâches. Des éléments supplémentaires sont utilisés dans la version Gestion Données Médicales - version Complexe (section 7.3.2). Ils sont détaillés ci-après : différents types d'évènements initiaux (multiple ou avec message), de branchements (exclusif ou inclusif), d'évènements supplémentaires (message ou erreur), de tâches (avec sous-processus), de flux entre les activités (flux de messages), et également des corridors. Le Tableau 7.1 présente un sous ensemble des éléments de BPMN utilisés dans cette thèse. Plus d'informations sur les éléments BPMN sont disponibles dans [30].

Table 7.1 – Éléments BPMN utilisés dans mes exemples.

Évènement initial (*Event Start*)	
Évènement initial multiple (*Multiple Start*)	
Évènement de message initial (*Message Start*)	
Évènement d'erreur (*Event Error*)	
Évènement terminal (*Event End*)	
Évènement de message terminal (*Message End*)	
Branchement exclusif (*Gateway Exclusive*)	
Branchement inclusif (*Gateway Inclusive*)	
Tâche (*Activity*)	
Tâche avec sous processus (*with Sub-Process Collapsed*)	
Flux de séquence (*Sequence Flow*)	
Association de données (*Data Association*)	
Flux de message (*Message Flow*)	
Donnée (*Data Object*)	
Corridors (*Lane*)	

7.1.2 . Langage d'interaction de IAT

IAT [109] est un outil permettant l'analyse de traces et de multi-traces représentant l'exécution de systèmes distribués par rapport à des modèles d'interaction.

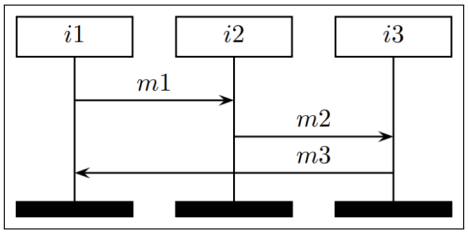


Figure 7.2 – Exemple d'interaction issu de la thèse d'Erwan Mahé [109].

La figure 7.2 représente de façon graphique les interactions entre les instances de sous-systèmes i_1, i_2 et i_3 dans un système. Ces interactions sont l'envoi et la réception des messages m_1, m_2 et m_3.

Pour permettre une analyse formelle de ce genre d'interaction, IAT utilise une formalisation de langages d'interaction, habituellement graphiques (tels que sur la figure 7.2), fondée sur de l'algèbre de termes et donc s'appuyant sur une sémantique formelle. Les langages d'interaction permettent de modéliser le comportement de systèmes en spécifiant des échanges de messages entre sous-systèmes (lignes de vie). Le comportement du système modélisé est exprimé en tant que traces d'exécution. Chaque trace est une séquence d'évènements atomiques, un évènement étant soit l'émission, soit la réception d'un message. Un ensemble de traces forme la sémantique d'une interaction.

Dans IAT, les interactions des systèmes sont modélisées à l'aide d'une spécification. Plusieurs fichiers distincts correspondent à cette spécification formelle :

— un fichier .imf qui contient les informations relatives au modèle du système et aux conditions de l'analyse ;

— un fichier .mtf qui contient une instanciation d'échanges de messages dans le système, constituant une trace d'exécution.

La table 7.3 présente une version simplifiée de la grammaire du langage d'interaction formel de IAT pour un fichier .imf. De la même façon que pour le langage BPMN, je ne présente ici que les éléments nécessaires pour mes exemples. Plus de détails sur ce langage sont disponibles dans [109].

imf	::=	S O M
S	::=	`@signature`{ L M_{ess} V I }
O	::=	`@option`{ ensemble d'options }
M	::=	`@model`{ M_{od} }
L	::=	`@lifetime`{ ensemble de lignes de vie }
M_{ess}	::=	`@message`{ ensemble de messages }
V	::=	`@variable`{ ensemble de variables }
I	::=	`@init`{ ensemble d'initialisations }
M_{od}	::=	Envois | op(M_{od}) | op(M_{od}), M_{od}
Envois	::=	Envoi | Envoi, Envois
Envoi	::=	$l_i - -m_k \rightarrow l_j$ | $l_i - -m_k(v_h) \rightarrow l_j$
op	::=	`loopH` | `seq` | `alt`

Table 7.2 – Grammaire simplifiée du langage d'interaction de IAT.

Comme présenté dans cette grammaire, le fichier .imf contient un bloc contenant la spécification des éléments du système (`@signature`), un bloc d'options pour l'analyse (`@option`), et un bloc modélisant les échanges de messages dans le système (`@model`).

Le bloc de spécification des éléments du système contient lui même différents blocs, le premier spécifiant les différentes lignes de vie utilisées (`@lifeline`), le deuxième pour les messages pouvant être émis/reçus (`@message`), le troisième pour les déclarations de variables (`@variable`) et le quatrième pour l'initialisation (`@init`) de ces dernières. Je ne détaille pas ici les différentes options (`@option`) possibles, car ce n'est pas le coeur de mon approche, mais ces informations peuvent être trouvées dans [109].

Le bloc modélisant les échanges de messages dans le système est composé d'envois de messages, de la forme $l_i - -m_k \rightarrow l_j$ (représentant $l_i \xrightarrow{m_k} l_j$), avec l_i et l_j étant des lignes de vie et m_k un message. Ce message peut contenir une variable émise, v_h, le message prenant alors la forme $l_i - -m_k(v_h) \rightarrow l_j$ (représentant $l_i \xrightarrow{m_k(v_h)} l_j$). Lorsque cette variable est partagée entre différentes lignes de vie, on le spécifie en ajoutant £ devant le nom de la variable lors de son utilisation. Ces envois sont organisés à l'aide de différents opérateurs. Ceux que j'utilise dans ma thèse sont : la séquence (`seq`(e_1, ..., e_n)), le choix (`alt`(e_1, ..., e_n)), et la boucle (`loopH`(e_1, ..., e_n)), les e_i ($1 \leq i \leq n$) étant des envois ou des envois englobés par un opérateur.

mtf	::=	{ Trace }
Trace	::=	Ligne_de_ vie | Ligne_de_ vie ; Trace
Ligne_de_ vie	::=	$[l_i]$ Interaction
Interaction	::=	Message | Message . Interaction
Message	::=	$l_i?m_k$ | $l_i!m_k$ | $l_i?m_k(v_h)$ | $l_i!m_k(v_h)$

Table 7.3 – Grammaire simplifiée du langage pour les traces d'exécution de IAT.

La table 7.3 présente une version simplifiée de la grammaire du langage d'interaction formel de IAT pour le fichier .mtf. Le fichier .mtf constituant une trace d'exécution, il contient quant à lui une instanciation des envois de messages. Cette trace est organisée comme un ensemble de lignes de vie de la forme $[l_i] \ldots ; [l_n]$. Ces lignes de vie vont pouvoir envoyer ou attendre la réception d'un message. L'envoi d'un message est représenté par $l_i!m_k$ (ou $m_k(v_h)$ s'il y a un envoi d'une variable v_h). La réception est représentée par $l_i?m_k$ (ou $m_k(v_h)$ s'il y a une réception d'une variable v_h). Leur concaténation est représentée par un point.

Ces explications sont normalement suffisantes pour exprimer mes exemples dans le formalisme reconnu par IAT, mais plus de détails sur la sémantique formelle sont disponibles dans [109].

J'ai présenté dans cette sous-section différents langages de modélisation, d'une part graphique avec BPMN et, d'autre part, formel avec le langage d'interaction de IAT. Ces langages vont me permettre de décrire des exemples d'application à différents niveaux.

7.2 . Principe de raffinement

Comme dit précédemment, les exemples que j'utilise pour illustrer mes travaux seront considérés à différents niveaux d'abstraction. Cela signifie qu'un raffinement est nécessaire pour passer d'un niveau à l'autre. Dans le cadre de ma thèse, ce raffinement est purement syntaxique et n'a pas d'influence sur les propriétés vérifiées.

7.2.1 . Étapes de raffinement

Je considère les étapes de raffinement suivantes pour un système :

— le système est tout d'abord modélisé à l'aide d'un langage de modélisation graphique (niveau Modèle Graphique) ;

— ensuite, il est raffiné en un modèle formel exprimé à l'aide d'un langage formel (niveau Modèle Formel) ;

— enfin, il est raffiné en un programme à l'aide d'un langage de programmation (niveau Programme)

Par la suite, je considère que le raffinement s'effectue toujours d'un niveau d'abstraction supérieur vers un niveau d'abstraction inférieur.

7.2.2 . Raffinement des éléments du système

Les éléments d'un système (par exemple, des processus) sont toujours raffinés d'un niveau d'abstraction supérieur vers en un **ensemble** d'éléments de même nature dans le niveau d'abstraction inférieur (par exemple, un processus en un ensemble de processus).

Remarques :

— Certains éléments peuvent ne pas être raffinés (vers un niveau inférieur). Cela arrive quand ces éléments n'ont pas de "sens" dans le niveau inférieur. Par exemple, un processus au niveau Modèle Formel n'est pas présent dans le raffinement vers le niveau Programme s'il n'est pas réalisé par l'implémentation du système.

— Certains éléments peuvent être présents dans des niveaux différents. Cela arrive quand l'élément n'est pas raffiné en plusieurs éléments. En d'autres termes, l'élément est raffiné en un singleton contenant uniquement cet élément (ainsi la cohérence du principe de raffinement est conservé).

— Des processus (de niveau supérieur) avec des finalités différentes sont raffinés en des ensembles distincts.

— Les processus (de niveau supérieur) sont raffinés en des ensembles de processus et non pas en des séquences. Le séquencement est présent dans les traces et non dans les éléments du système. Le raffinement du séquencement est présenté ci-après en section 7.2.3.

— Dans mon approche, je fais l'hypothèse que le raffinement spécifié est correct (je ne vérifie donc pas que cela soit réellement le cas).

7.2.3 . Raffinement du séquencement

Pour le raffinement du séquencement des processus, je distingue quatres cas :

— le flux à partir de l'évènement initial, appelé flux initial ;

— le flux entre les processus sans échange de données, appelé flux sans données ;

— le flux entre les processus avec échange de données, appelé flux avec données ;

— le flux de l'évènement terminal, appelé flux terminal.

Ces différents cas sont expliqués ci-après. Pour chaque cas, je redonne un mini-exemple au niveau Modèle Graphique (en BPMN) et j'explique son raffinement au niveau Modèle Formel et au niveau Programme. À noter que ce séquencement permet simplement de garder une cohérence dans le système de mes exemples, mais cela n'influence pas les propriétés que je vérifie.

— **Évènement initial :**

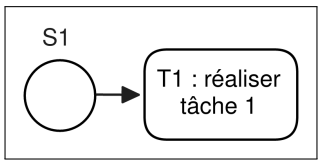


Figure 7.3 – Exemple de flux initial BPMN.

Considérons l'exemple de la figure 7.3 au niveau Modèle Graphique. Le modèle représente l'évènement initial **S1** suivi de la tâche **T1 : réaliser tâche 1**.

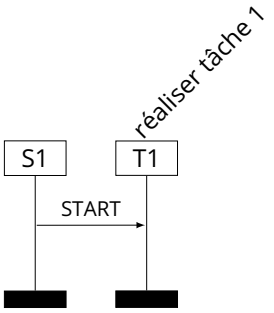


Figure 7.4 – Diagramme de séquence avec évènement initial.

Pour le niveau Modèle Formel, cela sera représenté par le message `START` entre la ligne de vie de **S1** et celle de **T1**. Cela correspond donc au diagramme de séquence de la figure 7.4.

```
@model{
  ...
  S1 -- START -> T1,
  ...
}
```

Figure 7.5 – Modèle IAT avec évènement initial.

Cet échange de message correspond au modèle formel du système pour IAT de la figure 7.5. Ainsi on retrouve bien la transition de déclenchement de **S1** à **T1**.

Ces évènements initiaux ne se retrouvent pas au niveau Programme (pour mes exemples), car ils ne correspondent à aucune action et n'ont donc pas d'équivalents dans le code.

— **Sans échange de données :**

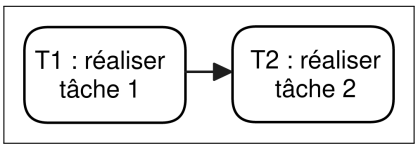


Figure 7.6 – Exemple de flux sans données BPMN.

Considérons l'exemple de la figure 7.6 au niveau Modèle Graphique. Le modèle représente la tâche **T1 : réaliser tâche 1** suivie de la tâche **T2 : réaliser tâche 2**. Comme cela peut être observé sur la figure, il n'y a pas d'association de données sur la flèche de flux de sequence entre ces tâches. Donc il n'y a pas de transfert de données.

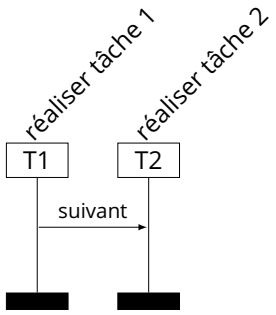


Figure 7.7 – Diagramme de séquence avec échange de données.

Pour le niveau Modèle Formel, cela sera représenté par le message `next` entre la ligne de vie de **T1** et celle de **T2**. Cela correspond donc au diagramme de séquence de la figure 7.7.

```
@model{
  ...
  T1 -- next -> T2, /* réaliser tâche 1 */
  ...
}
```

Figure 7.8 – Modèle IAT avec échange de données entre processus.

Cet échange de message correspond au modèle formel du système pour IAT de la figure 7.8. Ainsi on retrouve bien la transition de **T1** à **T2** sans échange de données.

Pour le niveau Programme, ce séquencement correspond aux appels de l'ensemble des fonctions raffinant **T1** suivis des appels de l'ensemble des fonctions raffinant **T2**, sans partager de données.

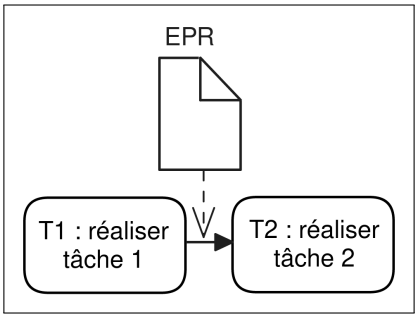


Figure 7.9 – Exemple de flux avec données BPMN.

— **Avec échange de données :**

Considérons l'exemple de la figure 7.9 au niveau Modèle Graphique. Le modèle représente la tâche **T1 : réaliser tâche 1** suivie de la tâche **T2 : réaliser tâche 2** avec le transfert des données **EPR**. Comme cela peut être observé sur la figure, une association de données est présente sur la flèche de flux de sequence entre ces tâches, avec les données **EPR**.

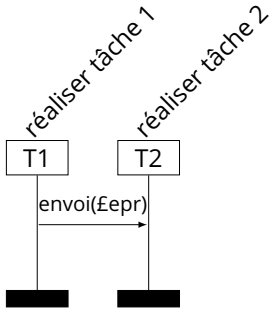


Figure 7.10 – Diagramme de séquence avec échange de données.

Pour le niveau Modèle Formel, cela est représenté par le message `send(£epr)` entre la ligne de vie de **T1** et celle de **T2**. Cela correspond donc au diagramme de séquence de la figure 7.10.

```
@model{
  ...
  T1 -- send(£epr) -> T2, /* réaliser tâche 1 */
  ...
}
```

Figure 7.11 – Modèle IAT avec échange de données entre processus.

Cet échange de message correspond au modèle formel du système pour IAT de la figure 7.11. Ainsi on retrouve bien l'envoi de la donnée **£epr** entre **T1** et **T2**.

Pour le niveau Programme, ce séquencement correspond aux appels de l'ensemble des fonctions raffinant **T1** suivis des appels de l'ensemble des fonctions

raffinant **T2**. Les paramètres modifiés par **T1**, correspondant au raffinement des données **£epr**, sont alors fournis en entrées aux fonctions raffinant **T2**.

— **Évènement terminal :**

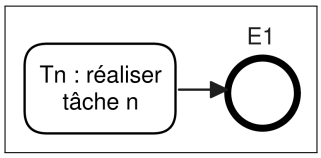


Figure 7.12 – Exemple de flux terminal BPMN.

Considérons l'exemple de la figure 7.12 au niveau Modèle Graphique. Le modèle représente la tâche **Tn : réaliser tâche n** suivi de l'évènement terminal **E1**.

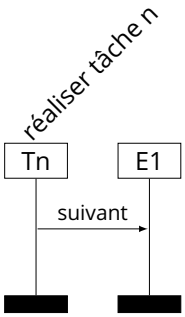


Figure 7.13 – Diagramme de séquence avec évènement terminal.

Pour le niveau Modèle Formel, cela sera représenté par le message `next` entre la ligne de vie de **Tn** et celle de **E1**. Cela correspond donc au diagramme de séquence de la figure 7.13.

```
@model{
  ...
  Tn -- next -> E1, /* réaliser tâche n */
  E1 -- END -> |
  ...
}
```

Figure 7.14 – Modèle IAT avec évènement terminal.

Cet échange de message correspond au modèle formel du système pour IAT de la figure 7.14. Ainsi, on retrouve bien la transition entre **Tn** et **E1**. Un envoi supplémentaire est considéré. En effet, la fin de l'exécution du processus (qui ne concerne donc aucune autre ligne de vie) est représentée par l'envoi du message `END` (nom que j'ai choisi) vers `|` (syntaxe du langage IAT).

Comme pour les évènement initiaux, ces évènements terminaux ne se retrouvent pas au niveau Programme (pour mes exemples), car ils ne correspondent à aucune action et n'ont donc pas d'équivalents dans le code.

7.2.4 . Raffinement d'un système composé de plusieurs processus

Lorsqu'un système est composé de plusieurs processus, cela est pris en compte différemment en fonction du niveau d'abstraction.

— **Au niveau Modèle Graphique**

Au niveau Modèle Graphique, on a un modèle BPMN par processus, chacun composé d'au moins un élément initial, d'un ensemble de tâches et d'au moins un élément terminal.

— **Au niveau Modèle Formel**

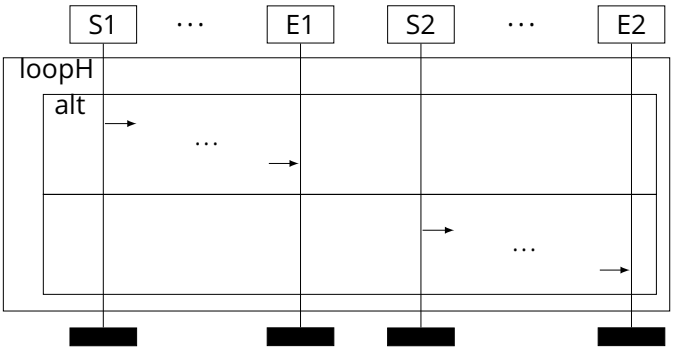


Figure 7.15 – Diagramme de séquence pour un système avec plusieurs processus.

Au niveau Modèle Formel, les modèles correspondant aux processus (en séquences d'évènements) sont ajoutés dans le modèle dans un choix alternatif (`alt`) pour permettre au système d'exécuter un des processus à la fois. De même, pour permettre l'exécution de plusieurs processus à la suite (zéro, une ou plusieurs fois), une boucle est ajoutée dans le modèle (`loopH`).

Considérons un système composé de deux processus. Le premier est composé de l'élément initial $S1$, d'un nombre quelconque de tâches, et de l'élément terminal $E1$. Le second est composé de l'élément initial $S2$, d'un nombre quelconque de tâches, et de l'élément terminal $E2$. Le diagramme de séquence de ce système est illustré sur la figure 7.15. Le modèle correspondant dans le fichier IAT est illustré sur la figure 7.16. Ainsi, lorsque le système est composé de plusieurs processus, chacun d'eux sera dans l'opérateur `alt`.

— **Au niveau Programme**

```
@model{
  loopH(
    alt(
      seq(
        S1 -- START -> ...,
        ...
        E1 -- END -> |
      ),
      seq(
        S2 -- START -> ...
        ...
        E2 -- END -> |
      )
    )
  )
}
```

Figure 7.16 – Modèle IAT de l'exemple GDM_{Spl}.

Au niveau Programme, on s'intéresse à l'implémentation du système et de ses processus. Comme dit précédemment, les processus correspondent à un exemple de fonctions C. Dès lors, l'exécution d'un processus correspond à l'appel des fonctions le composant (dans mes exemples, ces appels ont lieu dans la fonction `main`). Il existe donc une grande variété de possibilités pour les scénarios d'exécution.

J'ai présenté dans cette sous-section le principe de raffinement pour les exemples qui permettent d'illustrer mes travaux, pour passer aussi bien du niveau Modèle Graphique au niveau Modèle Formel, que du niveau Modèle Formel au niveau Programme. Voyons plus en détail ces différents exemples.

7.3 . Cas d'étude : Domaine de la santé

Cette section présente mes exemples liés au domaine de la santé, adaptés d'un exemple de Petkovic et al [132]. Ces exemples ont donc une base commune : ils concernent le système d'information d'un hôpital qui a besoin de traiter des données de patients, appelées **EPR**s pour **E**lectronic **P**atient **R**ecord. Chaque EPR contient certaines données personnelles (par exemple, les dates de naissance), et certaines données non personnelles (par exemple, les posologies dans les prescriptions médicales). Le personnel médical peut utiliser le système d'information de l'hôpital pour traiter les EPRs dans deux buts différents : fournir un traitement médical aux patients (représenté par la suite par le mot clé **Traitement**, ou *Treatment*) ou réaliser un essai clinique (représenté par la suite par le mot clé **Recherche**, ou

Research). Dans les deux cas, les médecins doivent demander un accès aux données personnelles des patients lorsqu'ils en ont besoin.

Le premier exemple est une version simplifiée, tandis que le deuxième reprend celui présenté par Petkovic et al [132].

7.3.1 . Gestion Données Médicales - version Simple

Pour des raisons de simplification, contrairement à l'exemple de Petkovic et al [132], et dans un premier temps, je ne considère ni les corridors (*lane*), ni les branchements, ni les évènements d'erreurs. En outre, je ne reprends qu'une sous-partie de leur processus pour le traitement médical (`S1`, `T10: check medical equipment`, `T01: collect symptoms`, `T02: make diagnosis`, `T03: prescribe medical treatment`, `T04: discharge patient` et `E1`), et j'ai simplifié leur processus pour l'essai clinique en regroupant différentes tâches pour n'en garder que trois sur les cinq initiales. J'ai également explicité, sur mes modèles BPMN, l'utilisation des données personnelles (contenues dans les EPRs).

Les sous-sections suivantes présentent le système aux trois niveaux d'abstraction, présentés en section 7.1. La première présente un modèle BPMN de ce système, la deuxième, le raffinement vers un modèle IAT, et la dernière, l'implémentation en C de ce système. Comme nous avons utilisé une version de cet exemple dans notre article [41], j'ai gardé la terminologie utilisée, et donc l'exemple est en anglais.

Modèle Graphique - BPMN

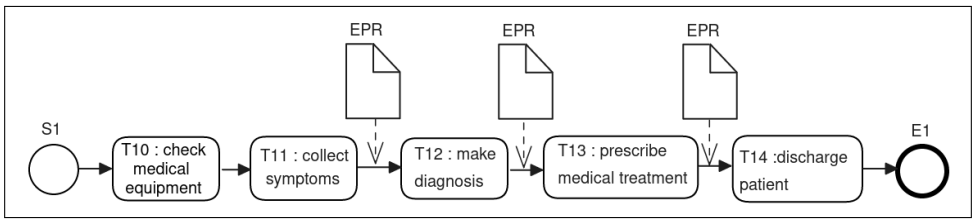


Figure 7.17 – Processus P1 pour le traitement médical.

Le modèle BPMN est composé de deux processus P1 et P2, représentant respectivement un traitement médical et un essai clinique. La figure 7.17 définit le processus P1, tandis que la figure 7.18 définit le processus P2.

Le processus **P1** est composé de l'évènement initial **S1**, des tâches **T10 : check medical equipment**, **T11 : collect symptoms**, **T12 : make diagnosis**, **T13 : prescribe medical treatment** et **T14 : discharge patient** en séquence, de données **EPRs** et de l'élément terminal **E1**. D'abord la tâche **T10 : check medical equipment** correspond aux actions réalisées pour vérifier le matériel médical avant d'ausculter le patient (par exemple, changer le papier de la table d'auscultation).

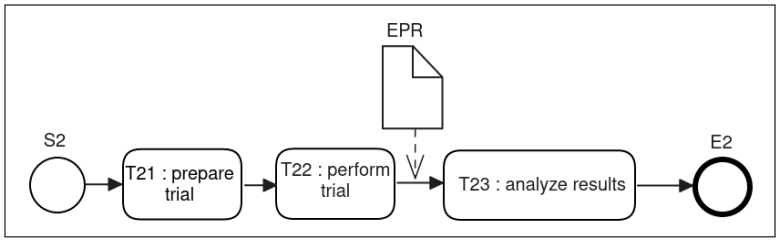


Figure 7.18 – Processus P2 pour l'essais clinique.

Puis la tâche **T11 : collect symptoms** correspond à la collecte des symptômes du patient, qui font partie des données du patient (**EPR**). Ensuite, à l'aide de ces données, le processus effectue la tâche **T12 : make diagnosis** qui correspond au diagnostic, complétant les données (**EPR**). Puis, à l'aide des données (contenues dans **EPR**), la tâche **T13 : prescribe medical treatment** permet de prescrire un traitement médical. Enfin la tâche **T14 : discharge patient** correspond à la fin de la prise en charge du patient.

Le processus **P2** est composé de l'élément initial **S2**, des tâches **T21 : prepare trial**, **T22 : perform trial**, **T23 : analyze results**, des données **EPR** et de l'élément terminal **E2**. La tâche **T21 : prepare trial** correspond à la préparation de l'étude, nécessaire à la tâche **T22 : perform trial** correspondant quant à elle à la réalisation de l'étude, ce qui utilise et génère des données liées aux patients (**EPR**). Enfin, ces données sont analysées pendant la tâche **T23 : analyze results**.

Cet exemple diffère légèrement de l'exemple « *Healthcare* » publié dans [41]. En effet, la tâche *T10 :check medical equipment* a été ajoutée afin de représenter une tâche qui n'utiliserait pas de données personnelles (ce qui permet de tester l'approche sur des tâches traitant des données personnelles et sur des tâches qui n'en traitent pas), tout en étant cohérente dans l'exemple. Dans l'exemple « *Healthcare* » de [41], cette tâche était *T14 : discharge patient*, mais après analyse, il était plus cohérent de considérer que cette tâche traite des données personnelles. Ainsi, une nouvelle tâche a été ajoutée.

Modèle Formel - IAT

Le système défini précédemment est raffiné en un modèle formel, utilisable par l'outil IAT [109]. Le diagramme de séquence correspondant est présenté en figure 7.19 et le modèle formel (fichier `.imf`) est présenté en figure 7.20. Le principe de ce raffinement est détaillé en section 7.2. Les processus $P1$ et $P2$ y sont définis en tant que séquences (`seq`) d'évènements. Ces évènements, sous forme d'échanges de messages, correspondent aux tâches et à leurs relations définies dans le modèle BPMN. Ces messages correspondent au flux de séquence entre

les tâches dans le modèle BPMN. Une flèche sans envoi de données en BPMN, correspond au message `next`. (par exemple, entre les tâches **T10 : check medical equipment** et **T11 : collect symptoms**), Une flèche avec transfert de EPR en BPMN, correspond au message `send(£ epr)` (par exemple, entre les tâches **T11 : collect symptoms** et **T12 : make diagnosis**).

Suivant le principe de raffinement détaillé en section 7.2, les modèles correspondant à P1 et P2 (en séquences d'évènements) sont ajoutés dans le modèle *via* un choix alternatif (`alt`), pour permettre au système d'exécuter un des deux processus. Ce choix alternatif étant lui-même dans une boucle (`loopH`). Ainsi cet exemple, à ce niveau d'abstraction, ne laisse pas la possibilité d'entrelacements d'évènements appartenant aux deux processus.

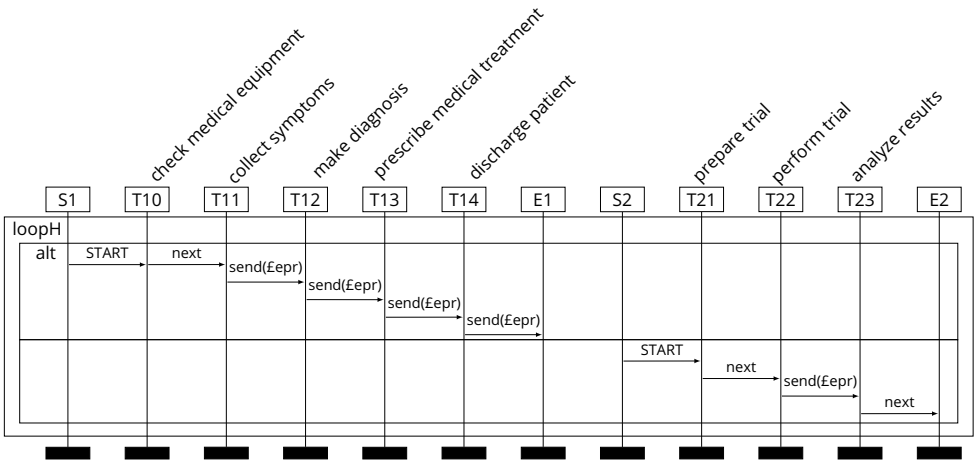


Figure 7.19 – Diagramme de séquence de l'exemple GDM_{Spl}.

En suivant le principe de raffinement décrit en section 7.2, les processus P1 et P2 sont raffinés en sequence d'interaction contenant l'ordonnancement des lignes de vie (i.e. processus) et des échanges de messages. Ainsi le processus P1 est représenté par les lignes de vie `S1`, `T10`, `T11`, `T12`, `T13`, `T14` et `E1`, tandis que le processus P2 est représenté par les lignes de vie `S2`, `T21`, `T22`, `T23` et `E2`. Les messages considérés sont `START`, `next`, `send(£epr)`, `END`. Les figures 7.19 et 7.20 présentent leur ordonnancement. Pour pouvoir utiliser le modèle précédent, différentes spécifications sont à indiquer dans le fichier. D'abord, les tâches possibles sont spécifiées dans le bloc des lignes de vie :

```
@lifeline{
   S1; T10; T11; T12; T13; T14; E1; S2; T21; T22; T23; E2
}
```

Ensuite, les messages possibles sont spécifiés dans le bloc des messages :

```
@message{ START; send(Integer); next; END }
```

Enfin, les données utilisées par les processus sont spécifiées dans le bloc des variables :

```
@model{
  loopH(
    alt(
      seq(
        S1 -- START -> T11,
        T10 -- next -> T11, /* check medical equipment */
        T11 -- send(£epr) -> T12, /* collectSymptoms */
        T12 -- send(£epr) -> T13, /* makeDiagnosis */
        T13 -- send(£epr) -> T14, /* prescribeTreatment */
        T14 -- next -> E1, /* dischargePatient */
        E1 -- END -> |
      ),
      seq(
        S2 -- START -> T21,
        T21 -- next -> T22, /* prepareTrial */
        T22 -- send(£epr) -> T23, /* performTrial */
        T23 -- send(£epr) -> E2, /* analyzeResults */
        E2 -- END -> |
      )
    )
  )
}
```

Figure 7.20 – Modèle IAT de l'exemple GDM_{Spl}.

```
@variable{ £epr: Integer; }
```

Actuellement, IAT ne permet pas d'utiliser des structures de données. J'ai donc choisi de représenter **EPR** comme une variable de type entier.

Ces données sont initialisées dans le bloc d'initialisation :

```
@init{ £epr = 4; }
```

La valeur d'initialisation n'est pas significative : une autre valeur que 4 aurait également pu être choisie.

La correspondance du raffinement est présentée dans la table 7.4. L'annexe A.1.2 présente le fichier complet de spécification pour IAT, correspondant au système de l'exemple GDM_{Spl}.

Pour ce cas d'utilisation, ce niveau d'abstraction (Modèle Formel) n'était pas abordé dans l'article [41].

Table 7.4 – Correspondances du raffinement entre modèle graphique et le modèle formel.

ML (BPMN)	**ML formel**
P1	S1 T10 T11 T12 T13 T14 E1
P2	S2 T21 T22 T23 E2
EPR	£epr

Implémentation en C

Les processus modélisés au niveau Modèle Formel sont raffinés au niveau Programme en tant que fonctions C. Seuls les processus correspondant à des parties intégrées dans un logiciel sont raffinés. En effet, les processus initiaux peuvent contenir des parties réalisées indépendamment du programme informatique (par exemple, préparer l'étude clinique ou bien la fin de prise en charge du patient). Ces raffinements sont résumés dans la table 7.5.

Table 7.5 – Correspondances du raffinement entre le modèle formel et le programme.

ML formel	**PL**
T13	makePrescr
T22	getData
T23	computeStats
£epr	Patient, TrialData, Dos

Dans cet exemple, la tâche T13, définie au niveau Modèle Formel, est raffinée en la fonction makePrescr au niveau Programme, qui code la réalisation d'une prescription médicale. De même, la tâche T22, définie au niveau Modèle Formel, est raffinée en la fonction getData, correspondant à l'accès aux données pour l'essai clinique, tandis que la tâche T23, est raffinée en la fonction computeStats, correspondant à la réalisation de calculs statistiques pour l'étude.

Les données personnelles identifiées au niveau Modèle Formel sont également raffinées vers le niveau Programme. Au niveau Modèle Formel, les données étaient modélisées par la variable **£epr** (Electronic Patient Record). Cette modélisation est raffinée au niveau Programme en trois types de structures de données C : `Patient`, `TrialData` et `Dos`. `Patient` contient les données plus spécifiques au patient, `TrialData` contient celles plus spécifiques à l'essai clinique (donc contenant entre autre des données issues des données de `Patient`), et `Dos` correspond à la posologie (*dosage*) dans une prescription médicale (par exemple, « Prendre un Doliprane 500mg toutes les 4 heures »).

La figure 7.21 présente une simplification du code C implémentant le système décrit précédemment. Les différents appels des fonctions (correspondants aux processus) sont effectués dans la fonction `main`. Ici, le corps des fonctions ne correspond pas à une réelle implémentation des fonctionnalités du système.

```c
/*--  DATA --*/
typedef struct {
    char date[S_SIZE];
    char medecine[S_SIZE];
} Dos;

typedef struct {
    int id;
    char name[S_SIZE];
    char lastname[S_SIZE];
    char birthdate[S_SIZE];
    char address[S_SIZE];
    char sex[S_SIZE];
    Dos dosList[NB_DOS];
} Patient;

typedef struct {
    char birthdateList[NB_PATIENT][S_SIZE];
    char sexList[NB_PATIENT][S_SIZE];
    Dos dosList[NB_DOS];
} TrialData;

/*--  FUNCTIONS --*/
Patient makePrescr(Dos newd, Patient p){
    ...
}

TrialData getData(Patient patientList[NB_PATIENT]){
    ...
}

int computeStats(TrialData data){
    ...
}

/*--  MAIN --*/
int main(){
    ...
    return 0;
}
```

Figure 7.21 – Exemple GDM$_{Spl}$ au niveau programme.

7.3.2 . Gestion Données Médicales - version Complexe

Comme dit précédemment, cet exemple complexe correspond à l'exemple défini par Petkovic et al. [132]. J'ai repris leur exemple pour permettre de tester mon approche sur des modèles plus complexes. Il est à noter que, dans leur approche, ils n'ont pas défini d'implémentation de leur modèle, donc ce cas d'utilisation ne

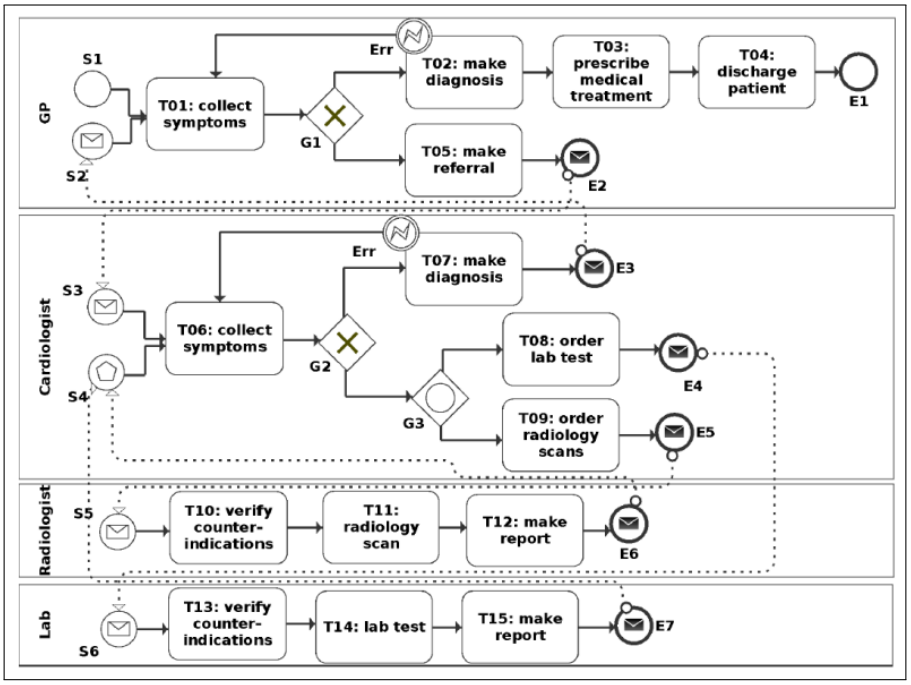


Figure 7.22 – Healthcare Treatment Process [132].

sera utilisé dans mon approche qu'au niveau Modèle Graphique. De plus, je ne me base que sur le modèle BPMN qu'ils ont défini et les informations y référant, car je considère également ce langage pour le niveau Modèle Graphique. Par contre, je n'ai pas utilisé le modèle formel qu'ils proposent car il n'était pas complet dans sa spécification. De plus, leur modèle ne repose pas sur l'échange de messages (tel qu'il fonctionne dans IAT). Une suite de mes travaux pourrait consister en traduire leur modèle dans le langage utilisé par IAT, en extrapolant les informations manquantes, mais cela ne semblait pas nécessaire pour une première évaluation de mes contributions.


Cet exemple était déjà mentionné dans [41] sous le nom « *Purpose Control* », mais je donne ici plus de détails.

Modèle Graphique - BPMN

Les figures 7.22 et 7.23 correspondent aux modèles BPMN définis par Petkovic et al. [132] représentant les deux processus du système informatique de l'hôpital. La première correspond au processus pour le traitement médical (*Healthcare Treatment Process*) et la seconde au processus pour l'essai clinique (*Clinical Trial Process*). En plus des éléments que l'on retrouve dans ma version simplifiée, cet exemple contient des tâches supplémentaires pour les différents processus, des

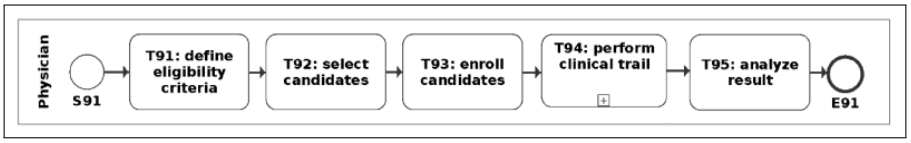


Figure 7.23 – Clinical Trial Process [132].

corridors (*lane*) selon le personnel de l'hôpital considéré, des branchements conditionnels, des envois de messages et des évènements d'erreurs.

Si l'on ne considère que les processus en eux-mêmes, cela revient au fonctionnement de ma version simplifiée, c'est-à-dire que l'on considère deux processus *Healthcare Treatment Process* et *Clinical Trial Process*, tous deux utilisant à certains moments des données personnelles (EPR). Le premier processus traitant les données avec la finalité de prescrire un traitement médical au patient (Traitement), le second avec la finalité de réaliser un essai clinique (Recherche). Ceci revient finalement à ma version simplifiée, avec *Healthcare Treatment Process* au lieu de $P1$ et *Clinical Trial Process* au lieu de $P2$.

Si, au contraire, on s'intéresse au contenu des processus, cela devient plus complexe. Le processus *Clinical Trial Process* est composé d'un corridor (*Physician*) composé de plusieurs éléments : l'évènement initial **S91**, les tâches **T91**, **T92**, **T93**, **T94** et **T95** en séquence et l'évènement terminal **E91**. Bien que le processus utilise les données personnelles EPR, cela n'est pas explicite dans le modèle, donc par défaut, je considère que chacun de ces éléments peut les utiliser. Toutes ces tâches sont en séquence, et donc s'exécutent les unes après les autres

Le processus *Healthcare Treatment Process*, quant à lui, est composé de plusieurs corridors (*GP*, *Cardiologist*, *Radiologist*, and *Lab*), ces corridors communiquant entre eux à l'aide d'envois et de réceptions de messages. Ces corridors sont composés de plusieurs éléments.

— Le corridor *GP* contient les évènements initiaux **S1** et **S2**, les tâches **T01 : collect symptoms**, **T02 : make diagnosis**, **T03 : prescribe medical treatment**, **T04 : discharge patient**, et **T05 : make referral**, les évènements terminaux **E1** et **E2**, et l'évènement d'erreur **Err**, ainsi que le branchement **G1** permettant de relier la tâche **T01** avec **T02** et **T05**.

— Le corridor *Cardiologist* contient les évènements initiaux **S3** et **S4**, les tâches **T06 : collect symptoms**, **T07 : make diagnosis**, **T08 : order lab test** et **T09 : order radiology scans**, les évènements terminaux **E3**, **E4** et **E5**, et l'évènement d'erreur **Err**, ainsi que le branchement **G2** permettant de relier la tâche **T06** avec **T07** et le branchement **G3**, et le branchement **G3** permettant de relier le branchement **G2** avec **T08** et **T09**.

— Le corridor *Radiologist* contient l'évènement initial **S5**, les tâches **T10 : verify counter-indications**, **T11 : radiology scan** et **T12 : make report**,

et l'évènement terminal **E6**.

— Le corridor *Lab* contient l'évènement initial **S6**, les tâches **T13 : verify counter-indications**, **T14 : lab test** et **T15 : make report**, et l'évènement terminal **E7**.

Dans chacun des corridors, les éléments sont reliés entre eux par un flux de séquence. Comme pour le processus *Clinical Trial Process*, bien que le processus utilise les données personnelles EPR, cela n'est pas explicité sur le modèle, donc par défaut je considère que chacun de ces éléments peuvent les utiliser.

Pour des soucis de visibilité, et comme aucune de ces tâches ne sera implémentée dans la suite, leurs significations ne sont pas détaillées ici, mais elles sont disponibles dans l'article de Petkovic et al. [132].

7.4 . Cas d'étude : Domaine des sites internet

J'ai défini un autre exemple, avec plusieurs variantes, afin d'illustrer mon langage et la vérification de propriétés dans un autre contexte ; mais également afin de considérer un exemple légèrement plus complexe, avec des processus partageant des tâches et des processus différents ayant néanmoins une même finalité.

Pour définir cet exemple, je me suis inspirée de l'exemple présenté dans l'article de Hayati et al. [70]. J'ai repris l'idée d'illustrer mon approche avec un système correspondant à un site web et la dualité entre deux finalités. Les deux finalités choisies sont la gestion de l'historique d'achats (**Administratif**) et la gestion d'envoi de publicités ciblées selon l'analyse des achats précédents (**Marketing**). Le système est conçu pour être utilisé par un client réalisant des achats. Les exemples suivants sont beaucoup plus simples que des sites web réels, mais permettent d'évaluer mon approche sur différents cas.

7.4.1 . Achat en Ligne - version Simple

Bien que cet exemple soit globalement plus complexe que celui présenté pour le domaine de la santé, la version simplifiée ne contient pas de boucle ni de branchement.


Cet exemple était mentionné dans [41] sous le nom « *Website* », mais je donne ici plus de détails.

Modèle Graphique - BPMN

Le système est défini par trois processus **P1**, **P2** et **P3** respectivement représentés par les figures 7.24, 7.25 et 7.26.

Les processus **P1** et **P2** ont pour finalité la gestion de l'historique d'achats (**Administratif**), et le processus **P3** a pour finalité la gestion d'envoi de publicités ciblées, selon l'analyse des achats précédents (**Marketing**).

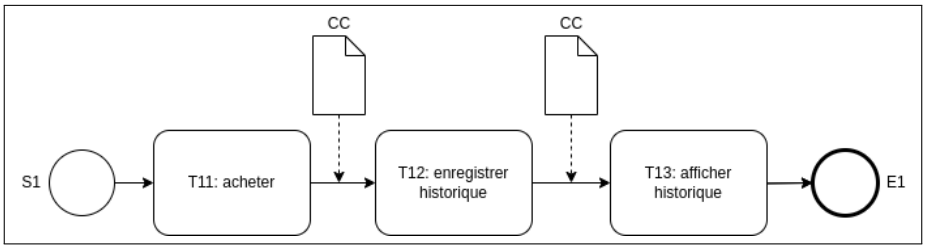


Figure 7.24 – Processus P1 de AL_{Spl}.

Le processus **P1** est composé de l'élément initial **S1**, des tâches **Acheter**, **Enregistrer historique**, **Afficher historique**, des données **CC** (signifiant Compte Client), et de l'élément terminal **E1**. Ce processus correspond aux étapes qui sont

réalisées lorsqu'un utilisateur réalise des achats qui sont enregistrées dans son historique : l'utilisateur effectue d'abord ses achats qui sont ajoutés à son compte client. Ensuite, cet historique est enregistré dans une base de données à partir des informations du compte client. Enfin, l'historique de son compte est affiché.

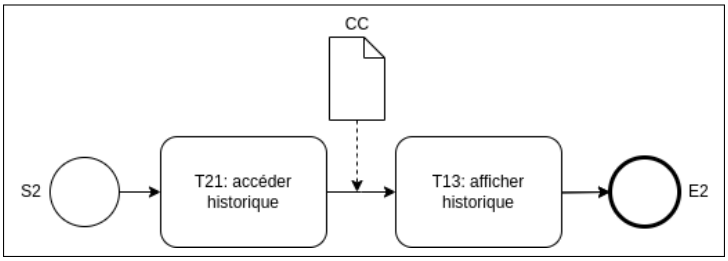


Figure 7.25 – Processus P2 de AL_{Spl}.

Le processus **P2** est composé de l'élément initial **S2**, des tâches **Accéder historique**, **Afficher historique**, des données **CC**, et de l'élément terminal **E2**. Ce processus correspond aux étapes qui sont réalisées lorsqu'un utilisateur consulte son historique : l'utilisateur demande ainsi à accéder à son historique, avant que les informations liées à son compte ne soient ensuite utilisées pour afficher l'historique.

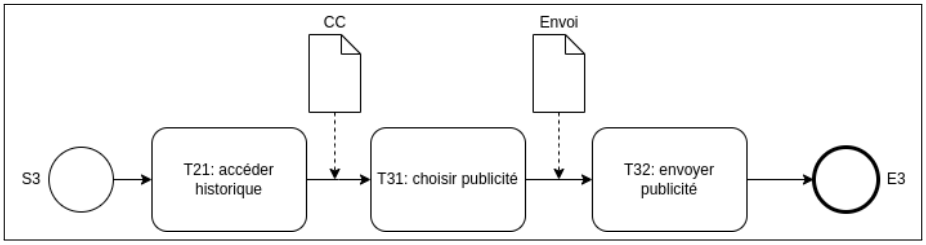


Figure 7.26 – Processus P3 de Achat en Ligne - version Simple.

Le processus **P3** est composé de l'élément initial **S3**, des tâches **Accéder historique**, **Choisir publicité**, **Envoyer publicité**, des données **CC**, des données **Envoi** (contenant la publicité à envoyer, ainsi que les coordonnées cibles), et de l'élément terminal **E3**. Ce processus correspond aux étapes qui sont réalisées lors de l'envoi de publicités ciblées, selon l'historique d'achat : le système accède d'abord à l'historique d'un client, puis à partir des informations liées au compte, une publicité est choisie, avant d'être envoyée à chaque coordonnée associée à un compte client.

Modèle Formel - IAT

Le système défini précédemment est raffiné en un modèle formel, dans le format utilisable par l'outil IAT [109]. Ce modèle suit le même principe que celui pour l'exemple GDM_{Spl} détaillé section 7.3.1.

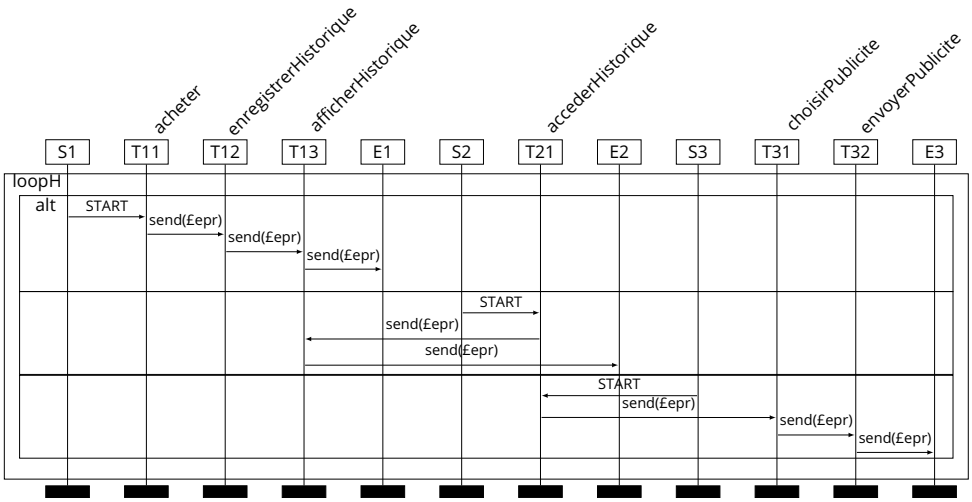


Figure 7.27 – Diagramme de séquence représentant le système du modèle AL_{Spl}.

Ainsi, les processus sont définis en tant que séquences d'évènements (`seq`). Ces évènements correspondent aux tâches et à leurs relations définies dans le modèle BPMN. Ils sont ajoutés dans le système *via* un choix alternatif (`alt`) contenu dans une boucle (`loopH`). Le diagramme de sequence correspondant au système est présenté sur la figure 7.27. Le modèle formel spécifié pour IAT est présenté sur la figure 7.28.

On retrouve sur ces figures l'ordonnancement des lignes de vie et des échanges de messages contenus dans le processus P1 précédent, correspondant ici à la première séquence d'évènements. D'une manière similaire, la deuxième séquence d'évènements correspond au processus P2, et la troisième au processus P3.

Le reste du fichier de spécification de IAT (e.g. bloc `@lifeline`, `@message`, `@variable`,`@init`) suit le même principe que celui de l'exemple Healthcare introduit dans la section 7.3.1. En effet, le processus P1 est représenté par les lignes de vie `S1`, `T11`, `T12`, `T13` et `E1`, le processus P2 est représenté par les lignes de vie `S2`, `T21`, `T13` et `E2`, et le processus P3 est représenté par `S3`, `T21`, `T31`, `T32` et `E2`. La correspondance du raffinement est présentée dans la table 7.6. On remarque que certaines lignes de vie sont partagées lors du raffinement. Par exemple, les processus P1 et P2 au niveau modèle contiennent tous les deux la tâche `T13`.

L'annexe A.3.3 présente le fichier complet de spécification pour IAT, correspondant au système de l'exemple Achat en Ligne - version Simple.

Implémentation en C

La définition de l'exemple au niveau Modèle Formel (ML Formel) a été vue en détail. Une fois cette modélisation effectuée, le système est raffiné vers le niveau Programme (PL). Les processus y sont raffinés en tant que fonctions C, correspon-

```
@model{
  loopH(
    alt(
      seq(
        S1 -- START -> T11,
        T11 -- send(£CC) -> T12, /* acheter */
        T12 -- send(£CC) -> T13, /* enregistrerHistorique */
        T13 -- send(£CC) -> E1, /* afficherHistorique */
        E1 -- END -> |
      ),
      seq(
        S2 -- START -> T21,
        T21 -- send(£CC) -> T13, /* accederHistorique */
        T13 -- send(£CC) -> E2, /* afficherHistorique */
        E2 -- END -> |
      ),
      seq(
        S3 -- START -> T21,
        T21 -- send(£CC) -> T31, /* accederHistorique */
        T31 -- send(£CC) -> T32, /* choisirPublicite */
        T32 -- send(£Envoi) -> E3, /* envoyerPublicite */
        E3 -- END -> |
      )
    )
  )
}
```

Figure 7.28 – Modèle formel de l'exemple AL_{Spl}.

dant à des parties des processus qui seront intégrées au logiciel. Une même tâche peut correspondre à plusieurs fonctions.

Ainsi, dans l'exemple :

— la tâche `T11 : Acheter` est raffinée en la fonction `buy` correspondant à la réalisation d'achats ;

— la tâche `T12 : Enregistrer historique` est raffinée en la fonction `store` correspondant à l'enregistrement des données du compte client ;

— la tâche `T13 : Afficher historique` est raffinée en les fonctions `display` correspondant à l'affichage d'un compte client, et `displayHistory` correspondant à l'affichage d'un historique d'achat ;

— la tâche `T21 : Accéder historique` est raffinée en les fonctions `getAccount` correspondant à l'accès à un compte client, et `getHistory` correspondant à l'accès à l'historique d'achat d'un client ;

— la tâche `T31 : Choisir publicité` est raffinée en la fonction `chooseAd` correspondant à la selection d'une publicité selon un historique d'achat ;

Table 7.6 – Correspondances du raffinement entre le modèle graphique et le modèle formel

ML (BPMN)	**ML formel**
P1	S1 T11 T12 T13 E1
P2	S2 T21 T13 E2
P3	S3 T21 T31 T32 E3
CC	£CC
Envoi	£Envoi

— la tâche T32 : Envoyer publicité est raffinée en la fonction sendAd correspondant à l'envoi d'une publicité à un client.

La table 7.7 résume ce raffinement.

Table 7.7 – Correspondances du raffinement du modèle formel vers le programme.

ML formel	**PL**
T11	buy
T12	store
T13	display, displayHistory
T21	getAccount, getHistory
T31	chooseAd
T32	sendAd
£CC	ClientAccount, Client, PurchaseList, Purchase
£Envoi	Client, Ad

Les données identifiées au niveau Modèle Formel sont également raffinées en différents types de données au niveau Programme. Au niveau Modèle Formel, les

données étaient modélisées par les variables **£CC** (Compte Client) et **£Envoi** (informations pour l'envoi de publicité). Elles sont raffinées au niveau Programme en des ensembles de structures de données C :

— £CC en les structures `ClientAccount`, `Client`, `PurchaseList` et `Purchase` ;

— £Envoi en les structures `Client`, `Ad`.

On remarque que ces raffinements ne sont pas disjoints. En effet, la structure `Client` est commune aux deux raffinements. La figure 7.29 présente un extrait du code du système Website.

```c
typedef struct {
    int id;
    char content[S_SIZE];
} Ad;

typedef struct {
    char name[S_SIZE];
    int price;
} Purchase;

typedef struct {
    char date[S_SIZE];
    Purchase purchaseList[NB_Purchase];
    int total;
} PurchaseList;

typedef struct {
    int id;
    char name[S_SIZE];
    char lastname[S_SIZE];
    char birthdate[S_SIZE];
    char email[S_SIZE];
} Client;

typedef struct {
    Client client;
    PurchaseList history[NB_Purchase];
} ClientAccount;


/*--  FUNCTIONS --*/
ClientAccount buy(Client c, PurchaseList a){  ...  }
void store(ClientAccount cc){  ...  }

void displayHistory(PurchaseList* a){  ... }

void display(ClientAccount cc){  ...  }

ClientAccount getCompte(Client c){  ...  }

PurchaseList* getHistory(Client c){  ...  }

Ad chooseAd(PurchaseList* purchases){  ...  }

void sendAd(Ad p, Client c){  ...  }

/*--  MAIN --*/
int main(){
    Client client = {.name="John", .lastname="Doe"};
    PurchaseList purchases = {.date = ..., ...};
    /*-- P1  --*/
    ClientAccount cc = buy(client, purchases);
    store(cc);
    display(cc);
    return 0;
}
```

Figure 7.29 – Extrait du programme de l'exemple du système AL_{Spl}.

7.4.2 . Achat en Ligne - version Complexe

Pour permettre de tester mon approche sur des codes C plus complexes, j'ai étendu cet exemple. Ce cas d'utilisation ne sera donc utilisé qu'au niveau Programme.


Cette extension n'existait pas au moment de la publication de l'article [41].

Implémentation en C

Cet exemple est une extension de la version simplifiée. Il contiendra donc le programme précédent ainsi que des extensions. Tout particulièrement, une nouvelle fonction, `actionUser`, est ajoutée. Cette fonction permet de simuler les actions réalisées par l'utilisateur, ce qui va influencer le code exécuté.

```
int actionUser(Client client, PurchaseList purchases){
    int actions[...] = ...;
    for(int i = 0; i < ...; i++){
        if(actions[i] == 0){
            /*-- P1  --*/
            ClientAccount cc = buy(client, purchases);
            store(cc);
            display(client);
        }
        else if(actions[i] == 1){
            /*-- P2  --*/
            PurchaseList history = getHistory(client);
            displayHistory(history);
        }
        else if(actions[i] == 2){
            /*-- P3  --*/
            PurchaseList history = getHistory(client);
            Ad p =  chooseAd(history);
            sendAd(p, client);
        }
    }
    return 0;
}
```

Figure 7.30 – Extrait du code de la fonction `ActionUser` de l'exemple AL_{Cpx}.

La figure 7.30 présente le code de cette fonction. On simule les différentes actions de l'utilisateur *via* un tableau d'entiers (`actions`). Selon les valeurs contenues dans ce tableau, différents branchements sont possibles. Ces branchements représentent les exécutions des processus définis au niveau modèle ($P1$, $P2$ et $P3$) en regroupant des appels aux fonctions définies pour la version simple de l'exemple

Achat en Ligne. Cela permet d'évaluer mon approche sur un code contenant des boucles et branchements influençant les processus exécutés, et donc les finalités du traitement réalisé. Par exemple, selon le code défini sur la figure 7.30, si l'action de l'utilisateur est évaluée à 0, alors les fonctions raffinant le processus $P1$ sont appelées. La finalité du traitement est donc celle associée à $P1$, c'est à dire la gestion de l'historique d'achats (**Administratif**). De même, si l'action de l'utilisateur est évaluée à 1, les fonctions raffinant le processus $P2$ sont appelées. La finalité du traitement est donc celle associée à $P2$, c'est à dire également la gestion de l'historique d'achats (**Administratif**). Par contre, si l'action de l'utilisateur est évaluée à 2, les fonctions raffinant le processus $P3$ sont appelées. La finalité du traitement est donc celle associée à $P3$, c'est à dire la gestion d'envoi de publicités ciblées, selon l'analyse des achats précédents (**Marketing**). En d'autres termes, la finalité du traitement réalisé dépend des actions réalisées par l'utilisateur.

Cette section a présenté plusieurs exemples de systèmes venant de différents domaines d'applications, le domaine de la santé et le domaine des sites internet, à différentes étapes de développement (aux niveaux Modèle Graphique, Modèle Formel et Programme). J'ai détaillé ces systèmes au niveau Modèle Graphique à l'aide du langage de modélisation BPMN, au niveau Modèle Formel à l'aide du langage d'interaction utilisé par IAT, et au niveau Programme à l'aide du langage C. Pour chacun de ces systèmes, j'ai défini deux variantes : une version simple et une version complexe. Au total, j'ai donc quatre exemples, à savoir Gestion Données Médicales - version Simple, Gestion Données Médicales - version Complexe, Achat en Ligne - version Simple et Achat en Ligne - version Complexe. Ces exemples vont permettre d'illustrer et d'évaluer mes contributions.

Le chapitre suivant présente une de ces contributions : le langage de spécification CSpeL. Ce langage permet de spécifier formellement les éléments de ces exemples qui sont nécessaires pour la vérification de propriétés liées au consentement, et ce, quel que soit le niveau d'abstraction des exemples.

8 - Langage de spécification de contexte - CSpeL

Pour vérifier qu'un système respecte des propriétés liées à la vie privée, notamment des propriétés liées au traitement des données, il est nécessaire de formaliser un modèle du système et le contexte lié à la vie privée dans lequel il s'inscrit (c'est à dire son environnement, mais également les notions liées à la vie privée à prendre en compte), ainsi que les propriétés à vérifier. Pour cela, j'ai défini CSpeL, *Context Specification Language*, un langage permettant de spécifier formellement :

— un système dans un contexte de vie privée ; et

— une trace d'exécution.

Nous verrons dans le chapitre suivant comment ce langage permet de vérifier des propriétés liées au traitement des données et à la vie privée. De plus, CSpeL présente l'avantage de pouvoir représenter des systèmes variés et cela à différentes étapes de leur cycle de vie. En effet, ce langage peut être utilisé pour formaliser un système indépendamment de son niveau d'abstraction (niveau modèle ou programme).

D'abord, la section 8.1 détaille la sémantique de CSpeL. Ensuite, la section 8.2 introduit la grammaire de CSpeL qui est utilisée en pratique. Enfin, la section 8.3 détaille une extension ajoutée au langage.

8.1 . Syntaxe et sémantique formelle

Cette section présente la sémantique de CSpeL, en particulier comment le système est modélisé, comment l'exécution du système est représentée et comment des propriétés de consentement vérifiées dans notre approche sont formalisées.

8.1.1 . Modèle formel du système dans son contexte lié à la vie privée

CSpeL permet de spécifier formellement le système dans son contexte lié à la vie privée. Comme illustré sur la figure 8.1, CSpeL reprend une sous-partie des éléments définis par PyCO, détaillé section 5. Pour rappel, PyCO est une ontologie du contexte lié à la vie privée, se limitant à des notions liées à la transparence, la responsabilité et au traitement des données, telles qu'elles sont définies dans GePyR, ma taxonomie présentée en section 4. Pour les éléments de type *Entities* de PyCO, je ne conserve que ceux qui sont automatisables, c'est à dire les entités non humaines (par exemple, le système composé de processus est conservé, mais pas la personne concernée (*Data Subject*)). Dans les éléments de type *Informations*, je n'ai pas conservé dans CSpeL, pour des raisons de simplification, les durées de conservation (*Storage periods*), car elles ne sont pas nécessaires pour les propriétés vérifiées dans le cadre de cette thèse. Je fais également une autre

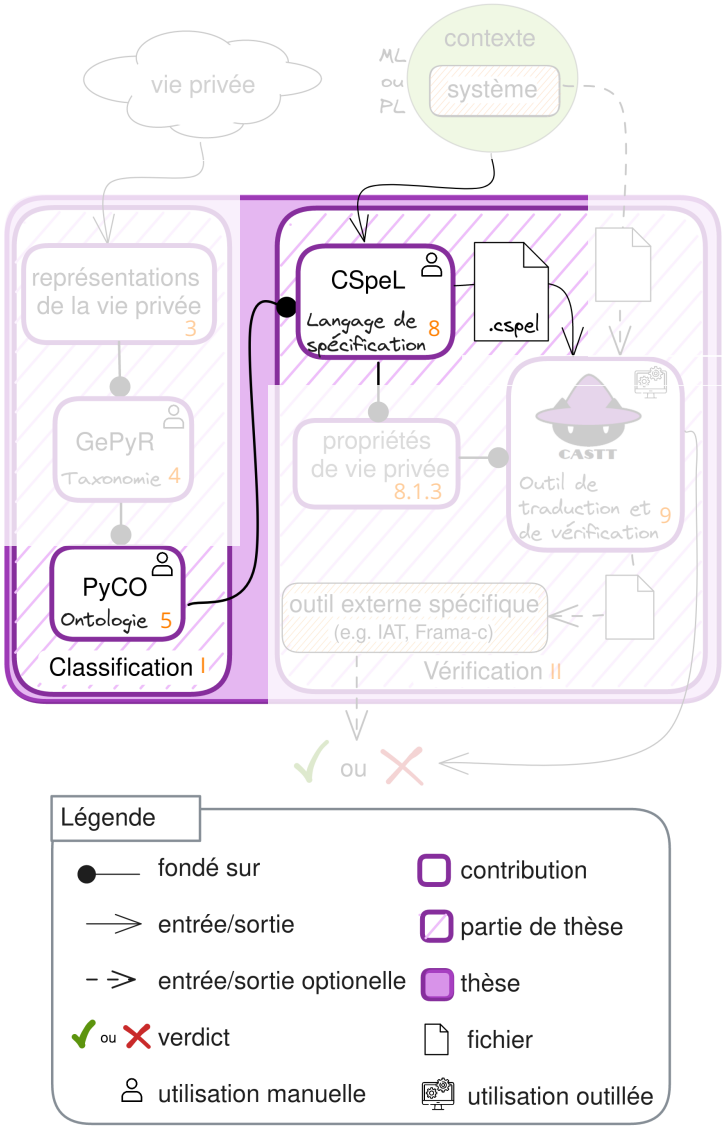


Figure 8.1 – Rappel - Lien entre PyCO et CSpeL.

simplification : bien que la Notice (*Notice*) ne soit pas définie en tant qu'élément à part entière, CSpeL fait référence aux éléments la composant, à savoir les finalités et les données personnelles traitées. Pour résumer, le langage ne modélise que les éléments suivants : le système (composé de processus), les données personnelles, les finalités de traitement, tous représentés par des ensembles. De plus, CSpeL permet de spécifier formellement les relations existant entre ces éléments, représentées par des fonctions. Ainsi, le système dans son contexte lié à la vie privée est défini formellement par la définition 8 :

Définition 8: Contexte

Un **contexte** $\mathcal{C}$ est un 6-uplet $(S, D, P, \gamma, \pi, \nu)$ où S est un ensemble fini de **processus** (*processes*), D un ensemble fini de **données personnelles** (*personal data*), P un ensemble fini de **finalités** de traitement (*purposes*) et, γ, π et ν sont des fonctions définies dans les domaines suivants :

$$\begin{cases} \gamma & \in & D \times P \to bool \\ \pi & \in & S \to \mathcal{P}(P) \\ \nu & \in & S \to \mathcal{P}(D) \end{cases}$$

— La fonction totale γ représente le **consentement** de la personne concernée (*Data Subject*), en définissant pour chaque donnée et pour chaque finalité si la personne concernée a consenti, ou non, au traitement de cette donnée pour cette finalité de traitement ;

— La fonction totale π représente les **significations du traitement** (c'est-à-dire à quoi sert le traitement), en définissant pour chaque processus du système ses finalités de traitement ;

— La fonction totale ν représente pour chaque processus du système les données personnelles qui lui sont **nécessaires** pour effectuer le traitement.

Dans la suite, $\top$ est utilisé pour dénoter la valeur booléenne "vrai" et $\bot$ pour dénoter la valeur booléenne "faux". Ainsi, si la personne concernée (*Data Subject*) a consenti au traitement de la donnée d pour la finalité p, cela est formalisé par $\gamma(d, p) = \top$. De façon similaire, si la personne concernée n'a pas consenti au traitement de la donnée d pour la finalité p cela est formalisé par $\gamma(d, p) = \bot$. Un processus s ayant été défini uniquement pour la finalité p est formalisé par $\pi(s) = \{p\}$. Un processus s ayant uniquement besoin de la donnée personnelle d est formalisé par $\nu(s) = \{d\}$.

Les exemples suivants illustrent la formalisation de l'exemple GDM_{Spl}, introduit dans la section 7.3.1, aux différents niveaux d'abstractions considérés lors de sa définition. La terminologie utilisée dans la définition de GDM_{Spl} est reprise pour cette formalisation (par exemple : $P1$, $T11$ ou bien encore *makePrescr*).

Exemple 4: Formalisation de GDM_{Spl} au niveau Modèle Graphique

$$S \triangleq \{P1; P2\}$$
$$D \triangleq \{EPR\}$$
$$P \triangleq \{Traitement; Recherche\}$$
$$\gamma \triangleq \begin{cases} (EPR, Traitement) & \mapsto & \top \\ (EPR, Recherche) & \mapsto & \bot \end{cases}$$
$$\pi \triangleq \begin{cases} P1 & \mapsto & \{Traitement\} \\ P2 & \mapsto & \{Recherche\} \end{cases} \qquad \nu \triangleq \begin{cases} P1 & \mapsto & \{EPR\} \\ P2 & \mapsto & \{EPR\} \end{cases}$$

L'Exemple 4 présente la formalisation de GDM$_{Spl}$ au niveau Modèle Graphique. À ce niveau d'abstraction, deux processus ont été définis : P1 pour réaliser le traitement médical et P2 pour réaliser l'essai clinique. Les données personnelles sont contenues dans le dossier du patient, EPR. Les deux processus auront besoin d'avoir accès à ces données. Dans mon exemple, la personne concernée par les données personnelles a consenti au traitement de ces dernières uniquement à des fins de traitement médical.

Exemple 5: Formalisation de GDM$_{Spl}$ au niveau Modèle Formel

$$
\begin{array}{rcl}
S & \triangleq & \{S1; T10; T11; T12; T13; T14; E1; S2; T22; T23; E2\} \\
D & \triangleq & \{\pounds epr\} \\
P & \triangleq & \{Traitement, Recherche\} \\
\gamma & \triangleq & \left\{ \begin{array}{lcl} (\pounds epr, Traitement) & \mapsto & \top \\ (\pounds epr, Recherche) & \mapsto & \bot \end{array} \right. \\
\pi & \triangleq & \left\{ \begin{array}{lcl} S1 & \mapsto & \{Traitement\} \\ T10 & \mapsto & \{Traitement\} \\ T11 & \mapsto & \{Traitement\} \\ T12 & \mapsto & \{Traitement\} \\ T13 & \mapsto & \{Traitement\} \\ T14 & \mapsto & \{Traitement\} \\ E1 & \mapsto & \{Traitement\} \\ S2 & \mapsto & \{Recherche\} \\ T22 & \mapsto & \{Recherche\} \\ T23 & \mapsto & \{Recherche\} \\ E2 & \mapsto & \{Recherche\} \end{array} \right. \\
\nu & \triangleq & \left\{ \begin{array}{lcl} S1 & \mapsto & \{\pounds epr\} \\ T10 & \mapsto & \{\pounds epr\} \\ T11 & \mapsto & \{\pounds epr\} \\ T12 & \mapsto & \{\pounds epr\} \\ T13 & \mapsto & \{\pounds epr\} \\ T14 & \mapsto & \{\pounds epr\} \\ E1 & \mapsto & \{\pounds epr\} \\ S2 & \mapsto & \{\pounds epr\} \\ T22 & \mapsto & \{\pounds epr\} \\ T23 & \mapsto & \{\pounds epr\} \\ E2 & \mapsto & \{\pounds epr\} \end{array} \right.
\end{array}
$$

L'Exemple 5 présente la formalisation de GDM$_{Spl}$ au niveau Modèle Formel. À ce niveau d'abstraction, les processus P1 et P2 ont été raffinés en plusieurs processus (appelés lignes de vie dans IAT) : S1, T10, T11, T12, T13, T14 et E1 pour P1 ainsi que S2, T22, T23 et E2 pour P2. Les données personnelles correspondent à la variable globale $\pounds epr$. Tous ces processus ont besoin d'utiliser ces données. Les finalités de traitement n'ont pas changé. L'association des finalités est conservée, c'est à dire que les processus raffinant P1 gardent la même finalité que P1, à savoir Traitement. Réciproquement, les processus raffinant P2 gardent la finalité de P2, à savoir Recherche. Le consentement est raffiné de la même façon, c'est à dire que le consentement, de la personne concernée, pour le traitement de ses données personnelles ($\pounds epr$ à ce niveau d'abstraction) n'est donné que pour la finalité de traitement médical, à savoir Traitement.

L'Exemple 6 présente la formalisation de GDM$_{Spl}$ au niveau Programme. À ce niveau d'abstraction, les processus ont encore été raffinés : T13 en `makePrescr`, T22 en `getData` et T23 en `computeStats`. Dans notre exemple simple, seuls les processus automatisables (i.e. sans intervention humaine) sont gardés. Ces proces-

sus correspondent à des fonctions dans le code logiciel du système. Les finalités de traitement n'ont pas changé.

Exemple 6: Formalisation de GDM$_{Spl}$ au niveau Programme

$$
\begin{aligned}
S &\triangleq \{\texttt{makePrescr}; \texttt{getData}; \texttt{computeStats}\};\\
D &\triangleq \{Patient, TrialData\}\\
P &\triangleq \{Traitement, Recherche\}\\
\gamma &\triangleq \begin{cases} (Patient, Traitement) & \mapsto \top \\ (Patient, Recherche) & \mapsto \bot \\ (TrialData, Traitement) & \mapsto \bot \\ (TrialData, Recherche) & \mapsto \bot \end{cases}\\
\pi &\triangleq \begin{cases} \texttt{makePrescr} & \mapsto \{Traitement\} \\ \texttt{getData} & \mapsto \{Recherche\} \\ \texttt{computeStats} & \mapsto \{Recherche\} \end{cases}\\
\nu &\triangleq \begin{cases} \texttt{makePrescr} & \mapsto \{Patient\} \\ \texttt{getData} & \mapsto \{Patient, TrialData\} \\ \texttt{computeStats} & \mapsto \{TrialData\}. \end{cases}
\end{aligned}
$$

Les données personnelles sont raffinées et correspondent aux structures de données $Patient$ et $TrialData$. Dans mon exemple, je considère qu'une posologie (i.e. une variable de type `Dos`) n'est pas, en elle même, une donnée personnelle, et donc ne fait pas partie des types considérés comme des données personnelles. Si elle est ajoutée à un patient, par exemple dans de le cadre d'une ordonnance, alors elle devient une donnée personnelle, car elle est alors mise dans un champ de la structure de type `Patient`, qui elle est une donnée personnelle. Au contraire, si une variable de type `Dos` est ajoutée dans une liste pour calculer un inventaire, alors elle n'est pas considérée comme une donnée personnelle. Les risques de ré-identification [1] sur ce type de données médicales, par exemple lors de la prescription de médicaments rares, ne sont pas traités dans le cadre de cette thèse.

Telles qu'elles sont définies dans l'exemple au niveau Programme (détaillé en section 7.3.1), et selon ce qui est défini dans la signature des fonctions, la fonction `makePrescr` a besoin des données de type $Patient$, `getData` des données de type $Patient$ et de type $TrialData$, et `computeStats` a seulement besoin des données de type $TrialData$. L'association des finalités est conservée, c'est à dire que les processus raffinant un processus p de plus haut niveau gardent la même finalité que p. Ainsi, `makePrescr` est définie pour la finalité Traitement, tandis que `getData` et `computeStats` le sont pour la finalité Recherche. Pour le consentement, dans notre exemple, seul le traitement des données de type $Patient$ pour la finalité de traitement médical est consenti par la personne concernée (*Data Subject*).

D'une façon équivalente, cette instanciation peut être réalisée sur les autres

1. https://en.wikipedia.org/wiki/Data_re-identification

exemples définis en section 7. Ces formalisations sont détaillées en Annexe A.2.1 pour l'exemple Gestion Données Médicales - version Complexe, en Annexe A.3.1 pour l'exemple Achat en Ligne - version Simple et en Annexe A.4.1 pour l'exemple Achat en Ligne - version Complexe.

Ces exemples montrent que l'on peut formaliser à l'aide du langage CSpeL, différents systèmes dans leur contexte lié à la vie privée, à la fois au niveau Modèle (Graphique ou Formel), qu'au niveau Programme, et que ce langage peut être utilisé dans différents domaines d'applications, le but de l'approche étant de vérifier qu'un système respecte des propriétés liées à la vie privée, notamment des propriétés liées au consentement. Maintenant qu'il est possible de formaliser le système dans son contexte, on veut pouvoir vérifier des exécutions de ce système ainsi formalisé. Pour cela nous allons également formaliser les traces d'exécution.

8.1.2 . Traces d'exécution

Pour permettre de fournir des garanties sur le traitement des données personnelles à l'aide d'une approche formelle, j'ai formalisé le système dans son contexte lié à la vie privée. Néanmoins, ces garanties sont également liées à l'exécution du système, qu'il faut donc pouvoir également l'exprimer de façon formelle. L'expression et la vérification de mes propriétés sont fondées sur les traces (d'exécution) du système, modélisé dans un contexte $\mathcal{C}$. La définition 9 présente la définition formelle d'une trace d'exécution dans CSpeL. Cette notion de trace est suffisamment abstraite pour représenter des traces, soit au niveau de l'exécution d'un modèle, soit au niveau de l'exécution d'un programme. Elle est suffisamment expressive pour permettre de vérifier mes propriétés formelles.

Définition 9: Trace d'exécution

Soit $\mathcal{C} = (S, D, P, \gamma, \pi, \nu)$ un contexte,
$\{s_i\}_{i\in\mathbb{N}} \subseteq S$ un ensemble de processus, et $\{d_i\}_{i\in\mathbb{N}}$ un ensemble de données (personnelles ou non personnelles). Les traces d'exécution de $\mathcal{C}$ sont définies selon la grammaire suivante :

$$T ::= \epsilon \mid Handle(s_i, d_i); T.$$

La trace ϵ est la trace vide, tandis que la trace $Handle(s_i, d_i); T$ correspond à l'exécution de l'évènement $Handle(s_i, d_i)$ indiquant que le processus s_i traite la donnée d_i, suivi par le reste T de la trace.

Il convient de noter que les données contenues dans les traces d'exécution peuvent être soit des données personnelles, soit des données non personnelles. J'ai fait ce choix afin de garder l'hypothèse que le système n'a pas de notions liées à la vie privée. Par exemple, il n'y a pas de distinction entre les données personnelles et les données non personnelles dans le système. Ces notions liées à la vie privée sont spécifiées par le contexte $\mathcal{C}$, indiqué dans le fichier CSpeL. De plus, pour chaque événement, une seule donnée est traitée à la fois par un processus. Lorsqu'un pro-

cessus traite plusieurs données, un évènement par donnée est présent dans la trace. Par exemple, le traitement des données d_1 et d_2 par le processus s est représenté par la concaténation des deux évènements $Handle(s, d_1)$ et $Handle(s, d_2)$. Même si d'autres types d'évènements pourraient être représentés, je ne considère que les évènements correspondant aux traitements d'une donnée par un processus, car cela est suffisant pour vérifier mes propriétés définies en section 8.1.3.

L'exemple 7 illustre La formalisation de traces d'exécution possibles pour le cas d'utilisation GDM_{Spl} introduit dans la section 7.3.1, aux différents niveaux d'abstractions considérés lors de la définition de cet exemple. Le nommage utilisé dans la définition de l'exemple reprend celui de la formalisation (par exemple : $P1$, $T11$ ou bien encore *makePrescr*).

Exemple 7: Traces d'execution pour l'exemple GDM_{Spl}

Considérons l'exemple GDM_{Spl} introduit dans la section 7.3.1, en s'intéressant à la finalité du **traitement médical**.

Au niveau Modèle Graphique,

la trace

$$Handle(P1, EPR); \epsilon$$

dénote le traitement de EPR par le processus $P1$, ce qui correspond, à ce niveau-là, à la réalisation du traitement médical.

Au niveau Modèle Formel,

la trace

$$Handle(T11, £epr); Handle(T12, £epr); Handle(T13, £epr); \epsilon$$

dénote la réalisation du traitement médical au niveau Modèle Formel. Les lignes de vie S1, T14 et E1 ne traitant aucune donnée, elles ne sont pas présentes dans les traces.

Au niveau Programme,

la trace

$$Handle(\texttt{makePrescr}, Patient); Handle(\texttt{makePrescr}, Dos); \epsilon$$

dénote l'exécution de la fonction `makePresc` pour la délivrance d'une prescription d'un patient. La fonction `makePresc` traite donc deux types de données, $Patient$ et Dos. La trace d'exécution de la fonction est composée de deux évènements $Handle(s_i, d_i)$, un pour chaque type de donnée. Pour rappel, nous considérons dans notre exemple que $Patient$ est une donnée personnelle mais pas Dos. Par contre l'accès à une donnée de type Dos *via* une donnée de type $Patient$ est considéré comme un traitement d'une donnée personnelle (car cela sera considéré comme un traitement d'une donnée de type $Patient$, suivi d'un traitement d'une donnée de type Dos).

Des formalisations similaires peuvent être réalisées pour des traces d'exécutions pour la réalisation de l'essai clinique, et également pour les autres exemples définis en section 7. Ces formalisations sont disponibles en Annexe A.1.1 pour des traces d'exécutions concernant l'autre finalité de traitement de cet exemple, c'est à dire l'essai clinique, en Annexe A.2.2 pour l'exemple GDM$_{Cpx}$, en Annexe A.3.2 pour l'exemple AL$_{Spl}$ et en Annexe A.4.2 pour l'exemple AL$_{Cpx}$.

Il est donc possible de formaliser à l'aide de mon langage CSpeL des traces d'exécution, en s'appuyant sur la formalisation du système dans un contexte spécifique, aussi bien aux niveaux Modèle Graphique, Modèle Formel que Programme. Maintenant que l'on a formalisé les traces d'exécution, intéressons-nous aux propriétés de consentement que nous allons vérifier sur ces traces, selon un contexte spécifique.

8.1.3 . Propriétés de traitement

Comme dit précédemment, mon but est de vérifier qu'un système respecte des propriétés liées au traitement des données dans le cadre du respect de la vie privée. Je propose de l'atteindre en validant les exécutions du système que j'ai formalisées précédemment. Pour cela, je propose de vérifier qu'une trace d'exécution T respecte une propriété $prop$ dans un contexte $\mathcal{C}$, ce qui est noté $\mathcal{C} \vdash_{prop} T$. La sémantique des éléments contenus dans T est définie en section 8.1.2 et celle des éléments contenus dans $\mathcal{C}$ en section 8.1.1. On peut noter qu'une autre notation, potentiellement plus standard, $\mathcal{C}, T \vdash prop$, inspirée des logiques temporelles en temps linéaire (*linear-time temporal logics*) [81], aurait pu être utilisée. Dans mon cas, la propriété étant fixe et l'induction étant réalisée sur la trace et non sur la propriété, la notation que j'ai choisie, inspirée des relations de type [135], convient mieux à mon analyse. Par rapport aux relations de type, mes traces peuvent être vues comme une abstraction des instructions du programme et les propriétés comme une généralisation des types.

D'après la définition fournie par GePyR, en section 4, la *caractérisation du Traitement* regroupe les notions liées au traitement autorisé des données personnelles. En particulier, on peut distinguer quatre grandes notions incluses dans le traitement des données personnelles : la finalité des traitements, la nécessité des données (pour le traitement), la durée du traitement et l'évolution du consentement (le droit à tout moment d'accorder ou de révoquer le consentement). Dans mon approche, je m'intéresse dans un premier temps aux notions de finalité des traitements et de nécessité des données. Plus précisément, j'aimerais vérifier que lorsqu'il y a un traitement d'une donnée personnelle, la personne concernée a donné son consentement pour au moins une des finalités de traitement. De plus, j'aimerais également vérifier que cette donnée est réellement nécessaire au traitement. J'exprime formellement ces propriétés à travers les notions de *conformité aux finalités consenties* et *conformité à la nécessité des données*. À noter la vérification est effectuée traitement par traitement, donc la vérification de mes propriétés ne dépend pas de l'ordonnancement des traitements. La définition 10 définit ma propriété de

conformité aux finalités consenties.

Définition 10: Conformité aux finalités consenties

L'évènement $Handle(s, d)$ **est *conforme aux finalités consenties*** selon un contexte $\mathcal{C} \triangleq (S, D, P, \gamma, \pi, \nu)$, ce qui est noté $\mathcal{C} \vdash_P Handle(s, d)$, si et seulement si la donnée traitée n'est pas une donnée personnelle, ou bien si parmi les finalités du processus, au moins l'une d'elles fait partie de celles pour lequel un consentement a été donné, c'est à dire :

$$\frac{d \notin D}{\mathcal{C} \vdash_P Handle(s, d)} \qquad \frac{\exists p \in \pi(s).\gamma(d, p) = \top}{\mathcal{C} \vdash_P Handle(s, d)}$$

Le fait que je ne vérifie pas toutes les finalités associées au processus, mais seulement qu'il en existe une, peut paraître contre-intuitif. Reprenons l'exemple AL_{Spl}, détaillé en section 7.4.1.

Au niveau Modèle Formel, le système comporte trois processus : **P1** et **P2** pour la finalité **Administratif** et **P3** pour la finalité **Marketing**. Lors du raffinement entre le niveau Modèle Graphique et le niveau Modèle Formel, résumé dans le tableau 7.6, on peut remarquer que les ensembles des processus raffinant P2 et P3 ont un processus en commun : **T21** (permettant d'accéder à l'historique d'achat). T21 est donc associé aux deux finalités. Pourtant, lorsque T21 est exécuté (ce qui correspond à l'évènement $Handle(T21, d)$ avec d une donnée), il n'est pas possible, atomiquement, de garantir pour quelle finalité exactement. Les deux finalités pouvant avoir été consenties différemment (comme dans notre exemple), imposer que les deux finalités soient consenties impliquerait que le traitement ne respecte pas la propriété, alors que le traitement serait en réalité conforme. Ceci explique pourquoi la propriété est vraie si "au moins" une finalité est consentie.

Par contre, dans le cas d'un traitement illicite (c'est à dire pour la finalité non consentie), le non respect de la propriété ne sera pas détecté lors de l'exécution de T21. Néanmoins, il sera détecté lors de l'exécution des autres processus correspondant à la finalité non consentie. Par exemple, si T21 est exécuté pour la finalité **Administratif**, alors d'autres processus raffinant **P2** seront également exécutés, tel que T13 (afficherHistorique). En effet, d'après le processus de raffinement défini en section 7.2, des processus (de niveau supérieur) avec des finalités différentes sont raffinés en des ensembles distincts. Ainsi, même si T21 fait partie des deux ensembles raffinant P2 et P3, ces ensembles étant distincts, au moins un autre T_i permettra de distinguer les deux cas. Donc, si T21 est exécuté pour la finalité **Marketing**, alors d'autres processus raffinant **P3** seront également exécutés, tel que T31 (choisirPublicite).

Définissons maintenant une deuxième propriété, la *conformité à la nécessité des données*.

Définition 11: Conformité à la nécessité des données

L'évènement $Handle(s, d)$ **est *conforme à la nécessité des données*** selon un contexte $\mathcal{C} \triangleq (S, D, P, \gamma, \pi, \nu)$, ce qui est noté $\mathcal{C} \vdash_N Handle(s, d)$, si et seulement si la donnée n'est pas personnelle, ou si la donnée personnelle d est nécessaire pour le traitement, i.e. :

$$\frac{d \notin D}{\mathcal{C} \vdash_N Handle(s, d)} \qquad \frac{d \in \nu(s)}{\mathcal{C} \vdash_N Handle(s, d)}$$

Les notions de *conformité aux finalités consenties* et la *conformité à la nécessité des données* sont étendues pour une trace d'exécution à l'aide de deux règles d'inférence indiquées sur la figure 8.2. La trace vide (ϵ) est toujours *conforme* aussi bien *aux finalités consenties* qu'à la nécessité des données, quel que soit le contexte. Une trace non vide est *conforme aux finalités consenties* (respectivement, *conforme à la nécessité des données*) si et seulement si chacun des évènements qui la composent est *conforme aux finalités consenties* (respectivement, *conforme à la nécessité des données*).

$$\frac{}{\mathcal{C} \vdash_{prop} \epsilon} \qquad \frac{\mathcal{C} \vdash_{prop} Handle(s, d) \quad \mathcal{C} \vdash_{prop} T}{\mathcal{C} \vdash_{prop} Handle(s, d); T}$$

avec $prop \in \{P; N\}$

Figure 8.2 – Conformité d'une trace selon un contexte $\mathcal{C}$.

La notation formelle utilisée dans ces définitions et règles d'inférences sont légèrement différentes de celles publiées dans [41]. Leur signification reste la même mais la nouvelle notation est plus homogène.

L'exemple 8 présente des preuves de conformité sur des traces spécifiées précédemment pour l'exemple GDM_{Spl} au niveau Modèle Graphique (Exemple 7).

Exemple 8: Exemples de preuves - GDM$_{Spl}$ au niveau Modèle Graphique

Considérons à nouveau notre exemple au niveau graphique, formalisé précédemment, $\mathcal{C} = (S, D, P, \gamma, \pi, \nu)$, défini par :

$$S \triangleq \{P1; P2\};$$
$$D \triangleq \{EPR\};$$
$$P \triangleq \{Traitement; Recherche\}$$
$$\gamma \triangleq \begin{cases} (EPR, Traitement) & \mapsto & \top \\ (EPR, Recherche) & \mapsto & \bot \end{cases}$$
$$\pi \triangleq \begin{cases} P1 & \mapsto & \{Traitement\} \\ P2 & \mapsto & \{Recherche\} \end{cases}$$
$$\nu \triangleq \begin{cases} P1 & \mapsto & \{EPR\} \\ P2 & \mapsto & \{EPR\} \end{cases}$$

Vérifions si les traces $Handle(P1, EPR); \epsilon$ et $Handle(P2, EPR); \epsilon$ respectent les propriétés définies ici.

Propriété de conformité aux finalités consenties :
Intuitivement, la première trace est *conforme aux finalités consenties*, car le consentement a été accordé de traiter *EPR* pour la finalité $Traitement$, et donc par $P1$. De même, intuitivement, la seconde trace ne l'est pas, car il n'y a pas de finalité consentie pour le traitement de *EPR* par $P2$. En effet, la seule finalité associée à $P2$ est $Recherche$, or le consentement au traitement des données personnelles n'a pas été accordé pour cette finalité. Prouvons le.

Nous voulons donc prouver que la **première trace**, $Handle(P_1, EPR); \epsilon$ est *conforme aux finalités consenties*, i.e. :

$$\mathcal{C} \vdash_P Handle(P_1, EPR); \epsilon$$

Pour cela, faisons l'arbre de preuve suivant :

$$\frac{\dfrac{\exists p \in \pi(P_1).\gamma(EPR, p) = \top}{\mathcal{C} \vdash_P Handle(P_1, EPR)} \text{ avec } p = \text{Traitement} \qquad \dfrac{}{\mathcal{C} \vdash_P \epsilon}}{\mathcal{C} \vdash_P Handle(P_1, EPR); \epsilon}$$

Donc la **première trace** est bien *conforme aux finalités consenties* dans le contexte $\mathcal{C}$.

Prouvons maintenant que la **seconde trace** n'est pas *conforme aux finalités consenties*, i.e. :

$$\mathcal{C} \nvdash_P Handle(P_2, EPR); \epsilon$$

Nous prouvons cette propriété par contradiction. Supposons donc que cette trace est *conforme aux finalités consenties*, i.e. :

$$\mathcal{C} \vdash_P Handle(P_2, EPR); \epsilon$$

1. En partant de cette propriété et selon la seconde règle d'inférence de la figure 8.2, $\mathcal{C} \vdash_P Handle(P2, EPR)$ est vraie.
2. Donc, selon la définition de *conformité aux finalités consenties*, soit $EPR \notin D$, soit $\gamma(EPR, p) = \top$ pour certaine une finalité p dans $\pi(P_2)$.
3. Le premier cas contredit la définition de D : EPR est une donnée personnelle dans $\mathcal{C}$.
4. Considérons alors l'autre cas. Selon la définition de π, $Recherche$ est la seule finalité de P_2. Cependant, $\gamma(EPR, Recherche) = \bot$, ce qui contredit $\gamma(EPR, Recherche) = \top$.
5. Chaque cas conduit à une contradiction, donc l'hypothèse initiale est fausse. $\mathcal{C} \vdash_P Handle(P_2, EPR); \epsilon$ est donc fausse.

Ainsi, la **seconde trace** n'est pas *conforme aux finalités consenties* dans le contexte $\mathcal{C}$.

Propriété de conformité à la nécessité des données :
Intuitivement ces deux traces sont *conformes à la nécessité des données* car *EPR* est nécessaire pour $P1$ et pour $P2$.
Prouvons le.

Nous voulons donc prouver que la **première trace**, $Handle(P_1, EPR); \epsilon$ est *conforme à la nécessité des données*, i.e. :

$$\mathcal{C} \vdash_N Handle(P_1, EPR); \epsilon.$$

Pour cela, faisons l'arbre de preuve suivant :

$$\frac{\dfrac{EPR \in \nu(P_1)}{\mathcal{C} \vdash_N Handle(P_1, EPR)} \qquad \dfrac{}{\mathcal{C} \vdash_N \epsilon}}{\mathcal{C} \vdash_N Handle(P_1, EPR); \epsilon}$$

Donc la **première trace** est bien *conforme à la nécessité des données* dans le contexte $\mathcal{C}$.

Nous voulons également prouver que la **deuxième trace**, $Handle(P_2, EPR); \epsilon$ est *conforme à la nécessité des données*, i.e. :

$$\mathcal{C} \vdash_N Handle(P_2, EPR); \epsilon$$

Pour cela, faisons l'arbre de preuve suivant :

$$\frac{\dfrac{EPR \in \nu(P_2)}{\mathcal{C} \vdash_N Handle(P_2, EPR)} \qquad \dfrac{}{\mathcal{C} \vdash_N \epsilon}}{\mathcal{C} \vdash_N Handle(P_2, EPR); \epsilon}$$

Donc la deuxième trace est bien *conforme à la nécessité des données* dans le contexte $\mathcal{C}$.

L'exemple 9 présente des preuves de conformité sur des traces spécifiées précédemment pour l'exemple GDM$_{Spl}$ au niveau Programme (Exemple 7).

Exemple 9: Exemples de preuves - GDM$_{Spl}$ au niveau Programme

Considérons à nouveau GDM$_{Spl}$ au niveau Programme, formalisé précédemment (Exemple 6), $\mathcal{C} = (S, D, P, \gamma, \pi, \nu)$, défini par :

$$S \triangleq \{\texttt{makePrescr}; \texttt{getData}; \texttt{computeStats}\};$$
$$D \triangleq \{Patient, TrialData\};$$
$$P \triangleq \{Traitement, Recherche\};$$
$$\gamma \triangleq \begin{cases} (Patient, Traitement) & \mapsto \top \\ (Patient, Recherche) & \mapsto \bot \\ (TrialData, Traitement) & \mapsto \bot; \\ (TrialData, Recherche) & \mapsto \bot; \end{cases}$$
$$\pi \triangleq \begin{cases} \texttt{makePrescr} & \mapsto \{Traitement\} \\ \texttt{getData} & \mapsto \{Recherche\} \\ \texttt{computeStats} & \mapsto \{Recherche\}; \end{cases}$$
$$\nu \triangleq \begin{cases} \texttt{makePrescr} & \mapsto \{Patient\} \\ \texttt{getData} & \mapsto \{Patient, TrialData\} \\ \texttt{computeStats} & \mapsto \{TrialData\}. \end{cases}$$

Vérifions si les traces

$$Handle(\texttt{makePrescr}, Patient); Handle(\texttt{makePrescr}, Dos); \epsilon$$

et

$$Handle(\texttt{getData}, Patient); Handle(\texttt{getData}, TrialData); \\ Handle(\texttt{computeStats}, TrialData); \quad \epsilon$$

sont conformes aux propriétés définies en définition 10 et en définition 11. Ces traces correspondent au raffinement des traces évaluées précédemment

au niveau Modèle Graphique.

Propriété de conformité aux finalités consenties :
Intuitivement la première trace est *conforme aux finalités consenties*, car le consentement a été accordé pour le traitement de *Patient* pour la finalité $Traitement$ associée à `makePrescr`, et *Dos* n'est pas une donnée personnelle. Cependant, la seconde ne devrait pas l'être, car la finalité de `getData` est $Recherche$ et le consentement pour le traitement de *Patient* (ou *TrialData*) n'est pas accordé pour cette finalité.

Prouvons que la **première trace** est *conforme aux finalités consenties*, i.e. :

$$\mathcal{C} \vdash_P Handle(\texttt{makePrescr}, Patient); Handle(\texttt{makePrescr}, Dos); \epsilon$$

$$\dfrac{\dfrac{\exists p \in \pi(\texttt{makePrescr}).\gamma(Patient, p) = \top}{\mathcal{C} \vdash_P Handle(\texttt{makePrescr}, Patient)} \text{ avec } p = \text{Traitement} \quad \vdots\ (1)}{\mathcal{C} \vdash_P Handle(\texttt{makePrescr}, Patient); Handle(\texttt{makePrescr}, Dos); \epsilon}$$

$$\dfrac{Dos \notin D \qquad \overline{\mathcal{C} \vdash_P \epsilon}}{\mathcal{C} \vdash_P Handle(\texttt{makePrescr}, Dos); \epsilon}\ (1)$$

Donc, la **première trace** est *conforme aux finalités consenties* dans le contexte $\mathcal{C}$.

Prouvons maintenant que la **seconde trace** n'est pas *conforme aux finalités consenties*, i.e. :

$$\mathcal{C} \nvdash_P Handle(\texttt{getData}, Patient); Handle(\texttt{getData}, TrialData); \\ Handle(\texttt{computeStats}, TrialData); \epsilon.$$

Pour cela, procédons par contradiction. Supposons donc que cette trace est *conforme aux finalités consenties*, i.e. :

$$\mathcal{C} \vdash_P Handle(\texttt{getData}, Patient); Handle(\texttt{getData}, TrialData); \\ Handle(\texttt{computeStats}, TrialData); \epsilon.$$

1. À partir de cette propriété et selon la seconde règle d'inférence de la figure 8.2, $\mathcal{C} \vdash_P Handle(\texttt{getData}, Patient)$ est vraie.

2. Donc, selon la définition de *conformité aux finalités consenties*, soit $Patient \notin D$, soit $\gamma(Patient, p) = \top$ pour une finalité p dans $\pi(\texttt{getData})$.
3. Le premier cas contredit la définition de $D : Patient$ est une donnée personnelle dans $\mathcal{C}$.
4. Considérons le second cas. Selon la définition de π, $Recherche$ est la seule finalité de `getData`. Cependant, $\gamma(Patient, Recherche) = \bot$, ce qui contredit $\gamma(Patient, Recherche) = \top$.
5. Chaque cas conduisant à une contradiction, l'hypothèse initiale ne peut être vraie. Donc

   $$\mathcal{C} \vdash_P Handle(\texttt{getData}, Patient); Handle(\texttt{getData}, TrialData); Handle(\texttt{computeStats}, TrialData); \epsilon$$

   est fausse.

Ainsi, la **seconde trace** n'est pas *conforme aux finalités consenties* dans le contexte $\mathcal{C}$.

Propriété de conformité à la nécessité des données :
Intuitivement, ces deux traces sont conformes. Comme `makePrescr` a besoin de *Patient*, la première trace est *conforme à la nécessité des données*. Similairement, `getData` a besoin de *Patient*, *TrialData*, et `computeStats` a aussi besoin de *TrialData*, donc la seconde trace est également *conforme à la nécessité des données*.
Prouvons le.

Nous voulons donc prouver que la **première trace** est *conforme à la nécessité des données*, i.e. :

$$\mathcal{C} \vdash_N Handle(\texttt{makePrescr}, Patient); Handle(\texttt{makePrescr}, Dos); \epsilon$$

$$\frac{\dfrac{Patient \in \nu(\texttt{makePrescr})}{\mathcal{C} \vdash_N Handle(\texttt{makePrescr}, Patient)} \quad \begin{matrix}\vdots \\ (1) \\ \vdots\end{matrix}}{\mathcal{C} \vdash_N Handle(\texttt{makePrescr}, Patient); Handle(\texttt{makePrescr}, Dos); \epsilon}$$

$$\frac{\dfrac{Dos \notin D}{\mathcal{C} \vdash_N Handle(\texttt{makePrescr}, Dos)} \quad \dfrac{}{\mathcal{C} \vdash_N \epsilon}}{\mathcal{C} \vdash_N Handle(\texttt{makePrescr}, Dos); \epsilon}\ (1)$$

Donc la **première trace** est bien *conforme à la nécessité des données* dans le contexte $\mathcal{C}$.

Nous voulons également prouver que la **seconde trace** est *conforme à la nécessité des données*, i.e. :

$$\begin{aligned}\mathcal{C} \vdash_N\ & Handle(\texttt{getData}, Patient);\\ & Handle(\texttt{getData}, TrialData);\\ & Handle(\texttt{computeStats}, TrialData); \epsilon\end{aligned}$$

$$\dfrac{\dfrac{Patient \in \nu(\texttt{getData})}{\mathcal{C} \vdash_N Handle(\texttt{getData}, Patient)}\ \vdots\ (1)}{\begin{aligned}\mathcal{C} \vdash_N\ & Handle(\texttt{getData}, Patient);\\ & Handle(\texttt{getData}, TrialData);\\ & Handle(\texttt{computeStats}, TrialData); \epsilon\end{aligned}}$$

$$\dfrac{\dfrac{TrialData \notin D}{\mathcal{C} \vdash_N Handle(\texttt{getData}, TrialData)}\ \vdots\ (2)}{\begin{aligned}\mathcal{C} \vdash_N\ & Handle(\texttt{getData}, TrialData);\\ & Handle(\texttt{computeStats}, TrialData); \epsilon\end{aligned}}\ (1)$$

$$\dfrac{\dfrac{TrialData \in \nu(\texttt{computeStats})}{\mathcal{C} \vdash_N Handle(\texttt{computeStats}, TrialData)} \qquad \dfrac{}{\mathcal{C} \vdash_N \epsilon}}{\begin{aligned}\mathcal{C} \vdash_N\ & Handle(\texttt{computeStats}, TrialData); \epsilon\end{aligned}}\ (2)$$

Donc, la **seconde trace** est bien *conforme à la nécessité des données* dans le contexte $\mathcal{C}$.

Les exemples précédents montrent qu'il est possible de vérifier que des traces sont *conformes aux finalités consenties* et *conformes à la nécessité des données*, à l'aide du même langage, CSpeL, à la fois au niveau modèle et au niveau programme. Cela montre que CSpeL est assez expressif pour pouvoir être adapté et être utilisé sur une variété de systèmes durant différentes étapes de leur cycle de vie. De plus, garder le même langage pour la vérification des propriétés pendant tout le processus de raffinement permet de garantir que le système raffiné continue de respecter les mêmes propriétés et de vérifier ainsi si le raffinement n'a pas introduit de non-

conformité. Maintenant que l'on a vérifié la conformité des traces d'exécution, regardons plus en détail la syntaxe et la grammaire de CSpeL, telles qu'elles ont été implémentées. Cela permet à l'utilisateur de spécifier un fichier en CSpeL et de le fournir à CASTT pour la vérification outillée.

8.2 . Implémentation du langage

Le formalisme introduit jusqu'ici permet de définir les notions génériques de *conformité aux finalités consenties* et de *conformité à la nécessité des données* pour tout système exécutable. Cela permet de s'adapter facilement à des systèmes et des niveaux d'abstraction variés. Pour pouvoir utiliser en pratique cette formalisation, j'ai défini le langage CSpeL. Il peut être utilisé au niveau Modèle Graphique, Modèle Formel et au niveau Programme.

Table 8.1 – Grammaire de CSpeL

M	::=	$C \mid CT$
C	::=	$\texttt{\textbackslash context}\{S,D,P,\gamma,\pi,\nu,IS\}$
		$\mid \texttt{\textbackslash context}\{S,D,P,\gamma,\pi,\nu\}$
S	::=	$\texttt{\textbackslash process}\{SSet\}$
D	::=	$\texttt{\textbackslash personalData}\{DSet\}$
P	::=	$\texttt{\textbackslash purposes}\{PSet\}$
γ	::=	$\texttt{\textbackslash isGranted}\{TSet\}$
π	::=	$\texttt{\textbackslash hasPurposes}\{APSet\}$
ν	::=	$\texttt{\textbackslash needData}\{ADSet\}$
IS	::=	$\texttt{\textbackslash init}\{SSet\}$
T	::=	$\texttt{\textbackslash trace}\ \{\ PCSet\ \}$
$SSet$	::=	s $\mid$ s,$SSet$
$DSet$	::=	d $\mid$ d, $DSet$
$PSet$	::=	p $\mid$ p, $PSet$
$TSet$	::=	(d :p) $\mid$ (d :p), $TSet$
$APSet$	::=	(s :$\{PSet\}$) $\mid$ (s :$\{PSet\}$), $APSet$
$ADSet$	::=	(s :$\{DSet\}$) $\mid$ (s :$\{DSet\}$), $ADSet$
$PCSet$	::=	$\texttt{\textbackslash handle}$(s, d); $PCSet \mid \texttt{VOID}$

Le tableau 8.1 donne la syntaxe formelle de CSpeL. Les littéraux d, s, et p sont des chaînes de caractères qui dénotent respectivement une donnée, un processus et une finalité. Un modèle CSpeL M est un contexte C, éventuellement suivi d'une trace T.

Un contexte, introduit par le mot-clé `\context`, contient 6 éléments qui correspondent à ceux de la définition 1 : S (représentant les processus), D (représentant

les données personnelles), P (représentant les finalités de traitement), γ (représentant le consentement), π (représentant la signification des traitements), et ν (représentant les données nécessaires aux traitements). Il peut également contenir un ensemble de processus IS pour spécifier où γ et ν sont initialisés. Par exemple, cela permet de spécifier que les éléments sont initialisés dans la fonction `main`, ce qui aura pour influence que cette fonction aura le droit d'avoir accès aux données, même si cela n'est pas spécifié (dans π et ν). L'ensemble S est introduit par le mot-clé `\process`, l'ensemble D par le mot-clé `\personalData`, et l'ensemble P par `\purposes`. De même, la fonction totale γ est introduite par le mot-clé `\isGranted`. La fonction π par le mot-clé `\hasPurposes`, et ν par `\needData`. Actuellement, il n'y a pas de vérification que les fonctions définies de cette manière sont totales. Cela permettrait de garantir à l'utilisateur que sa spécification est complète, mais cela n'est pas bloquant en soi pour la vérification des propriétés. Une trace est introduite par le mot-clé `\trace`. Il s'agit d'une séquence $PCSet$ de traitement de données. La séquence vide est `VOID`. Un événement de traitement de données dans une trace est introduit par le mot-clé `\handle` et associe un processus à une donnée.

L'exemple 10 présente la spécification en CSpeL pour l'exemple GDM$_{Spl}$ défini au niveau Modèle dans l'exemple 4.

Exemple 10: Spécification en CSpeL pour GDM$_{Spl}$ Modèle Graphique

```
\model {
  \context {
    \process      { P1, P2 },
    \personalData { EPR },
    \purposes     { Treatment, Research },
    \isGranted    { (EPR:Treatment) },
    \hasPurposes  { (P1: { Treatment }),
                    (P2: { Research }) },
    \needData     { (P1: { EPR }),
                    (P2: { EPR }) },
  }
  \trace          { \handle(P1,EPR); VOID }
}
```

Pour rappel, le but de cette approche est de vérifier qu'un système respecte des propriétés liées au consentement. Comme nous l'avons vu, CSpeL permet de spécifier formellement à la fois un système dans un contexte de vie privée et une trace d'exécution, indépendamment de son niveau d'abstraction (Modèle Graphique, Modèle Formel, ou Programme). Grâce à ces spécifications, nous pouvons vérifier les propriétés de consentement. Cependant, dans cette première forme, l'utilisateur

ne peut pas spécifier les propriétés qu'il souhaite vérifier. Pour le permettre, j'ai étendu le langage comme détaillé ci-dessous.

8.3 . Extension du language - Choix des propriétés

Lors de la vérification de trace, les deux propriétés définies en section 8.1.3 peuvent être vérifiées sur des traces d'exécution. Cependant, dans la version de CSpeL, présentée en section 8.2 l'utilisateur ne peut pas choisir quelle propriété il souhaite vérifier.

Le but de l'extension du langage est d'offrir la possibilité à l'utilisateur d'exprimer quelles propriétés de consentement il veut vérifier sur la trace, que ce soit une seule de ces propriétés, leur conjonction ou leur disjonction.

Table 8.2 – Extension de la grammaire pour le choix de propriété.

M	::=	$\ldots \mid CP \mid CTP$
$\ldots$		
P	::=	$\texttt{\textbackslash prop\{}\ PropExpr\ \texttt{\}}$
$PropExpr$	::=	$Prop \mid Prop\ \texttt{\&\&}\ PropExpr \mid Prop\ \texttt{||}\ PropExpr$
$Prop$	::=	$\texttt{\textbackslash necessity} \mid \texttt{\textbackslash purpose}$

La table 8.2 présente l'extension de la grammaire qui a été réalisée afin de permettre ce choix. La grammaire précédente (illustrée dans la table 8.1) est étendue avec la possibilité de spécifier un nouvel élément `\prop`. Dans cet élément, l'utilisateur va pouvoir indiquer quelle propriété il veut vérifier : la *conformité à la nécessité des données* (via `\necessity`), la *conformité aux finalités consenties* (via `\purpose`), la conjonction des deux (à l'aide du connecteur `&&`), ou bien leur disjonction (à l'aide du connecteur `||`). Si aucun choix n'est explicitement spécifié, cela revient à la conjonction de toutes les propriétés.

L'exemple 11 illustre cette spécification et son influence sur la vérification.

Exemple 11: Choix des propriétés à vérifier

Reprenons l'exemple GDM_{Spl} au niveau Programme. Son contexte est défini comme suit.

```
\context {
    \process      { makePrescr, getData, computeStats },
    \personalData { Patient, Data},
    \purposes     { Treatment, Research },
    \isGranted    { (Patient: Treatment);
                    (Data: Treatment) },
    \hasPurposes  { (makePrescr:Treatment);
```

```
                    (getData:Treatment,Research);
                    (computeStats:Research)},
    \needData     { (makePrescr:Patient);
                    (getData:Patient, Data);
                    (computeStats:Data)}
}
```

Afin de pouvoir représenter des traces valides et des traces invalides, le consentement au traitement des données de type `Data` n'a été accordé que pour la finalité *Traitement*.

Considérons les traces suivantes :

1. `\handle(makePrescr, Data); VOID`
2. `\handle(makePrescr, Patient); VOID`
3. `\handle(getData, Patient); VOID`
4. `\handle(getData, Data); VOID`

Selon le choix des propriétés à vérifier, la validité de ces traces change :

	1	2	3	4
`\purpose`	✓	✓	×	×
`\necessity`	×	✓	×	✓
`\purpose && \necessity`	×	✓	×	×
`\purpose || \necessity`	✓	✓	×	✓

Remarque : Actuellement, ce choix n'est pris en compte que lors de la vérification de trace (expliquée en section 9.2). Il faudrait le prendre également en compte lors des traductions effectuées par CASTT (expliquées en section 9.3 pour le niveau Modèle Formel et en section 9.4 pour le niveau Programme). Cela pourra faire l'objet de travaux futurs. Par ailleurs, l'ajout d'autres propriétés (par exemple, la *conformité aux durées de conservation*) impliquerait d'ajouter également ces nouvelles propriétés dans la spécification du choix décrite ci-dessus, afin que l'utilisateur puisse choisir ces nouvelles propriétés.


Cette extension a été ajoutée après la publication de [41].

Cette section a présenté CSpeL, un langage formel de spécification d'un système dans son contexte de vie privée, afin de vérifier qu'il respecte des propriétés de vie privée. CSpeL peut être utilisé durant différentes étapes du cycle de vie du système, c'est à dire que ce langage permet de spécifier le système et son contexte, à différents niveaux d'abstraction (aux niveaux Modèle Graphique, Modèle Formel et

Programme). Cette section a également présenté les définitions formelles des deux propriétés liées au consentement, la *conformité aux finalités consenties* et la *conformité à la nécessité des données*, qui sont vérifiables à partir de cette spécification. J'ai illustré l'utilisation de CSpeL sur les exemples définis précédemment.

La section suivante présente CASTT, l'outil que j'ai implémenté afin d'automatiser la vérification du respect de ces propriétés pour des systèmes spécifiés en CSpeL.

9 - Outil de vérification et de traduction de spécification pour la vérification de propriété liées à la vie privée - **CASTT**

J'ai développé l'outil CASTT[1], pour vérifier qu'un système respecte des propriétés liées à la vie privée, les propriétés de *conformité aux finalités consenties* et de *conformité à la nécessité des données*. Pour cela, l'outil repose sur une spécification du contexte (lié à la vie privée) écrite en CSpeL (mon langage de spécification détaillé en section 8). CASTT a trois fonctionnalités :

1. CASTT peut analyser une trace, correspondant à une execution du système, donnée par l'utilisateur dans le fichier de spécification écrit en CSpeL.
2. Il permet également de traduire la spécification en un langage compréhensible par un outil de vérification au niveau Modèle Formel, IAT [109], afin de vérifier des propriétés de consentement à ce niveau.
3. Enfin, j'ai aussi implémenté une traduction vers un outil au niveau Programme, Frama-C [19]. Cela permet ainsi d'analyser directement le système à ce niveau.

La section 9.1 présente le fonctionnement global de CASTT. Les différentes fonctionnalités de CASTT sont ensuite détaillées en section 9.2 pour l'analyse de trace, en section 9.3 pour la traduction au niveau Modèle Formel, et en section 9.4 pour la traduction au niveau Programme.

9.1 . Présentation globale de **CASTT**

La figure 9.1 présente le fonctionnement global de CASTT. Cet outil permet de vérifier à partir d'une spécification du système, dans un contexte de vie privée, si ce système traite les données personnelles en respectant des propriétés liées à la vie privée. Cette spécification est écrite en CSpeL, détaillé en section 8. Les propriétés que le système doit respecter sont définies en section 8.1.3. Pour effectuer cette vérification, selon les besoins, CASTT peut se contenter d'une spécification du contexte et d'une trace d'execution formalisés en CSpeL. Il est également possible de fournir en entrée un modèle formel du système à vérifier ou le code lui correspondant. Dans ce cas, fournir la trace d'exécution n'est pas nécessaire, mais CASTT doit être alors utilisé conjointement avec un outil externe spécifique au niveau où s'effectue la vérification (Modèle Formel ou Programme).

La figure 9.2 présente le fonctionnement de CASTT selon les besoins de l'utilisateur. En effet, si l'utilisateur veut vérifier un système à partir d'une trace d'exé-

1. Le code de l'outil n'est pas sur un répertoire public, mais une archive avait été fournie aux relecteurs lors de la soumission de notre article à TAP [41].

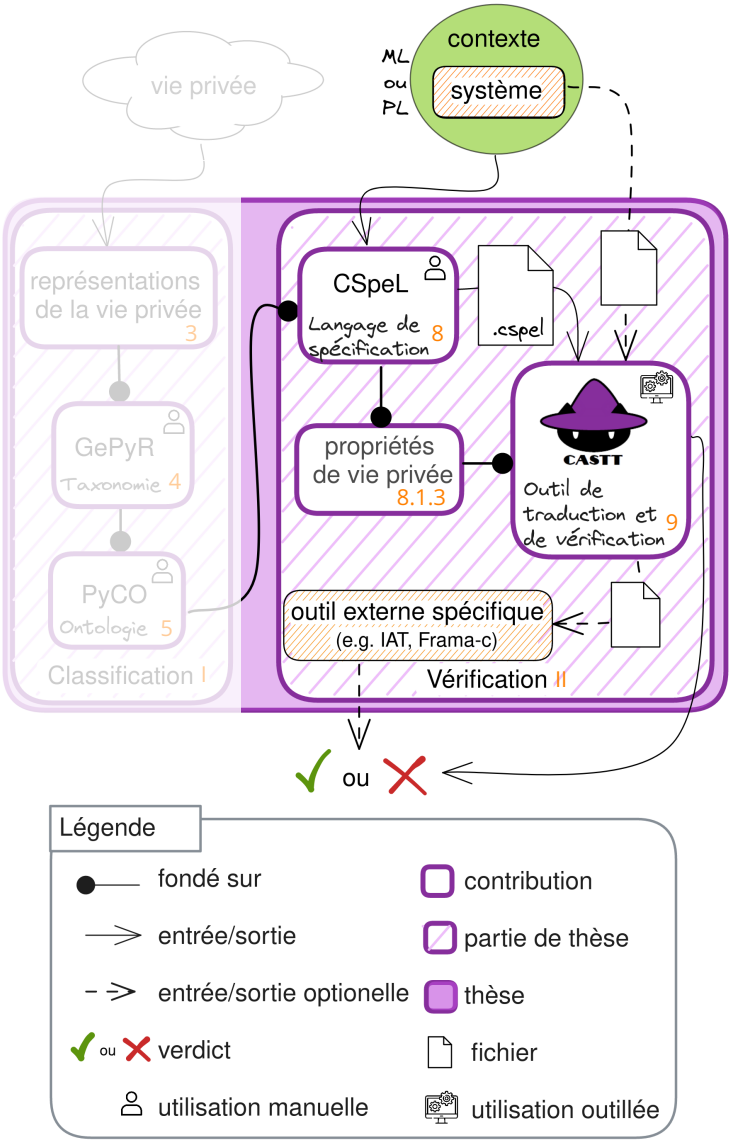


Figure 9.1 – Illustration des liens entre CASTT et CSpeL.

cution, il peut fournir simplement un fichier CSpeL (contenant la spécification du système ainsi que la trace à évaluer). L'outil vérifiera alors que la trace fournie est valide ou non selon la spécification. Cette fonctionnalité peut être utilisée quel que soit le niveau d'abstraction du système (Modèle ou Programme). Ce mécanisme de vérification de trace est détaillé dans la section suivante (section 9.2). Il est en revanche nécessaire que l'utilisateur puisse fournir la trace à vérifier, ce qui nécessite à ce stade une écriture manuelle par l'utilisateur.

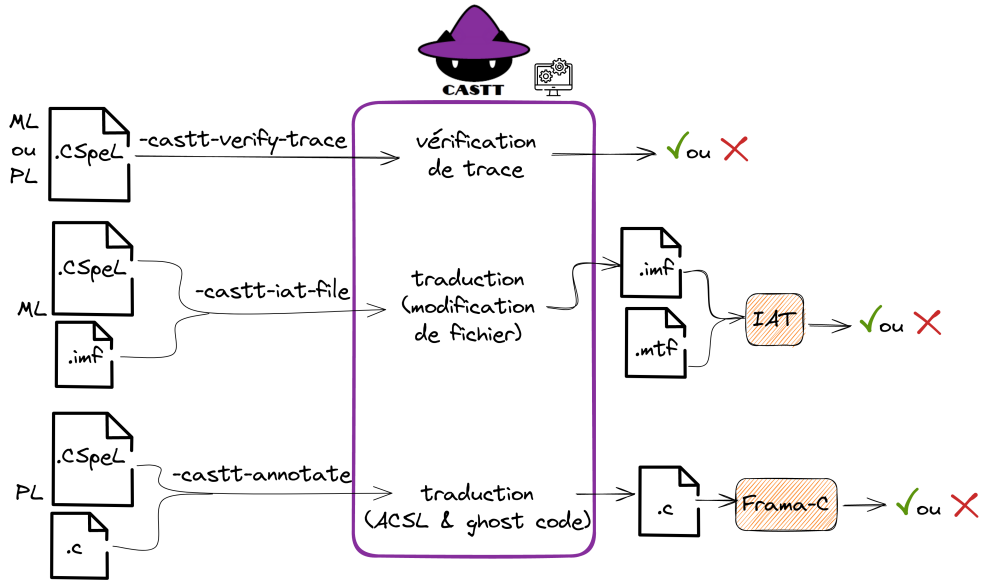


Figure 9.2 – Représentation du fonctionnement de CASTT.

Pour vérifier directement le système sans passer par une spécification de trace d'exécution, l'utilisateur a deux possibilités selon le niveau d'abstraction du système à vérifier. Au niveau Modèle Formel, l'utilisateur peut fournir un fichier `.imf` en plus de la spécification du contexte en CSpeL. La définition de ce fichier est détaillé en section 7.1.2. CASTT traduit alors la spécification CSpeL et modifie le fichier `.imf` en conséquence. L'outil IAT peut ensuite être utilisé pour analyser ce nouveau fichier en suivant son processus de vérification habituel. Ce mécanisme de traduction est détaillé en section 9.3.1.

Au niveau Programme, l'utilisateur peut fournir un fichier `.c` en plus de la spécification du contexte en CSpeL. CASTT traduit alors la spécification CSpeL en annotations ACSL (language de spécification pour le C [20]) et modifie le fichier `.c` en conséquence. Ensuite, Frama-C peut être utilisé pour vérifier ce nouveau fichier en suivant son processus de vérification habituel. Ce mécanisme de traduction est détaillé en section 9.4.

9.2 . Analyse de trace en CSpeL

Cette section présente le mécanisme d'analyse de trace implémenté dans CASTT. Pour cela, je présente d'abord son principe de fonctionnement, puis son implémentation, et finalement les expérimentations que j'ai réalisées pour évaluer cette fonctionnalité.

9.2.1 . Principe

L'analyse de traces spécifiées en CSpeL réalisée par CASTT correspond à de la vérification hors-ligne (*offline runtime verification*) [58]. Ce type de vérification permet de vérifier des propriétés sur des exécutions complètes du système (par exemple les traces ou les logs).

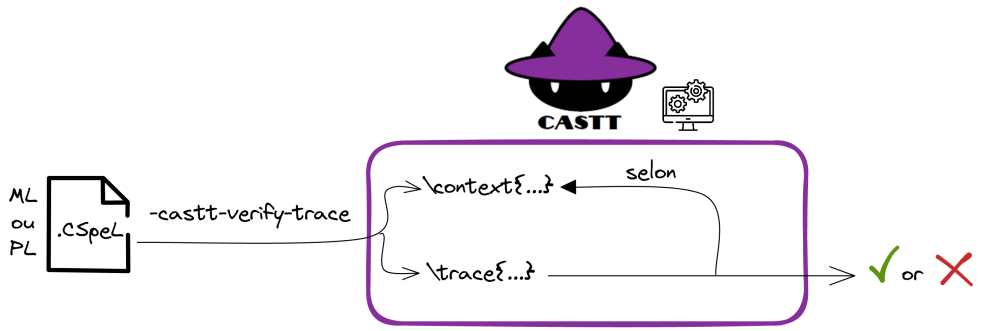


Figure 9.3 – Utilisation de CASTT pour vérifier des traces.

Comme le montre la figure9.3, CASTT peut vérifier qu'une trace représentant l'exécution d'un système au niveau Modèle ou Programme, satisfait les propriétés définies dans la section 8.1.3 (les propriétés de *conformité aux finalités consenties* et de *conformité à la nécessité des données*), selon un certain contexte spécifié (la trace et le contexte étant décrits en CSpeL). Actuellement, les traces sont écrites manuellement, mais elles pourraient être générées automatiquement à partir d'un modèle simulé ou d'exécutions d'un programme concret.

9.2.2 . Implémentation

L'algorithme 1 effectue une analyse d'une trace afin de vérifier si elle respecte les propriétés définies en section 8.1.3. Cet algorithme est implémenté en OCaml [157] dans CASTT. Pour chaque évènement de la trace, il vérifie si les données traitées font partie de l'ensemble des données personnelles. Si cela est le cas, l'algorithme vérifie si la personne concernée a préalablement consenti à l'une des finalités associées au processus : la trace n'est pas valide si un tel accord n'a pas été obtenu et l'algorithme retourne alors faux. Ensuite, l'algorithme vérifie si cette donnée personnelle est nécessaire au traitement, la trace ne peut être valide que si cela est le cas. L'évaluation se poursuit pour tous les évènements de la trace. Si au moins un des évènements n'est pas conforme, alors la trace est invalide. Par souci d'exhaustivité, l'algorithme vérifie par défaut la totalité de la trace, mais une option peut être donnée à CASTT (`-castt-verify-stop-early`) pour arrêter la vérification dès qu'un évènement est invalide. Dans l'algorithme, la notation `(process, data)` ← `trace.current event`, permet de représenter que l'on récupère les informations de l'évènement courant, c'est-à-dire quel processus traite

quelle donnée. La notation `trace ← trace.next()`, permet de représenter le fait que l'on avance dans la trace jusqu'à l'évènement suivant (jusqu'à ce que la trace soit vide).

Algorithme 1 : Évaluation de trace

Input : Un contexte et une trace
Output : Résultat de l'évalutation

```
1  is_valid = True ; // une trace vide est valide
2  while trace is not void do
3      (process, data) ← trace.current event;
4      trace ← trace.next();
5      if data ∈ context.personal data then
6          /* Conformité de finalité */
7          is_consented = False;
8          foreach purpose ∈ context.hasPurposes(process) do
9              if context.isGranted(data, purpose) then
10                 is_consented = True ;
11         /* Conformité de nécessité des données */
12         is_necessary = False ;
13         if data ∈ context.needData(process) then
14             is_necessary = True ;
15         is_valid = is_valid && is_consented && is_necessary
16 return is_valid;
```

Je vais maintenant étudier la correction et la complexité de l'algorithme en détaillant les preuves correspondantes.

Théorème 1: Correction de l'algorithme 1

Pour tout contexte $\mathcal{C} = (S, D, P, \gamma, \pi, \nu)$ et toute trace T, l'algorithme retourne en sortie la variable `is_valid` à vrai (*true*) si et seulement si la trace est conforme selon le contexte défini.

Preuve 1: du théorème 1

1. Prouvons dans un premier temps que si l'algorithme retourne la variable `is_valid` à vrai alors la trace est conforme. Si l'algorithme retourne la variable `is_valid` à vrai alors :
   (a) soit l'algorithme n'est pas passé dans la boucle (ligne 2). Dans ce cas la trace T est vide ($T = \epsilon$), ce qui correspond à la définition de la règle de la trace vide de la figure 8.2. Donc, la trace est conforme.
   (b) soit l'algorithme est passé dans la boucle (ligne 2). Dans ce

cas la trace T n'est pas vide ($T = Handle(s, d); T'$, avec $s =$ `process`, $d =$ `data` et $T' =$ `trace.next()`). Alors :

i. soit la condition $data \in context.personaldata$ (ligne 5) est fausse (ce qui correspond à $d \notin D$), alors la valeur de `is_valid` est inchangée. Si elle était vraie, elle le reste. Si elle était fausse, alors cela était dû à un évènement précédent. Cela reviendrait à effectuer la même instruction que la ligne 15 :
`is_valid = is_valid && is_consented && is_necessary`, avec `is_consented` valant vrai, de même pour `is_necessary` (cela n'est pas indiqué dans le code pour l'optimiser). Cela correspond aux premières règles des définitions 10 et 11 (celles pour les données non personnelles).

ii. soit la condition (ligne 5) est vraie (ce qui correspond à $d \in D$). Dans ce cas, pour que `is_valid` soit vrai, il faut que `is_necessary` et `is_consented` soient vrais à tous les tours de boucle (d'après la ligne 15). Cela correspond à la définition de la règle de la concaténation de trace de la figure 8.2. Cela signifie que tous les évènements sont *conformes à la nécessité des données* et *conformes aux finalités consenties*.

A. `is_necessary` est vrai si et seulement si la condition $data \in context.needData(process)$ (ligne 13) est vraie. Cela correspond à $d \in \nu(p)$, ce qui correspond à la seconde règle de la définition 11. Donc, l'évènement est *conforme à la nécessité des données*.

B. `is_consented` est vrai si et seulement si la condition `context.isGranted(data, purpose)` (ligne 9) est vraie. Cela est évalué pour toutes les finalités associées au processus (`foreach purpose` $\in$ `context.hasPurposes(process)` ligne 8). Cela signifie qu'il existe au moins une finalité associée au processus qui est consentie pour le traitement de cette donnée. Cela correspond à $\exists p \in \pi(s).\gamma(d, p) = \top$ et donc à la seconde règle de la définition 10. Donc l'évènement est *conforme aux finalités consenties*.

Donc si l'algorithme retourne vrai, alors la trace est conforme.

2. Prouvons dans un second temps que si la trace T est conforme alors l'algorithme retourne la variable `is_valid` à vrai :

(a) soit la trace est vide ($T = \epsilon$) (elle est conforme selon la dé-

finition de la règle de la trace vide de la figure 8.2). Alors la condition *trace is not void* est fausse (ligne 2), donc l'algorithme ne rentre pas dans la boucle. L'algorithme retourne alors la variable `is_valid` (ligne 16). Celle-ci ayant été initialisée à vrai (ligne 1) et n'ayant pas été modifiée, l'algorithme retourne donc vrai.

(b) soit la trace n'est vide pas ($T = Handle(s,d);T'$), alors, comme elle est conforme, tous ses évènements sont conformes (selon la définition de la règle de la concaténation de traces de la figure 8.2). Ainsi, la condition *trace is not void* est vraie (ligne 2), donc l'algorithme rentre dans la boucle (avec s = `process`, d = `data` et T'= `trace.next()`).

i. soit la $d \notin D$ (ce qui correspond aux premières règles des définitions 10 et 11, pour les données non personnelles). Alors, la condition *data* $\in$ *context.personal data* (ligne 5) est fausse. Dans ce cas, la valeur de `is_valid` est inchangée. Si elle était vraie (comme lors de l'initialisation), elle le reste. Si elle était fausse, alors cela était dû à un évènement précédent.

ii. soit $d \in D$, alors l'évènement ($Handle(s,d)$) est *conforme aux finalités consenties* et *conforme à la nécessité des données* (car la trace est conforme).

A. D'après la seconde règle de la définition 10, il existe une finalité associée au processus qui est consentie pour le traitement de cette donnée ($\exists p \in \pi(s).\gamma(d,p) = \top$), donc la condition `context.isGranted(data, purpose)` (ligne 9) est vraie au moins pour une des finalités associés au processus. Donc, en executant la boucle qui parcourt ces finalités (ligne 8), au moins une fois la condition est vraie. Ainsi, la variable `is_consented` a été mise à vrai (au moins une fois).

B. D'après la seconde règle de la définition 11, la donnée traitée est nécessaire au processus ($d \in \nu(s)$), donc la condition $data \in context.needData(process)$ (ligne 13) est vraie. Donc `is_necessary` est mise à vrai.

Donc `is_valid` est mise à vrai (ligne 15). Cela est le cas pour tous les évènements de la trace (qui traitent des données personnelles, $d \in D$).

Donc, après avoir parcouru tous les évènements de la trace, `is_valid` vaut vrai.

Donc si la trace est conforme, alors l'algorithme retourne vrai.

Donc, en sortie, l'algorithme retourne la variable `is_valid` à vrai si et seulement si la trace est conforme selon le contexte défini.

Théorème 2: Complexité théorique en temps au pire cas de l'algorithme 1

La complexité théorique de l'algorithme est de $O(t \times f)$ avec f le plus grand nombre de finalités de traitements associées aux processus dans le context $\mathcal{C}$ d'entrée et t la taille de T.

Preuve 2: du théorème 2

L'algorithme a deux boucles imbriquées, donc la complexité dépendra de la complexité des deux boucles. La première a une complexité selon la taille de la trace (car l'algorithme parcourt tous les évènements de la trace), notée t, et l'autre a une complexité selon le nombre de finalités associées au processus (car l'algorithme parcourt toutes les finalités associées au processus), noté f. Donc la complexité théorique en temps au pire de l'algorithme est de $O(t \times f)$.

Une optimisation pourrait être ajoutée pour arrêter l'algorithme dès qu'un évènement n'est pas conforme, mais cela ne changerait pas la complexité en temps au pire.

9.2.3 . Éxperimentations

Questions de Recherche (RQ) : l'analyse de traces de CASTT est évaluée selon les questions de recherche suivantes :

RQ1 *Est-ce que CASTT peut vérifier une propriété de consentement sur une trace au niveau modèle ?*

RQ2 *Est-ce que CASTT peut vérifier une propriété de consentement sur une trace au niveau programme ?*

RQ3 *Est-ce que CASTT peut détecter des traces invalides, c'est à dire des traces qui ne respectent pas la conformité aux finalités consenties ou la conformité à la nécessité des données ?*

RQ4 *Est-ce que CASTT est utilisable sur des grandes traces* ?

Cas d'utilisation : Pour les expérimentations, j'ai utilisé l'exemple GDM_{Spl} (aux niveaux Modèle Graphique, Modèle Formel et Programme) détaillé en section 7.3.1, celui GDM_{Cpx} (aux niveaux Modèle Graphique et Modèle Formel) détaillé en section 7.3.2, et l'exemple AL_{Spl} (aux niveaux Modèle Graphique, Modèle Formel et Programme) détaillé en section 7.4.1. L'exemple GDM_{Cpx} n'as pas d'expérimentations au niveau programme, car comme précisé lors de sa spécification, cet exemple n'a pas d'implémentation définie.

Pour ces différents exemples, j'ai utilisé des scénarios représentant différentes exécutions des systèmes. Ces scénarios sont détaillés dans le tableau 9.1 pour l'exemple GDM$_{Spl}$, dans le tableau A.5 de l'annexe A.7.1 pour GDM$_{Cpx}$, et dans les tableaux A.6, A.7 et A.8 de l'annexe A.7.2 pour AL$_{Spl}$. Ces scénarios contiennent une spécification de contexte et une trace d'exécution écrites en CSpeL. Ces tableaux présentent, pour chaque exemple, les spécifications de contexte ainsi que la trace T_i utilisée selon le niveau d'abstraction (LVL), puis le nombre total de scénarios utilisés pour l'exemple. Des traces valides et invalides sont spécifiées. Les spécifications de contexte sont communes selon le niveau d'abstraction (ex : un même contexte pour toutes les traces du niveau Modèle Graphique de l'exemple GDM$_{Spl}$). Pour des raisons de place, les traces utilisées dans les scénarios pour l'exemple AL$_{Spl}$ sont détaillées dans plusieurs tableaux à chaque niveau d'abstraction considéré. Dans le tableau A.6 pour le niveau Modèle Graphique et Modèle Formel, dans les tableaux A.7 et A.8 pour le niveau Programme. Le tableau 9.1 présente la spécification du contexte aux niveaux Modèle Graphique, Modèle Formel et Programme, selon la définition de l'exemple GDM$_{Spl}$ en section 7.3.1. Les traces spécifiées permettent d'évaluer l'approche sur différents cas possibles de traces, représentant différents scénarios d'exécution : à un évènement, avec des séquences d'évènements, avec des évènements tous conformes, avec des évènements non conformes (à différentes places dans la trace), selon les deux propriétés. Ainsi,

— Au niveau Modèle Graphique :
  - la trace $T1$ correspond à l'exécution du processus $P1$, ce qui est conforme ;
  - la trace $T2$ correspond à l'exécution du processus $P2$, ce qui est non conforme (selon la propriété de *conformité aux finalités consenties*) ;
  - la trace $T3$ correspond à l'exécution de $P1$, suivi de l'exécution de $P2$ ce qui est non conforme (selon la propriété de *conformité aux finalités consenties*) ;
  - la trace $T4$ correspond à l'exécution de $P2$, suivi de l'exécution de $P3$ ce qui est non conforme (selon la propriété de *conformité aux finalités consenties*) ;
  - la trace T correspond à l'exécution de $P2$, suivi de l'exécution de $P3$ ce qui est non conforme (selon la propriété de *conformité aux finalités consenties*) ;

— Au niveau Modèle Formel :
  - la trace $T1$ correspond à l'exécution des processus raffinant $P1$, qui est donc conforme ;
  - la trace $T2$ correspond à l'exécution des processus raffinant $P2$, donc non conforme ;

— Au niveau Programme :

- la trace $T1$ correspond à l'exécution des processus raffinant $P1$, qui est donc conforme ;
- la trace $T2$ correspond à l'exécution des processus raffinant $P2$, donc non conforme ;
- la trace $T3$ correspond à deux exécutions (en séquence) des processus raffinant $P1$, ce qui est conforme ;
- la trace $T4$ correspond à plusieurs exécutions (avec interlacement) des processus raffinant $P1$, ce qui est conforme ;
- la trace $T5$ correspond à l'exécution des processus raffinant $P1$ mais qui traitent des données personnelles non nécessaires, ce qui est non conforme (selon la propriété de *conformité à la nécessité des données*) ;
- la trace $T6$ correspond à l'exécution des processus raffinant $P1$ suivi de ceux raffinant $P2$ mais qui traitent des données personnelles non nécessaires, ce qui est non conforme (selon la propriété de *conformité à la nécessité des données*) ;

Table 9.1 – Tableau des scénarios pour l'exemple GDM$_{Spl}$.

LVL	\context	\trace	
ML	\process{P1,P2}, \personalData{EPR}, \purposes{Treatment, Research}, \isGranted{(EPR : Treatment)}, \hasPurposes{(P1 :{Treatment}), (P2 :{Research})}, \needData{(P1 :{EPR}), (P2 :{EPR})}	T1	\handle(P1, EPR);VOID
		T2	\handle(P2, EPR);VOID
		T3	\handle(P1, EPR); \handle(P2, EPR);VOID
		T4	\handle(P2, EPR); \handle(P1, EPR);VOID
ML formel	\process{T11,T12,T13,T22}, \personalData{EPR}, \purposes{Treatment, Research}, \isGranted{(EPR : Treatment)}, \hasPurposes{(T11 :{Treatment}), (T12 :{Treatment}), (T13 :{Treatment}), (T22 :{Research})}, \needData{(T11 :{EPR}),(T12 :{EPR}), (T13 :{EPR}),(T22 :{EPR})}	T1	\handle(T11, EPR); \handle(T12, EPR); \handle(T13, EPR); VOID
		T2	\handle(T22, EPR); VOID
PL	\process{makePrescr, getData, computeStats}, \personalData{Patient, Data}, \purposes{Treatment, Research}, \isGranted{(Patient : Treatment)}, \hasPurposes{ (makePrescr :{Treatment}), (getData :{Treatment,Research}), (computeStats :{Research})}, \needData{(makePrescr :{Patient}), (getData :{Patient, Data}), (computeStats :{Data})}, \init{main}	T1	\handle(makePrescr, Patient); \handle(makePrescr, Dos); VOID
		T2	\handle(getData, Patient); \handle(getData, Data); \handle(computeStats, Data); VOID
		T3	\handle(makePrescr, Patient); \handle(makePrescr, Dos); \handle(makePrescr, Patient); \handle(makePrescr, Dos); VOID
		T4	\handle(makePrescr, Patient); \handle(makePrescr, Dos); \handle(makePrescr, Patient); \handle(makePrescr, Dos); \handle(makePrescr, Dos); \handle(makePrescr, Patient); VOID
		T5	\handle(makePrescr, Patient); \handle(makePrescr, Data); VOID
		T6	\handle(makePrescr, Patient); \handle(makePrescr, Dos); \handle(getData, Patient); \handle(getData, Data); \handle(computeStats, Data); \handle(computeStats, Patient); VOID
Total	12		

Résultats : Pour répondre à aux questions de recherche **RQ1** à **RQ4**, j'ai exécuté CASTT sur chacun des fichiers de test, contenant les scénarios précédents, à l'aide de la commande suivante, dans laquelle `testFile.cspel` correspond au nom du fichier de test :

```
frama-c -castt-verify-trace -castt-consent-file <testFile.cspel>
```

Table 9.2 – Résultats expérimentaux pour l'analyse de trace de CASTT.

Exemple	LVL*	NbTests	Taille des traces	Traces valid détectées	Traces invalide détectées
GDM_{Spl}	ML	6	1 à 3 évènements	✓	✓
	PL	6	2 à 6 évènements	✓	✓
GDM_{Cpx}	ML	10	1 à 20 évènements	✓	✓
AL_{Spl}	ML	9	1 à 5 évènements	✓	✓
	PL	9	5 à 24 évènements	✓	✓

* Niveau

Chaque expérimentation fournit instantanément (c'est à dire, en moins d'une seconde) son résultat. Ceux-ci sont résumés dans le tableau 9.2. Cette table présente pour chacun des exemples (GDM_{Spl}, GDM_{Cpx} et AL_{Spl}), et pour chaque niveau d'abstraction (aux niveaux Modèles et Programme), le nombre de fichiers CSpeL de tests analysés, le nombre minimum et maximum d'évènements dans les traces, et si la validité des traces a été correctement détectée. Ces expérimentations contiennent différentes traces valides et différentes traces invalides. Ces traces sont définies, manuellement, pour être valides ou invalides, le but étant de vérifier si CASTT les détecte correctement.

Ces résultats démontrent que CASTT fournit toujours le verdict attendu sur nos exemples, à la fois aux niveaux Modèles et au niveau Programme. Cela permet de répondre positivement aux questions de recherche **RQ1**, **RQ2**, et **RQ3**.

Pour répondre à la question **RQ4**, j'ai exécuté un script mesurant le temps d'exécution. Ce script exécute la commande précédente sur différents fichiers CSpeL. Ces fichiers correspondent à un seul contexte, celui de l'exemple GDM_{Cpx} présenté précédemment, mais avec des tailles de trace variant de 10 à 1 000 000 évènements. Ces traces sont générées par la fonctionnalité annexe de génération de trace, détaillée en section 9.5.1. Ce script exécute la vérification 10 fois pour une même taille de trace et calcule la moyenne du temps pris pour ces vérifications. Les résultats sont présentés sur la figure 9.4. Ces résultats, visibles sur la courbe de la figure, permet de vérifier que l'algorithme de vérification de trace est linéaire en temps selon la taille de la trace, lorsque le nombre de « purpose » est négligeable par rapport à la taille de la trace. Cela montre également que des grandes traces, i.e. avec 1 000 000 évènements, sont vérifiées par CASTT en moins de 4 secondes. Je peux donc répondre positivement à la question de recherche **RQ4**.

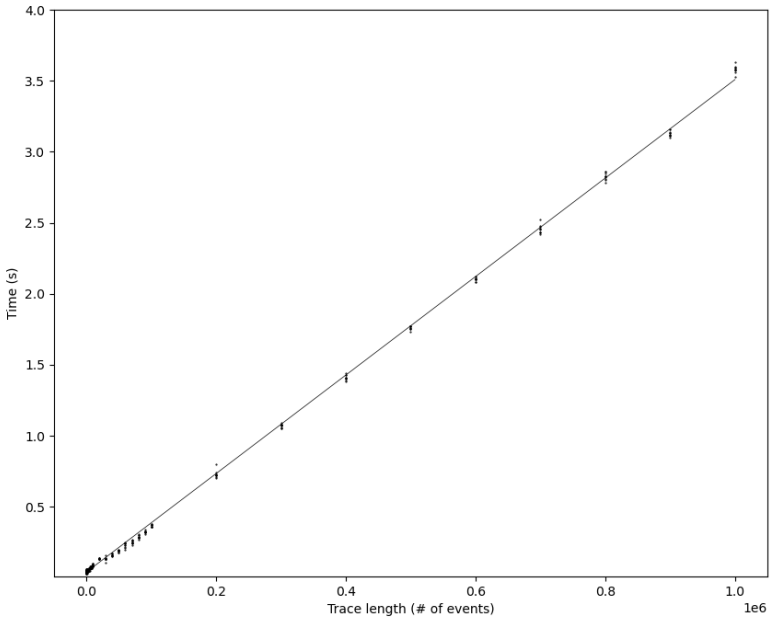


Figure 9.4 – Durée de la vérification de trace selon la taille de la trace analysée [41].

9.3 . Traduction pour un outil Modèle

Cette section présente la traduction de CSpeL dans un formalisme utilisable par IAT, un outil dédié à l'analyse de système au niveau Modèle formel. Actuellement seule la traduction relative à la vérification de la propriété *conformité aux finalités consenties* est implémentée.

Dans un premier temps, je présente l'outil IAT, puis le principe du mécanisme de traduction, son implémentation et enfin les expérimentations que j'ai réalisées pour évaluer ce mécanisme.

9.3.1 . IAT

IAT [109] est un outil permettant l'analyse de traces et de multi-traces [39] pour l'exécution de systèmes distribués, fondé sur des modèles d'interaction. Pour aider à l'analyse, cet outil propose plusieurs fonctionnalités : modélisation, représentation graphique, manipulation de modèles ou bien encore exploration de graphes. Les modèles utilisés pour l'analyse sont exprimés à l'aide d'une spécification formelle représentant des modèles d'interaction. Ce type de spécification a déja été détaillé dans la section 7.1.2, tandis que son application sur nos exemples a été présentée en section 7.3.1 et section 7.4.1.

9.3.2 . Principe

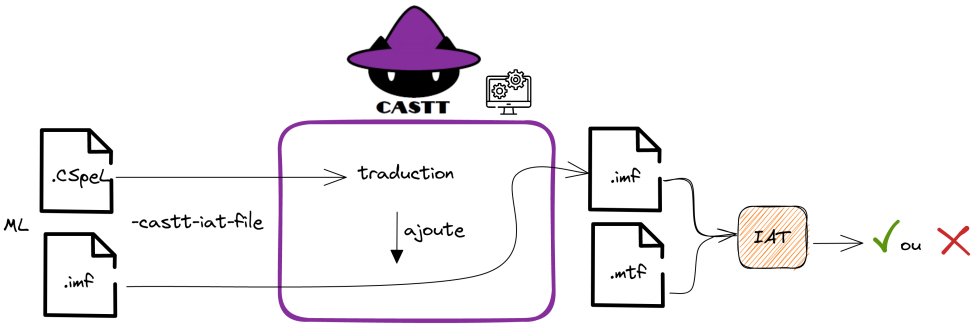


Figure 9.5 – Utilisation du mécanisme de traduction au niveau Modèle de CASTT.

Comme illustré sur la figure 9.5, le principe de cette traduction est de prendre en entrée un fichier écrit en CSpeL, contenant une spécification du système dans le contexte de vie privée associé, et de traduire ce fichier en une entrée utilisable pour IAT. Plusieurs fichiers distincts sont nécessaires pour le fonctionnement de IAT :

— un fichier .imf qui contient les informations relatives au modèle du système et aux conditions de l'analyse ;

— un fichier .mtf qui contient une instanciation d'échanges de messages dans le système, constituant une trace d'exécution.

Ainsi, notre traduction va modifier seulement le fichier .imf, pour pouvoir modifier les informations liées au modèle afin d'ajouter les notions liées à la vie privée. Le fichier .mtf, contenant la trace d'exécution, n'est pas modifié par cette traduction mais sera utilisé pour la vérification.

```
{
    [S1] S1!START;
    [T11] T11?START.T11!collectSymptoms(4);
    [T12] T12?collectSymptoms(4).T12!makeDiagnosis(4);
    [T13] T13?makeDiagnosis(4).T13!prescribeTreatment(4);
    [T14] T14?prescribeTreatment(4).T14!dischargePatient;
    [E1] E1?dischargePatient.E1!END
}
```

Figure 9.6 – Instanciation pour IAT de la prescription du traitement médical de GDM_{Spl}.

Par exemple, le fichier .mtf correspondant à l'instanciation de l'exécution du processus de prescription du traitement médical correspond au contenu indiqué sur la figure 9.6.

Ce fichier contient un ensemble d'envois (ex : `S1!START`) et de réceptions (ex : `T11?START`) par ligne de vie. La valeur de la donnée envoyée est mise à 4, car cela correspond à l'initialisation correspondante dans le fichier .imf (mais cette valeur en elle même est arbitraire dans cet exemple). Les détails de la traduction de CSpeL vers .imf sont expliqués dans la section suivante.

9.3.3 . Implémentation

L'implémentation de la traduction de la spécification écrite en CSpeL modifiant un fichier `.imf` de IAT est accessible dans CASTT à l'aide de la commande `-castt-iat-file`.

Table 9.3 – Génération de la traduction pour IAT.

CSpeL	Code généré

\personalData
& \purposes
	@variable{
	...
	epr_treatment : Bool;(1)
	epr_research : Bool;(1)
	}

\isGranted
& \personalData
& \purposes
	@init{
	...
	£epr_treatment = ⊤(1)
	£epr_research = ⊥(1)
	}

\process
& \hasPurposes
& \needData
	@model{
	...
	T11[£epr_treatment](1) −− collectSymptoms(£epr) → T12
	...
	}

(1) éléments générés

La table 9.3 présente les correspondances entre les différents éléments liés à CSpeL et les éléments générés dans le fichier `.imf`. La spécification des données personnelles (\personalData) et la spécification des finalités (\purposes) sont utilisées pour ajouter dans le bloc des variables du fichier (`@variable`), des variables booléennes correspondant à l'association entre donnée personnelle et finalité (par exemple, `epr_treatment : Bool;`). Ces variables sont ensuite initialisées dans le bloc d'initialisation du fichier (`@init`), selon le consentement (\isGranted) spécifié par l'utilisateur (par exemple, `£epr_treatment = ⊤`). Pour rappel, la présence du couple (donnée personnelle, finalité) dans la spécification du consentement (\isGranted) signifie que, pour cette finalité, un consentement a été fourni pour le traitement de ce type de donnée personnelle. Par défaut, le consentement est considéré comme n'étant pas accordé.

Pour chaque processus (par exemple, `T11`) spécifié comme traitant des données personnelles (`\process`), une garde (par exemple, [£epr_treatment]) est ajoutée dans l'échange de message (par exemple, `T11[£epr_treatment] -- collectSymptoms(£epr) → T12`), selon les finalités de traitement et les données personnelles nécessaires à ce processus. Une garde permet de mettre des conditions sur les envois de messages. Ces informations sont définies par l'utilisateur dans le fichier CSpeL à l'aide de `\hasPurposes` et `\needData`. Par la suite, l'outil IAT vérifie si l'échange des messages dans les fichiers de trace d'exécution respecte la garde présente dans la spécification de l'échange du message. Comme indiqué précédemment, cette traduction ne permet actuellement de vérifier que la propriété de *conformité aux finalités consenties*. La traduction pour vérifier la propriété de *conformité à la nécessité des données* n'est pas encore implémentée, elle est laissée pour travaux futurs (détaillés en section 10.3).

9.3.4 . Expérimentations

Questions de Recherche (RQ) Le mécanisme de traduction de CASTT pour un outil au niveau Modèle Formel est évalué selon les questions de recherche suivantes :

RQ5 *CASTT peut-il traduire un fichier CSpeL pour un outil au niveau Modèle Formel ?*

RQ6 *CASTT peut-il être utilisé pour vérifier des systèmes au niveau Modèle Formel ?*

RQ7 *CASTT peut-il être utilisé pour détecter les traces invalides concernant la conformité aux finalités consenties ?*

RQ8 *CASTT peut-il être utilisé pour détecter les traces invalides concernant la conformité à la nécessité des données ?*

RQ9 *Est-ce que CASTT est utilisable sur de grands fichiers* ?

Cas d'utilisation

Pour les expérimentations, j'ai utilisé l'exemple GDM_{Spl} (détaillé en section 7.3.1), et celui AL_{Spl} (détaillé en section 7.4.1). Pour chacun de ces exemples, je n'ai sélectionné que le niveau Modèle Formel (figures 7.20 et 7.28).

Table 9.4 – Scénarios pour la traduction IAT - exemple GDM_{Spl}.

Fichier CSpeL	Fichiers .mtf	
\process{T11,T12,T13,T22, T23}, \personalData{epr}, \purposes{treatment, research}, \isGranted{(epr : treatment)}, \hasPurposes{(T11 :{treatment}), (T12 :{treatment}), (T13 :{treatment}), (T22 :{research}), (T23 :{research})}, \needData{(T11 :{epr}), (T12 :{epr})}	T1	[S1] S1 !START; [T11] T11 ?START.T11 !collectSymptoms(4); [T12] T12 ?collectSymptoms(4).T12 !makeDiagnosis(4); [T13] T13 ?makeDiagnosis(4).T13 !prescribeTreatment(4); [T14] T14 ?prescribeTreatment(4).T14 !dischargePatient; [E1] E1 ?dischargePatient.E1 !END
	T2	[S2] S2 !START; [T21] T21 ?START.T21 !prepareTrial; [T22] T22 ?prepareTrial(4).T22 !performTrial(4); [T23] T23 ?performTrial(4).T23 !analyzeResults(4); [E2] E2 ?analyzeResults(4).E2 !END

Le tableau 9.4 présente les scénarios de test, *i.e.* le contexte CSpeL et l'exécution du système à vérifier (fichier `.mtf`) pour l'exemple GDM_{Spl}. Ces scénarios représentent les exécutions des différents processus définis dans l'exemple, celui pour réaliser le traitement médical (T1) et celui pour réaliser l'essai clinique (T2).

Table 9.5 – Scénarios pour la traduction IAT - exemple AL_{Spl}.

Fichier CSpeL	Fichiers .mtf	
\process{T11, T12, T13, T21, T31, T32}, \personalData{CC,Envoi}, \purposes{Administratif, Marketing}, \isGranted{(CC : Administratif)}, \hasPurposes{(T11 :{Administratif}), (T12 :{Administratif}), (T13 :{Administratif}), (T21 :{Administratif, Marketing}), (T31 :{Marketing}), (T32 :{Marketing})}, \needData{(T11 :{CC,Envoi}), (T12 :{CC})} (T13 :{CC})} (T21 :{CC, Envoi})} (T31 :{CC})} (T32 :{Envoi})}	T1	[S1] S1 !START; [T11] T11 ?START.T11 !acheter(4); [T12] T12 ?acheter(4).T12 !enregistrerHistorique(4); [T13] T13 ?enregistrerHistorique(4).T13 !afficherHistorique(4); [E1] E1 ?afficherHistorique(4).E1 !END
	T2	[S2] S2 !START; [T21] T21 ?START.T21 !accederHistorique(4); [T13] T13 ?accederHistorique(4).T13 !afficherHistorique(4); [E2] E2 ?afficherHistorique(4).E2 !END
	T3	[S3] S3 !START; [T21] T21 ?START.T21 !accederHistorique(4); [T31] T31 ?accederHistorique(4).T31 !choisirPublicite(4); [T32] T32 ?choisirPublicite(4).T32 !envoyerPublicite(4); [E3] E3 ?envoyerPublicite(4).E3 !END

Le tableau 9.5 présente les scénarios de tests, pour l'exemple AL_{Spl}. Ces scénarios représentent les exécutions des différents processus définis dans l'exemple, lorsque l'utilisateur réalise des achats (T1), lorsque l'utilisateur consulte son historique (T2), et enfin celui pour l'envoi de publicités personnalisées (T3).

Les fichiers `.imf` complets sont disponibles en annexe A.1.2 pour l'exemple GDM_{Spl} et en annexe A.3.3 pour l'exemple AL_{Spl}.

Résultats

Pour répondre à aux questions de recherche **RQ5** à **RQ9**, j'ai d'abord exécuté CASTT sur chacun de ces exemples à l'aide de la commande `frama-c <options>`, avec les options suivantes :

```
-castt-consent-file <file.cspel> -castt-iat-file <file.imf>
```

CASTT étant un plug-in de Frama-C, la commande `frama-c` est nécessaire pour l'exécuter, même si IAT est un outil indépendant de Frama-C. Le fichier `<file.cspel>` contient la spécification écrite en CSpeL, tandis que le fichier `<file.imf>` contient le système défini pour IAT. CASTT a généré, pour chaque test, un nouveau fichier correspondant au fichier `.imf` donné en entrée, avec les modifications nécessaires pour vérifier la propriété *conformité aux finalités consenties*. Ces fichiers générés se trouvent en annexe A.1.3 pour l'exemple GDM_{Spl} et en annexe A.3.4 pour l'exemple AL_{Spl}. La génération de ces fichiers suit le principe présenté précédemment.

J'ai ensuite exécuté IAT à l'aide de la commande suivante :

```
./iat analyze <generatedFile.imf> <file.mtf>
```

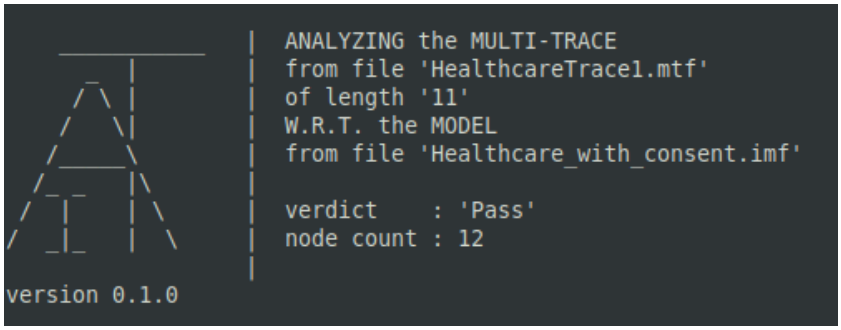
Le fichier `<generatedFile.imf>` est le fichier généré par CASTT, et le fichier `<file.mtf>` est l'exécution à vérifier dont les contenus sont décrits dans les tables précédentes.

IAT fournit alors un verdict sur la validité de l'exécution. La figure 9.7 présente des exemples de verdict de IAT pour des exécutions valides et non valides.

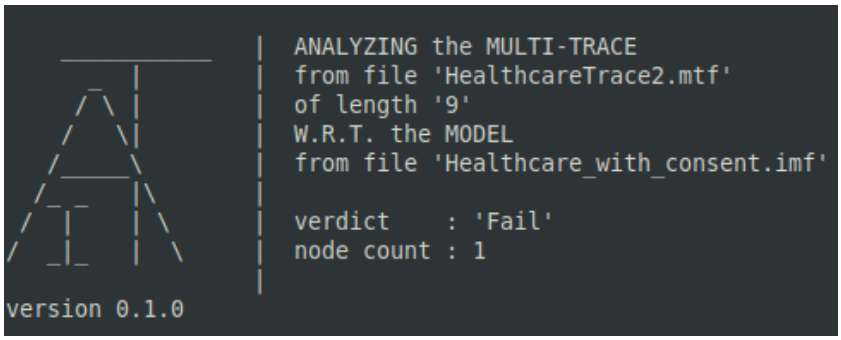
Exemple	Spécification traduite	Nb exécutions	Taille de exécution	Exécutions valides détectées	Exécutions invalides détectées
GDM_{Spl}	✓	2	9 à 11 événements	✓	✓
AL_{Spl}	✓	3	7 à 9 événements	✓	✓

Table 9.6 – Résultats expérimentaux pour le mécanisme de traduction pour IAT de CASTT.

Chaque expérimentation fournit presque instantanément (*i.e* en moins de 2 secondes) son résultat. Ceux-ci sont résumés dans le tableau 9.6. Cette table présente, pour chacun des exemples (GDM_{Spl} et AL_{Spl}), si la traduction de la spécification CSpeL a pu être réalisée, le nombre de scénarios analysés (fichiers `.mtf` contenant les exécutions), le nombre minimum et maximum d'évènements dans les exécutions (envoi ou réception de message), et si la validité a été correctement détectée. Ces expérimentations contiennent différentes exécutions valides et invalides. Ces



Exécution valide



Exécution non valide

Figure 9.7 – Verdicts IAT pour les cas d'une exécution valide et d'une exécution non valide.

exécutions sont définies manuellement de telle sorte qu'elles soient valides ou invalides, selon la définition de l'exemple (le but étant de pouvoir vérifier si elles sont correctement détectées).

Bien que préliminaires, ces résultats démontrent que CASTT réussit à traduire une spécification écrite en CSpeL en modifiant un fichier utilisable par IAT, de telle sorte que les exécutions valides et invalides soient correctement détectées. Je peux donc répondre positivement aux questions de recherche **RQ5**, **RQ6**, et **RQ7**. Par contre la traduction de la propriété de *conformité à la nécessité des données* n'étant pas implémentée actuellement, la question de recherche **RQ8** est laissée en travaux futurs (présentés en section 10.3.9). De même, ces résultats étant préliminaires, le mécanisme de traduction pour IAT n'a pas été évalué pour de grands fichiers. La question de recherche **RQ9** est donc également laissée en travaux futurs.


Ce mécanisme de traduction correspondait à une perspective de l'article [41].

9.4 . Traduction pour un outil PL

Cette section présente la traduction de CSpeL dans un formalisme utilisable par Frama-C, un outil dédié à l'analyse de système au niveau Programme.

Dans un premier temps, je présente Frama-C, notamment les différents plug-ins de vérification que j'ai utilisés, puis le principe du mécanisme de traduction, son implémentation et enfin les expérimentations que j'ai réalisées pour évaluer ce mécanisme.

9.4.1 . Frama-C : WP/Eva/E-ACSL

Frama-C [19] est une plateforme open-source d'analyse de code C. Elle fournit plusieurs plug-ins pour l'analyse de code source C, étendu avec des annotations formelles écrites dans un langage de spécification dédié, nommé ACSL [20]. Ses trois principaux plug-ins de vérification sont WP [25], Eva [26] et E-ACSL [154].

WP s'appuie sur des méthodes déductives [68] pour prouver les propriétés ACSL grâce à des prouveurs associés, tels que Alt-Ergo [44]. Eva est un outil statique fondé sur l'interprétation abstraite [45] qui déclenche des alarmes pour tout comportement potentiellement indéfini et toute propriété ACSL invalide. E-ACSL est un vérificateur d'assertions à l'exécution [38] qui vérifie les propriétés ACSL pendant l'exécution de programmes concrets.

Comme expliqué dans les sections suivantes, j'ai utilisé ces différents plug-ins pour vérifier les propriétés de *conformité aux finalités consenties* et de *conformité à la nécessité des données* au niveau Programme.

9.4.2 . Principe

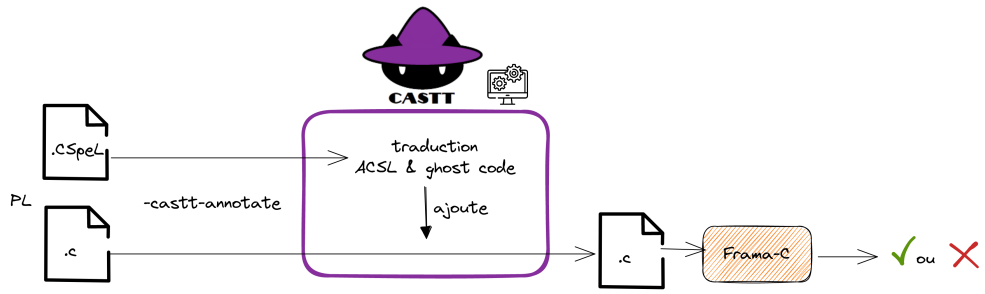


Figure 9.8 – Utilisation du mécanisme de traduction au niveau Programme de CASTT.

La figure 9.8 montre le principe de traduction de CASTT, pour l'utilisation de Frama-C, afin de vérifier les propriétés de *conformité aux finalités consenties* et de *conformité à la nécessité des données* (expliquées en section 8.1.3) sur un code source écrit en C. CASTT prend en entrée une spécification de contexte écrite en CSpeL et un code source écrit en C afin de générer un nouveau code source C étendu avec des annotations ACSL. Ensuite, n'importe quel analyseur Frama-C peut être utilisé pour vérifier les annotations ACSL incorporées dans ce code généré.

Table 9.7 – Génération du code.

Définition en CSpeL	Extrait de code généré
\personalData	enum _PersonalData {TRIALDATA = 0, PATIENT = 1} typedef enum _PersonalData PersonalData;
\purposes	enum _Purposes {RESEARCH = 0, TREATMENT = 1} typedef enum _Purposes Purposes;
\personalData & \purposes	/*@ ghost bool Consent[2][2];*/
\process & \personalData	/*@ ghost bool Need[3][2];*/
\personalData & \purposes & \isGranted & \init	/*@ ghost Consent[PATIENT][RESEARCH] = 0; */ /*@ ghost Consent[PATIENT][TREATMENT] = 1; */ /*@ ghost Consent[TRIALDATA][RESEARCH] = 0; */ /*@ ghost Consent[TRIALDATA][TREATMENT] = 0; */
\process & \personalData & \needData & \init	/*@ ghost Need[MAKEPRESCR][PATIENT] = 1; */ /*@ ghost Need[MAKEPRESCR][TRIALDATA] = 0; */ /*@ ghost Need[GETDATA][PATIENT] = 1; */ /*@ ghost Need[GETDATA][TRIALDATA] = 1; */
\personalData & \hasPurposes	/*@ assert Consent[PATIENT][TREATMENT] == 1; */
\personalData	/*@ assert Need[MAKEPRESCR][PATIENT] == 1; */

Vérifier toutes les annotations implique que les propriétés de conformité originales sont satisfaites. Dans la pratique, l'utilisateur peut utiliser n'importe lequel des plug-ins de vérification présentés précédemment, E-ACSL, Eva, ou WP, ou une combinaison de ceux-ci, afin de vérifier le code généré.

9.4.3 . Implémentation

Le tableau 9.7 présente le code généré à partir de la spécification écrite en CSpeL. Les ensembles des données personnelles, spécifié par \personalData, et des finalités, spécifié par \purposes, sont traduits en des types énumérés (typedef enum) dans le code généré. En se basant sur ces ensembles, CASTT génère la matrice Consent qui spécifie, pour chaque type de donnée personnelle, pour quelles finalités le consentement a été donné. Cette matrice est déclarée en tant que code ghost. En ACSL, les variables et les instructions annotées en tant que code ghost se comportent comme des variables et instructions C classiques mais ne sont visibles que dans les spécifications ACSL. De cette façon, elles n'interfèrent pas avec le code C de l'utilisateur [59]. La commande \isGranted est utilisée pour générer les instructions ghost initialisant la matrice précédente. Cette initialisation a lieu dans les processus indiqués à l'aide de \init. Similairement, la matrice Need, indiquant quelles données sont nécessaires pour le fonctionnement de chaque processus, est générée à partir de l'ensemble des processus (\process) et de l'ensemble des données personnelles (\personalData). Elle est initialisée à partir du contenu de la commande \needData.

Pour chaque instruction de chaque fonction (*i.e.* processus), si cette instruction correspond à un traitement d'une donnée personnelle (par exemple, lecture, écriture), des assertions ACSL (`assert`) sont générées. Ces clauses correspondent à la traduction de mes propriétés. Pour la propriété *conformité aux finalités consenties*, la clause vérifie que le consentement a été accordé pour traiter cette donnée personnelle pour au moins l'une des finalités de la fonction. Pour la propriété *conformité à la nécessité des données*, la clause vérifie que cette donnée est nécessaire pour cette fonction. Ces informations sont définies par l'utilisateur dans le fichier CSpeL via les commandes `\personalData` (pour identifier les données personnelles) et `\hasPurposes` (pour récupérer les finalités des fonctions). Ainsi, cela permet de detecter de façon exhaustive les traitements qui ne sont pas conformes dans ce contexte.

Ensuite, pour toute annotation ACSL `/*@ assert Consent[`d`][`p`] ≡ 1;*/`, (avec d une donnée personnelle et p une finalité) et pour toute annotation `/*@ assert Need[`s`][`d`] ≡ 1;*/` (avec d une donnée personnelle et s un nom de fonction), les plug-ins de vérification de Frama-C utilisés vont vérifier si ces propriétés sont satisfaites. De cette façon, cela assure à la fois que l'utilisateur a consenti au traitement de la donnée d pour la finalité p, et que la donnée d est nécessaire à la fonction s.

9.4.4 . Experimentations

Le mécanisme de traduction de CASTT pour un outil au niveau Programme est évalué en tenant compte des questions de recherche suivantes :

RQ10 *CASTT peut-il traduire un fichier CSpeL pour un outil au niveau Programme ?*

RQ11 *CASTT peut-il être utilisé pour vérifier des systèmes au niveau Programme ?*

RQ12 *CASTT peut-il être utilisé pour détecter les traces ne respectant pas la conformité aux finalités consenties ?*

RQ13 *CASTT peut-il être utilisé pour détecter les traces ne respectant pas la conformité à la nécessité des données ?*

RQ14 *CASTT permet-il de réduire le nombre de lignes de spécification à écrire par l'utilisateur ?*

RQ15 *CASTT est-il utilisable pour des codes de taille importante (i.e. avec un grand nombre d'appels de fonctions) ?*

Cas d'utilisation

Pour les expérimentations, nous avons utilisé des exemples fondés sur des scénarios représentant différentes exécutions des systèmes présentés à la section 7. Nous avons utilisé les scénarios détaillés dans le tableau 9.8 pour l'exemple GDM_{Spl}, et dans le tableau 9.9 pour l'exemple AL_{Spl}. L'exemple GDM_{Cpx} n'a pas été utilisé pour ces évaluations, car il n'a pas d'implémentation définie.

Résultats

Pour répondre aux questions **RQ10**, **RQ11**, **RQ12** et **RQ13**, j'ai exécuté un script de *correction*. Ce script est composé des commandes suivantes :

```
frama-c <source_file.c>
-castt-annotate
-castt-consent-file <context.cspel>
-then-last
-print -ocode <annotated_file.c>
```

Le fichier `<source_file.c>` contient le code source du système, le fichier `<context.cspel>` contient la spécification du contexte en CSpeL, et le fichier `<annotated_file.c>` le fichier C correspondant au fichier source avec les annotations générées suite à l'exécution de cette commande. Un exemple de fichier source est disponible en annexe A.1.4 et de fichier généré est disponible en annexe A.1.5 pour GDM_{Spl}. Un exemple de fichier source est disponible en annexe A.3.5 et de fichier généré est disponible en annexe A.3.6 pour Achat en Ligne - version Simple.

Table 9.8 – Scénarios pour la vérification à l'aide de Frama-C - GDM_{Spl}.

\context		main

\process{makePrescr, getData, computeStats}, \personalData{Patient, Data}, \purposes{Treatment, Research}, \isGranted{(Patient : Treatment)}, \hasPurposes{ (makePrescr :{Treatment}), (getData :{Treatment,Research}), (computeStats :{Research})}, \needData{(makePrescr :{Patient}), (getData :{Patient, Data}), (computeStats :{Data})}, \init{main}	T1	`Patient patient = {... };`
		`Dos newd = {... };`
		`patient = makePrescr(newd, patient);`

	T2	`Patient patient = {... };`
		`Dos newd = {... };`
		`Patient list1 [NB_PATIENT] = patient;`
		`Data d = getData(list1);`
		`int res = computeStats(d);`

	T3	`Patient patient = {... };`
		`Dos newd = {... };`
		`patient = makePrescr(newd, patient);`
		`patient = makePrescr(newd, patient);`

	T4	`Patient patient = {... };`
		`Dos newd = {... };`
		`patient = makePrescr(newd, patient);`
		`Patient list1 [NB_PATIENT] = patient;`
		`Data d = getData(list1);`
		`int res = computeStats(d);`

Ensuite, pour la vérification statique, le script exécute la commande

```
frama-c -<analyzer> <annotated_file.c>
```

où l'option `-analyzer` est soit `-eva` pour exécuter le plug-in Eva ou `-wp` pour le plug-in WP.

Table 9.9 – Scénarios pour la vérification à l'aide de Frama-C - AL$_{Cpx}$.

<table>
<tr><td>\context</td><td>main</td></tr>
<tr><td rowspan="11">\process{buy,store,display, getAccount
getHistory,chooseAd,sendAd, actionUser},
\personalData{ClientAccount,Client},
\purposes{Administrative, Marketing},
\isGranted{(ClientAccount : Administrative),
(Client : Administrative)}
\hasPurposes{(buy :{Administrative}),
(store :{Administrative}),
(display :{Administrative}),
(getAccount :{Administrative,Marketing}),
(getHistory :{Administrative,Marketing})
(sendAd :{Marketing})},
(actionUser :{Administrative,Marketing})},
\needData{(buy :{ClientAccount,Client}),
(store :{ClientAccount}),
(display :{ClientAccount}),
(getAccount :{ClientAccount,Client}),
(getHistory :{Client}),(sendAd :{Client}),
(actionUser :{ClientAccount,Client})}</td><td>Ad ad = {... };
Purchase p1 = {... };
Client client = {... };
PurchaseList purchases = {... };
int res = 0;
res = actionUser(client,purchases);</td></tr>
<tr><td>actionUser</td></tr>
<tr><td>T1 | int actions[2] = {0,1};</td></tr>
<tr><td>T2, T3 | int actions[1] = {2};</td></tr>
<tr><td>store</td></tr>
<tr><td>T3 | Client c = cc.client;</td></tr>
</table>

Le script surveille également le code généré par CASTT avec E-ACSL pour la vérification dynamique. Dans ce cas, nous remplaçons `-analyzer` par `-e-acsl -then-last -print -ocode <monitor.c>` afin de générer le moniteur E-ACSL. Ensuite, il est compilé et exécuté comme suit :

```
e-acsl-gcc.sh -C -O <monitored_binary> <monitor.c>
./<monitored_binary>.e-acsl
```

Table 9.10 – Résultats des expérimentations de la vérification de code par Frama-C.

Plug-in	**Résultat attendu**	**Signification**	**Évaluation**
WP	tous les buts (*goals*) sont prouvés	Valide	✓
Eva	toutes les assertions sont valides		✓
E-ACSL	aucune erreur n'est levée à l'exécution		✓
WP	certains buts ne sont pas prouvés	Invalide	✓
Eva	certaines assertions sont invalides		✓
E-ACSL	une erreur est levée à l'exécution		✓

Toutes les expériences fournissent instantanément leurs résultats, qui sont résumés dans les tableaux 9.10 et 9.11. Le premier tableau montre, pour chaque

plug-in de Frama-C, les résultats attendus, leur signification pour mon approche, et si le résultat de l'expérimentation correspondait au résultat attendu. De même que précédemment, les scénarios sont définis manuellement de telle sorte qu'ils soient valides ou invalides, selon la définition de l'exemple (le but étant de pouvoir vérifier s'ils sont correctement détectés).

Pour un résultat plus fin, c'est à dire pour s'assurer que l'annotation non respectée est bien celle attendue, la méthode diffère selon le plug-in utilisé. Pour cette évaluation avec Eva et WP, la vérification se fait manuellement à l'aide de l'interface graphique de Frama-C. Pour E-ACSL, l'erreur levée durant l'exécution spécifie quelle annotation ACSL n'est pas satisfaite à l'exécution.

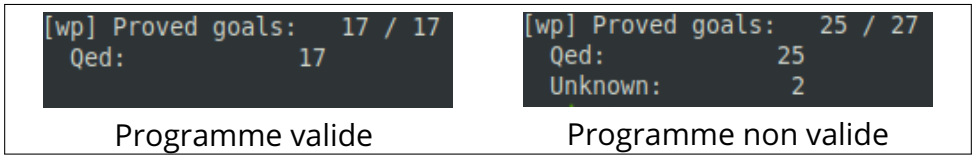


Figure 9.9 – Exemples de verdicts WP lorsqu'un programme est valide ou invalide.

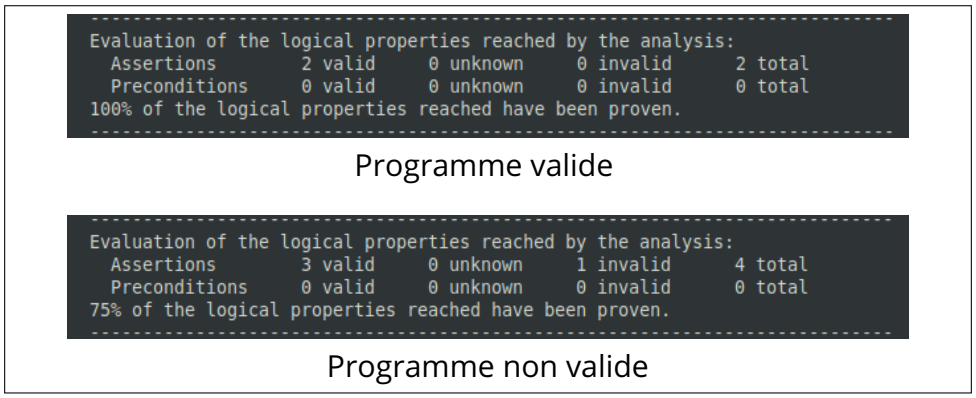


Figure 9.10 – Exemples de verdicts Eva lorsqu'un programme est valide ou invalide.

La figure 9.9 présente des exemples de verdicts de WP, la figure 9.10 des exemples de verdicts de Eva et la figure 9.11 une erreur levée par E-ACSL.

Ces expérimentations montrent que CASTT, combiné avec un de ces plug-ins de vérification de Frama-C, réussit à vérifier la conformité du code fourni, selon le contexte spécifié. Je peux donc répondre positivement aux questions de recherche **RQ10**, **RQ11**, **RQ12** et **RQ13**.

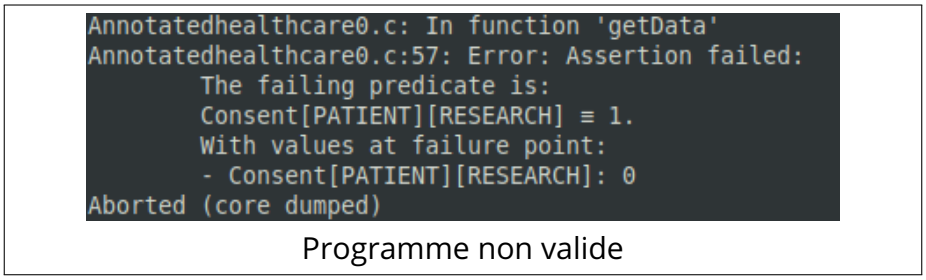


Figure 9.11 – Exemple d'erreur levée par E-ACSL lorsqu'un programme est invalide.

Table 9.11 – Résultats des expérimentations pour la génération de code par CASTT.

Exemple	GDM_{Spl}			AL_{Cpx}		
	T1	T2	T3	T1	T2	T3
# lignes dans le code source	53	54	50	121	122	121
# lignes générées par CASTT	33	35	33	92	94	90
# lignes dans le fichier CSpeL	10			18		

60-75%

× 3-5

Le tableau 9.11 présente pour chaque cas de test : le nombre de lignes de code dans le fichier source original, le nombre de lignes générées par CASTT, le nombre de lignes nécessaires pour WP et la taille de la spécification en CSpeL. Un exemple de fichier généré par CASTT est donné en annexe A.1.5. Comme cela est visible sur le tableau, dans ces cas de tests, les fichiers CSpeL sont utilisés pour générer des fichiers d'une taille 3 à 5 fois plus grande (en nombre de lignes). Ces lignes générées correspondent à environ 60% à 75% du fichier source original.

De plus, le morceau de code correspondant aux exécutions des processus peut évoluer indépendamment du fichier de spécification. Par exemple, sur nos sept exemples utilisés pour la génération de code, un seul fichier CSpeL a été utilisé pour chacun des exemples (GDM_{Spl} ou AL_{Cpx}) sur différents fichiers C. Nous pouvons donc répondre positivement à la question de recherche **RQ14**.

Pour répondre à la question **RQ15**, j'ai exécuté un script mesurant le temps d'exécution. Le script exécute les commandes précédentes sur un même fichier C (excepté pour le nombre d'appels de fonction) avec un même fichier CSpeL. Pour cela, j'utilise un générateur d'appels de fonction pour augmenter le nombre d'appels dans le fichier source (fonctions appelées dans la fonction `main`). Comme le plug-in WP est modulaire (chaque fonction est analysée indépendamment des autres), il n'est pas inclus dans ces tests car les résultats n'auraient pas été pertinents, les modifications n'influant que sur le nombre d'appels et non sur le contenu des fonctions appelées.

La figure 9.12 présente les résultats obtenus pour des fichiers contenant entre 10 et 100000 appels de fonction. Pour chaque taille, le test est exécuté 10 fois ce qui permet de calculer un temps moyen. Le script récupère le temps pour la vérification avec et sans les annotations générées par CASTT, pour calculer le surcoût généré par mon approche. Deux résultats ne correspondent pas à ce qui aurait pu être attendu : le temps d'analyse pris par Eva pour 40000 et 60000 appels de fonctions, est inférieur pour le code contenant les annotations générées par CASTT que pour le code sans les annotations. Nous n'avons pas d'explications à l'heure actuelle.

Cette évaluation montre que le temps de génération des annotations de CASTT est négligeable par rapport au temps d'analyse des plug-ins de vérification. Elle montre également que nos annotations générées ne ralentissent pas trop le processus de vérification (généralement moins de 10% pour Eva et généralement moins de 5% pour E-ACSL). Elle montre aussi que plus la taille du programme original est importante, plus le surcoût relatif est faible. En particulier, lorsque l'évaluation porte sur un code comportant 1000000 d'appels de fonctions, le surcoût pour Eva est d'environ $1,6\%$, alors qu'il est de $0,50\%$ pour E-ACSL. CASTT fonctionne beaucoup plus rapidement que Eva ou E-ACSL. En particulier, il est toujours au moins 5 fois plus rapide dès que l'on dépasse 5000 d'événements. Le temps d'annotation par CASTT est environ 7 fois inférieur au temps de vérification par Eva ou E-ACSL. Par conséquent, je peux répondre positivement à **RQ15**.

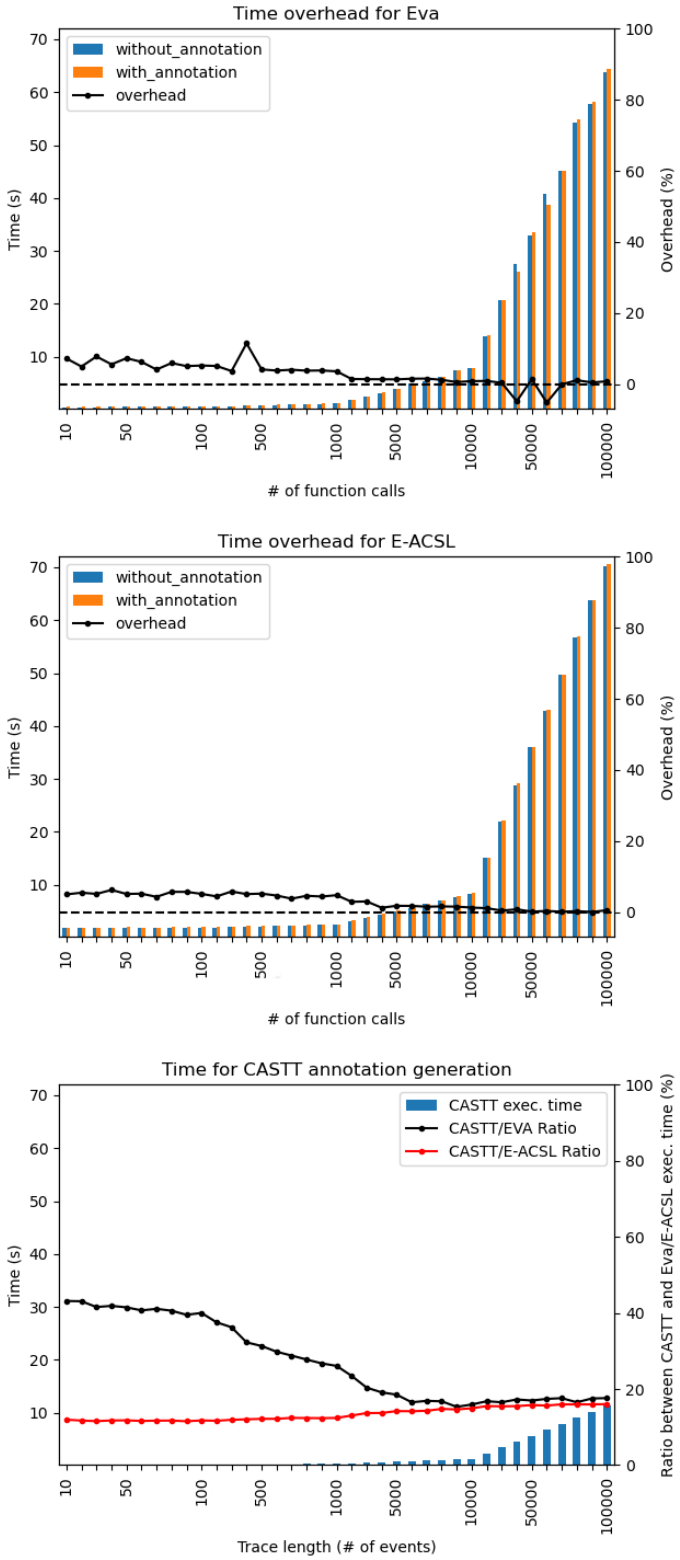


Figure 9.12 – Temps d'exécution selon le nombre d'appels de fonction [41].

9.5 . Fonctionnalités annexes

Cette section présente des fonctionnalités annexes de CASTT. Ces fonctionnalités ne sont pas directement liées à la vérification de propriétés mais servent de support à mon approche. Actuellement deux fonctionnalités annexes sont implémentées : un mécanisme de génération de traces et un mécanisme de génération d'appels de fonction.

Ces mécanismes, bien qu'existants et utilisés pour la publication [41], n'y étaient pas détaillés (par manque de place).

9.5.1 . Générateur de traces

Problème

Les traces utilisées pour les expérimentations étant actuellement spécifiées manuellement, les traces pour les expérimentations de *correction* sont de tailles relativement réduites (de 1 à 8 évènements par trace). Elles correspondent à des scénarios et leur but est de vérifier le fonctionnement de notre approche. Mais leur taille réduite ne permet pas de savoir quelles sont les limites de notre outil.

But

Le but du générateur de traces est de pouvoir générer automatiquement une trace à partir d'un contexte spécifié en CSpeL, selon une taille donnée et de l'ajouter à ce fichier, dans le but de pouvoir tester des traces composées d'un grand nombre d'évènements (jusqu'à un million).

Principe

Le principe de fonctionnement du générateur de traces est illustré sur la figure 9.13.

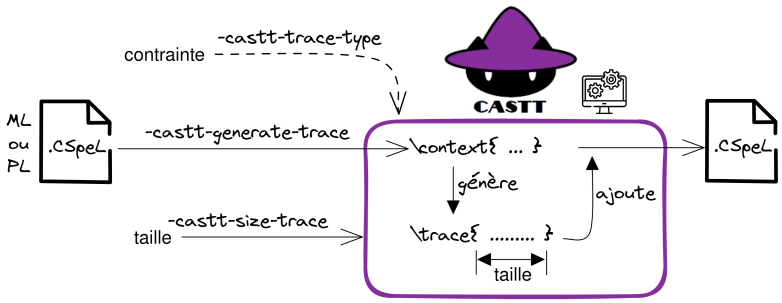


Figure 9.13 – Schéma fonctionnel de génération des traces par CASTT.

L'utilisateur donne en entrée un fichier écrit en CSpeL contenant une spécification du contexte. À l'aide de l'option `-castt-generate-trace`, il demande

à CASTT de générer une trace à partir de ce contexte et contenant un nombre d'évènements égal à la taille demandée, soit `-castt-trace-size`. L'utilisateur peut optionnellement donner une contrainte pour cette génération, *via* l'option `-castt-trace-type` :

- `purpose` pour générer une trace ne contenant que des traitements *conformes aux finalités consenties* ;
- `necessity` pour générer une trace ne contenant que des traitements *conformes à la nécessité des données* ;
- `end-non-purpose` pour générer une trace contenant des traitements *conformes aux finalités consenties* suivis d'un dernier traitement non *conforme aux finalités consenties* ;
- `end-non-necessity` pour générer une trace contenant des traitements *conformes à la nécessité des données* suivis d'un dernier traitement non *conforme à la nécessité des données* ;
- `all-compliant` pour générer une trace contenant uniquement des traitements à la fois *conforme aux finalités consenties* et *conforme à la nécessité des données*.

Si aucune contrainte n'est spécifiée (ou si celle-ci a pour valeur `none`), la trace sera générée en sélectionnant aléatoirement les évènements.

Implémentation

Pour cette fonctionnalité, on récupère le fichier de spécification (écrit en CSpeL), et la taille de la trace demandée. Si le fichier de spécification est bien formé, la génération peut alors commencer. Un nouveau fichier est généré, conformément aux contraintes spécifiées le cas échéant, correspondant au fichier de spécification complété de la trace demandée. Comme indiqué précédemment, différentes contraintes peuvent être demandées pour la génération : `purpose`, `necessity`, `end-non-purpose`, `end-non-necessity`, `all-compliant`, bien ou pas de contrainte spécifique (équivalent à la contrainte `none`). Le choix de la contrainte va influer sur les ensembles de sélection, comme expliqué ci-dessous.

La figure 9.14 représente le principe de la génération d'une trace, quelle que soit la contrainte demandée. À partir du contexte spécifié en CSpeL et de la contrainte demandée, on va construire deux ensembles : un dit de « corps » et un dit de « fin » (avec potentiellement une étape de filtrage à partir de l'ensemble de tous les évènements possibles pour créer l'ensemble de « fin »). Ensuite, on génère la trace à partir de ces ensembles. Les $n-1$ premiers éléments de la trace (n étant la taille demandée) sont sélectionnés aléatoirement dans l'ensemble « corps » et le dernier élément de la trace est sélectionné aléatoirement dans l'ensemble « fin » (nécessaire pour les tests de fin de trace). Les contenus de ces ensembles dépendent du contexte mais aussi de la contrainte demandée.

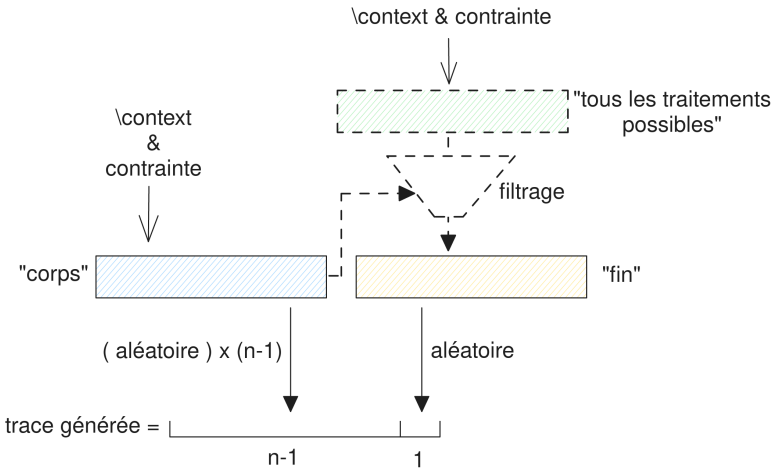


Figure 9.14 – Principe de génération d'une trace.

Contrainte `purpose`

Comme illustré sur la figure 9.15, pour une trace ne contenant que des traitements *conformes aux finalités consenties* : les ensembles de corps et de fin sont les mêmes. Ils sont construits de telle sorte qu'ils ne contiennent que des évènements étant *conformes aux finalités consenties*. Pour cela on s'appuie sur la définition du consentement (`\isGranted`) et de l'association entre les finalités et les processus (`\hasPurposes`).

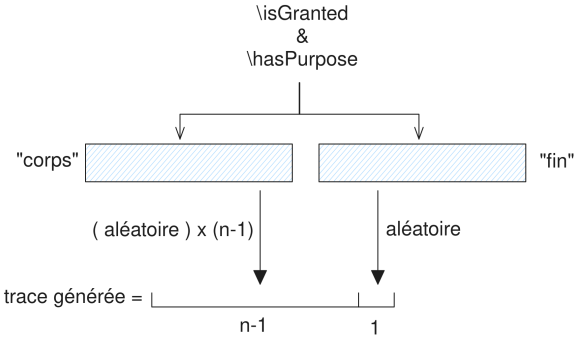


Figure 9.15 – Génération d'une trace pour la contrainte `purpose`.

Contrainte `necessity`

Comme illustré sur la figure 9.16, pour une trace ne contenant que des traitements *conformes à la nécessité des données* : la méthodologie est la même que pour la trace ne contenant que des traitements *conformes aux finalités consenties*, mais en considérant cette fois des évènements *conformes à la nécessité des données*, en s'appuyant sur la spécification des données personnelles (`\personalData`) et sur l'association entre les données personnelles et les processus (`\needData`).

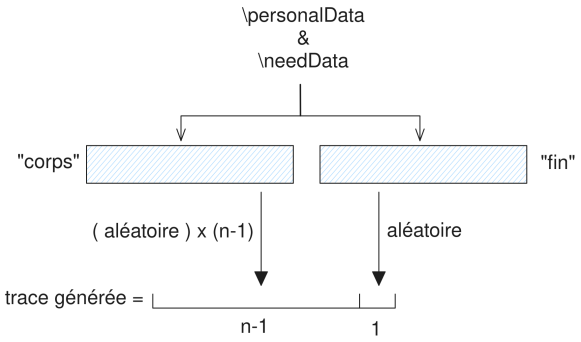


Figure 9.16 – Génération de trace pour la contrainte `necessity`.

Contrainte `end-non-purpose`

Comme illustré sur la figure 9.17, pour une trace contenant des traitements *conformes aux finalités consenties* suivis d'un dernier traitement non *conforme aux finalités consenties* : les ensembles de corps et de fin sont différents. Le premier est construit de telle sorte à ne contenir que des évènements *conformes aux finalités consenties*. Pour cela, on s'appuie sur la définition du consentement (`\isGranted`) et sur l'association entre les processus et les finalités (`\hasPurposes`) de telle sorte que ces évènements soient *conformes aux finalités consenties*. Le deuxième ensemble est construit de telle sorte à ne contenir que des évènements non *conformes aux finalités consenties*. Pour cela on construit un ensemble contenant tous les évènements possibles, à partir de la spécification des données personnelles (`\personalData`), et de la spécification des processus (`\process`), puis on retire tous ceux *conformes aux finalités consenties*.

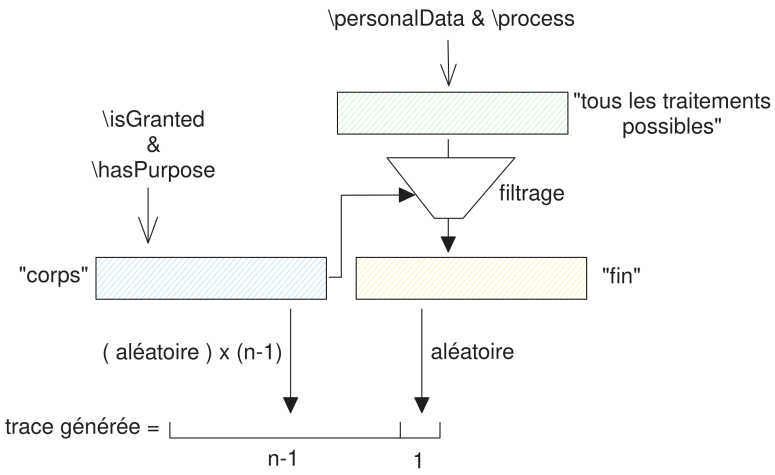


Figure 9.17 – Génération de trace pour la contrainte `end-non-purpose`.

Contrainte `end-non-necessity`

Comme illustré sur la figure 9.18, pour une trace contenant des traitements *conformes à la nécessité des données* suivis d'un dernier traitement non *conforme à la nécessité des données* : la méthodologie est la même que pour la trace contenant des traitements *conformes aux finalités consenties* suivis d'un dernier traitements non *conformes aux finalités consenties*, mais en considérant des évènements *conformes à la nécessité des données*. La sélection s'appuie cette fois-ci sur la spécification des données personnelles (`\personalData`) et sur l'association entre les données personnelles et les processus (`\needData`).

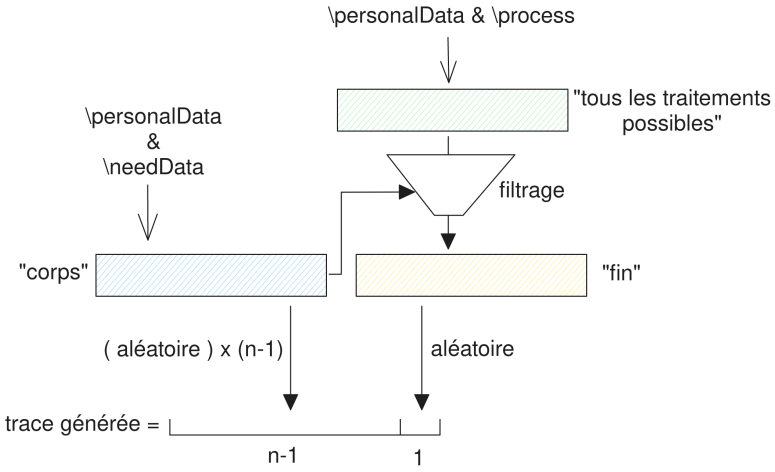


Figure 9.18 – Génération de trace pour la contrainte `end-non-necessity`.

Contrainte `all-compliant`

Pour une trace ne contenant que des traitements à la fois *conformes aux finalités consenties* et *conformes à la nécessité des données* : les ensembles de corps et de fin sont les mêmes. Pour les construire, on réalise l'intersection de l'ensemble des traitements *conformes aux finalités consenties* avec celui des traitements *conformes à la nécessité des données*.

Pas de contrainte (ou contrainte `none`)

Comme illustré sur la figure 9.19, pour une trace ne contenant que des traitements sans contraintes spécifiques : Les ensembles de corps et de fin sont les mêmes. Ils sont construits en contenant tous les évènements possibles, à partir de la spécification des données personnelles (`\personalData`) et de la spécification des processus (`\process`).

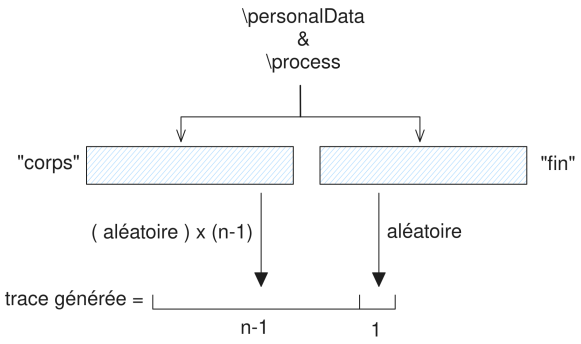


Figure 9.19 – Génération de trace sans contrainte.

Une fois la trace construite, un nouveau fichier CSpeL est généré. Ce fichier contient la spécification du fichier source et la trace qui a été générée. Si la génération demandée est impossible (par exemple, aucun traitement ne respecte la contrainte), une trace vide est générée. Une extension permettant de spécifier également des données non personnelles est laissée en travaux futurs, comme expliqué en section 10.3.5.

9.5.2 . Générateur d'appels de fonctions

Problème

La taille des fichiers C utilisés pour les tests de *correction* sont de faible taille (51 à 86 lignes de code). Ce nombre limité est dû au fait que les fichiers sont écrits manuellement à partir des exemples que nous avons définis pour notre approche. Ces exemples sont détaillés en section 7. Cependant, leur taille réduite ne permet pas de savoir quelles sont les limites de mon outil.

But

Le but du générateur d'appels de fonctions est d'augmenter la taille des expérimentations pour l'outil CASTT pour la partie annotation de code (fonction de traduction). Autrement dit, une des utilisations de ce générateur est de générer des appels de fonction correspondant aux traces générées par le générateur de traces. Le but de ce générateur de code source C serait d'implémenter la fonction `main` d'un fichier C fourni par l'utilisateur, à partir d'une spécification donnée en CSpeL.

Principe

La figure 9.20 illustre l'utilisation de la fonctionnalité de génération d'appels de fonction. L'utilisateur donne en entrée le code source écrit en C et un fichier contenant une spécification des appels à générer, écrit en GSpeL (ce langage de spécification est détaillé dans la section suivante). À l'aide de l'option `-castt-generate-function-calls`, il demande à CASTT de générer les appels de fonctions. À partir de ces fichiers, CASTT ajoute dans l'AST du code source les

appels de fonctions voulus. Il est alors possible de demander à Frama-C de générer un nouveau fichier C à partir de l'AST, en ajoutant `-then-last -print -ocode <filename.c>`.

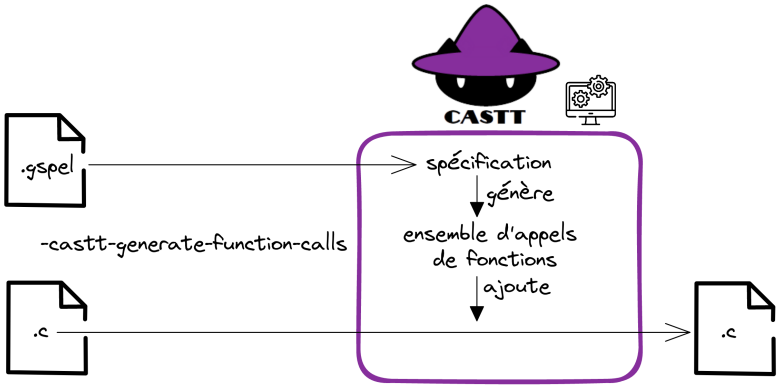


Figure 9.20 – Utilisation de la fonctionnalité de génération d'appels de fonctions.

GSpeL - Generation Specification Language

GSpeL est un langage dédié à la spécification de génération d'appels de fonction. Sa grammaire est détaillée dans le tableau 9.12.

Table 9.12 – Grammaire de GSpeL

TG	::=	[FList] \function_name = $fname$
FList	::=	FElem | FElem ; FList
FElem	::=	(\function_name = $fname$, \param_list = [PList] , \return_var_name = $vname$, \times = t)
PList	::=	p | p ; PList

Un fichier GSpeL permet de spécifier quels sont les appels à générer, et à la fin de quelle fonction les ajouter. Les appels de fonctions à ajouter sont définis dans une liste. Pour chaque fonction, on spécifie son nom (\function_name), ses paramètres (\param_list), la variable pour stocker sa valeur de retour (\return_var_name), et le nombre de fois que l'on veut générer cet appel (\times). L'exemple 12 montre une spécification écrite en GSpeL.

Exemple 12: Fichier GSpeL

```
[
  (
    \function_name=makePrescr,
    \param_list=[patient;newprescr],
    \return_var_name=patient,
    \times=5
  );
  (
    \function_name=getData,
    \param_list=[list1],
    \return_var_name=d,
    \times=3
  )
]
\function_name=main
```

Ce fichier GSpeL spécifie une demande pour générer 5 fois la fonction `makePrescr` avec comme paramètres les variables `patient` et `newprescr`, la valeur de retour étant écrite dans la variable `patient`, puis 3 fois la fonction `getData` avec comme paramètre la variable `list1`, la valeur de retour étant écrite dans la variable `d`. Ces appels de fonction doivent être générés à la fin de la fonction `main`.

Implémentation

Au niveau de l'implémentation, l'option `-castt-generate-function-calls` a été ajoutée à CASTT. CASTT parse le fichier de spécification contenant les informations pour la génération. À la fin de la fonction spécifiée (*i.e.* avant le `return`), l'outil génère les instructions nécessaires. Pour cela, la fonction est parcourue afin d'identifier l'endroit d'insertion (les instructions précédentes et suivantes étant gardées en mémoire, pour ne pas perdre des instructions). L'outil « étend » la liste des fonctions pour construire la liste complète des fonctions à générer. Ensuite, pour chacune des fonctions contenues dans la liste, l'outil génère une instruction d'appel. Ces instructions sont ajoutées à l'emplacement d'insertion (entre les instructions précédentes et les suivantes)

Limitations

Actuellement, il existe plusieurs limitations à la génération d'appels de fonction :

— le cas où il n'y a pas de valeur de retour n'est pas géré (car non nécessaire pour les cas d'utilisation) ;

— aucune gestion d'erreur n'est implémentée. Il faut absolument que les paramètres et la variable de retour correspondent à des variables existantes

dans la fonction parente (celle où ajouter l'appel). De même, la fonction à appeler doit exister dans le code.

— les appels ne peuvent être ajoutés que dans une seule fonction.

Cette section a présenté CASTT, l'outil que j'ai développé pour vérifier des propriétés de consentement depuis une spécification écrite en CSpeL. CASTT a trois fonctionnalités pour la vérification : un mécanisme de vérification de trace, un mécanisme de traduction vers un outil au niveau Modèle Formel (IAT), et un mécanisme de traduction vers un outil au niveau Programme (Frama-C). J'ai présenté le principe de fonctionnement de ces mécanismes, leur implémentation et les expérimentations que j'ai réalisées pour les évaluer. Cette section a également présenté deux fonctionnalités de support de CASTT pour les expérimentations : un mécanisme de génération de traces et un mécanisme de génération d'appels de fonction.

La section suivante conclut la partie vérification de mon approche, ainsi que des travaux futurs éventuels qui pourraient être réalisés pour améliorer les contributions de cette partie.

10 - Conclusion et perspectives de la partie vérification de propriétés de respect de la vie privée

10.1 . Discussion et limites

Cette section discute les choix fait pour notre approche et leurs limitations.

10.1.1 . Focus sur la catégorie de *caractérisation du Traitement*

Pour mon approche de vérification, j'ai choisi de me concentrer sur des notions liées à la catégorie *caractérisation du Traitement*, selon sa définition de GePyR, donnée en Section 4.1 : « La *caractérisation du Traitement* fait référence à toute notion liée au traitement autorisé des données personnelles ». Pour rappel ces notions sont distinctes de celles de la catégorie de *caractérisation de la Visibilité*.

La catégorie qui nous intéresse ici inclut quatres sous-catégories liées :

— aux finalités du traitement (*Purpose*) ;

— à la nécessité des données (pour le traitement) (*Necessity*) ;

— à la durée du traitement (*Retention Duration*) ;

— à l'évolution de l'accord pour le traitement (*Consent Evolution*).

Les notions liées au traitement ont peu de solutions de vérification dans l'état de l'art, comme montré en section 4.4. Pourtant, ce sont des notions importantes, comme on peut le voir dans la section 4.1 et dans la section 4.2. Cette catégorie a en effet un grand nombre de spécialisations dans les différentes représentations de la vie privée. De plus, les condamnations pour non respect du consentement lors du traitement de données personnelles sont de plus en plus nombreuses et les montants d'amendes sont de plus en plus importants[1] [2]. Ainsi, vérifier que le traitement des données personnelles dans un système respecte la vie privée est important, que ce soit du point de vue des entreprises (pour être conforme à la législation et ne pas subir de sanctions) ou du point de vue des personnes concernées (qui ont une garantie sur le fait que le traitement de leurs données personnelles est réalisé en respectant de leur vie privée).

Ceci explique pourquoi mon approche de vérification a été concentrée sur cette catégorie, sans tenir compte des autres : la *caractérisation de la Visibilité*, la *ca-*

1. https://www.lemonde.fr/pixels/article/2019/01/21/donnees-personnelles-la-cnil-condamne-google-a-une-amende-record-de-50-millions-d-euros_5412337_4408996.html

2. https://www.lemonde.fr/pixels/article/2021/09/02/donnees-personnelles-whatsapp-sanctionne-d-une-amende-record-par-le-regulateur-irlandais_6093152_4408996.html

ractérisation de la Transparence, la *caractérisation de l'Exactitude des données* et la *caractérisation de l'Exactitude des données*, définies en section 4.1.

10.1.2 . Niveaux d'abstraction

Pour mon approche, je me suis intéressée à différents niveaux d'abstraction (modèle et programme). Ces niveaux d'abstraction sont en lien avec une partie des étapes de développement dans le cycle de vie d'un logiciel (conception et implémentation).

Ces niveaux ont été sélectionnés selon mon étude de l'état de l'art. En effet, la plupart des articles étudiés pouvaient être classés selon ces différents niveaux. Dans les approches de la littérature, on peut également identifier deux sous-niveaux au sein du niveau modèle. On peut ainsi distinguer un sous-niveau modèle graphique (utilisant des langages de modélisations graphiques, sans sémantique formelle, tel que BPMN [30]) et un sous-niveau modèle formel (utilisant des langages de modélisation ayant une sémantique formelle, tel que dans la solution proposée par Ta et Eiza [165]).

En me basant sur ces niveaux d'abstraction, j'ai choisi des outils permettant de vérifier des propriétés sur des systèmes définis à ces niveaux (modèle et programme), c'est à dire IAT et Frama-C.

10.1.3 . Spécification de la vie privée indépendante du système

Un choix important dans mon approche est la séparation entre la spécification des notions de vie privée et la spécification du système (modélisation au niveau modèle et implémentation au niveau programme). En effet mon approche considère que le système en lui même ne s'appuie pas forcément sur des notions de vie privée. Par exemple, le programme du système ne fera potentiellement pas de branchement sur des tests de consentement. Plusieurs raisons expliquent ce choix. Tout d'abord cela permet à mon approche de s'appuyer sur des systèmes existants n'ayant pas pris en compte ces notions. Cela permet également de déléguer cette spécification à une personne experte des problématiques de vie privée, par exemple le délégué à la protection des données (*data protection officer*) spécifié dans le RGPD [55]. De plus, notre approche peut malgré tout être appliquée à des systèmes ayant des mécanismes internes de protection (tel que le branchement existant dans les exemples GDM_{Cpx} en section 7.3.2, au niveau modèle, et AL_{Cpx} en section 7.4.2, au niveau programme). Tout dépend de la spécification donnée par l'utilisateur. Ainsi, mon approche est souple en termes de spécification du système.

10.1.4 . Complexité des exemples

Les exemples ayant servi pour réaliser mes tests sont présentés section 7. J'ai défini ces exemples pour illustrer et tester mon approche. Ils s'échelonnent sur plusieurs niveaux de complexité et différents cas peuvent être testés (par exemple, si des processus ont plusieurs finalités associées). L'exemple GDM_{Cpx} est un exemple de la littérature concernant le domain médical [132], que j'ai simplifié pour définir

l'exemple GDM_{Spl}. J'ai également défini deux autres exemples concernant un autre domaine d'application, les sites internets, permettant ainsi de montrer la variété d'applications possibles de mon approche. De même que pour le domain médical, pour le domaine des sites internets, j'ai défini un exemple simple AL_{Spl} et un complexe AL_{Cpx}. Malgré tout, ces exemples sont de petite taille et ne sont pas des cas d'usage réels. Pour surmonter la première limite, j'ai effectué des tests de charge pour mon outil à l'aide de générateurs que j'ai implémentés. Ils ne sont toutefois pas toujours représentatifs de programmes industriels. Actuellement nous n'avons pas trouvé d'exemples réels (et libres d'accès) pouvant être utilisés pour évaluer mon approche, c'est à dire des exemples de système traitant de donnés personnelles, avec un système défini à l'aide d'un langage de modélisation ou bien un système implémenté en C.

10.1.5 . Approche générique

L'aspect générique de mon approche est un des ses principaux atouts. En effet, le but était de pouvoir exprimer les mêmes propriétés de consentement sur des systèmes variés (donc différents domaines d'application), à différentes étapes de développement du cycle de vie d'un logiciel (donc different niveaux d'abstraction). Pour cela, mes contributions présentées dans la partie classification dans le chapitre I, m'ont permis d'identifier les éléments à prendre en compte pour la vérification de propriétés liées au consentement, ainsi que leurs relations. Ceci m'a permis de proposer CSpeL et CASTT. Mon approche peut être ainsi adaptée selon les besoins de l'utilisateur (suivant son domaine de travail et à quel niveau de développement il intervient). Cela permet, de plus, d'avoir une vérification tout au long du cycle de développement. Par contre, cet aspect générique, basé sur une étude de la littérature et de ses définitions très variées de la vie privée, pourrait ne pas couvrir dans les détails toutes les spécificités qui peuvent exister dans certaines représentations (tel que le RGPD [55] par exemple).

10.1.6 . Traces d'exécutions

Dans un premier temps, par soucis de généricité, mon approche se fondait sur des traces d'exécutions suffisamment génériques pour pouvoir représenter des systèmes à différents niveaux d'abstraction (la définition de ces traces est donnée en section 8.1.2). Il faut spécialiser l'approche pour permettre la vérification directe de systèmes eux-mêmes. Pour cela, j'ai mis en place des mécanismes de traductions : une traduction pour un outil au niveau modèle formel (IAT), présentée en section 9.3.1, et une traduction pour un outil au niveau programme (Frama-C), détaillée en section 9.4. Ces traductions permettent de vérifier non plus des traces d'exécutions spécifiées par l'utilisateur, mais directement les systèmes. Une autre possibilité aurait été de faire générer des traces par les systèmes ; cependant le choix de mettre en place un mécanisme de traduction permet d'avoir un mécanisme de vérification varié (c'est à dire de pouvoir utiliser différents outils de vérification reposant sur différentes approches) bénéficiant d'outils plus matures que CASTT

(analyse statique faisant appel à des solveurs, analyse statique reposant sur de l'interprétation abstraite et de l'analyse dynamique).

10.2 . Travaux connexes

Plusieurs approches dans la littérature permettent de vérifier des propriétés formelles à la fois au niveau du modèle et à celui du programme. Parmi elles, la méthode B [2] permet aux ingénieurs de modéliser les comportements d'un système à un niveau abstrait dans le langage B, et de raffiner le modèle itérativement jusqu'à un modèle concret qui peut être automatiquement traduit dans un langage de programmation comme le C. À l'inverse, Greenaway et al. [64] proposent un outil pour abstraire la sémantique de C dans des spécifications de plus haut niveau. Cependant, ces approches ciblent les propriétés de sûreté, exprimant la façon dont le programme est censé se comporter.

En ce qui concerne les propriétés de sécurité, il faut tenir compte de la présence d'attaquants. Les approches existantes considèrent soit le niveau du modèle, soit le niveau du programme, mais jamais les deux. Par exemple, Bernhard et al. [22] spécifient un nouveau modèle de protocole de vote satisfaisant des propriétés formelles spécifiques, telles que la *secrecy* [158], tandis que Dufay et al. [53] spécifient un langage basé sur JML et utilisent l'analyse statique pour vérifier une propriété de non-interférence.

Table 10.1 – Comparaison d'approches pour la vérification du traitement des données personnelles [41].

Solution	Propriétés formelles	ML	PL	Langage	Outil	Méthode de vérification
[18]	✓	✓	×	non nommé	×	contrats intelligents
[21]	✓	✓	×	CAPVerDE	CAPVerDE	solveurs SMT
[165]	✓	✓	×	non nommé	DataProVe	algorithme ad-hoc
[114]	✓	✓	×	Prolog	basé sur Prolog	vérification à l'exécution
[121]	×	×	✓	non nommé	Poly	application de politiques framework
[168]	×	×	✓	non nommé	CASTOR	correspondance sémantique
[65]	×	×	✓	OpenAPI	OpenAPI	application de politiques framework
[70]	×	×	✓	JIF	JIF	flux d'information control
[mon approche]	✓	✓	✓	CSpeL	CASTT Frama-C	algorithme ad-hoc & plug-ins Frama-C

Le tableau 10.1 compare les travaux relatifs à la vérification des propriétés formelles liées au consentement au niveau du modèle (ML) ou du programme (PL). Pour définir ce tableau, j'ai utilisé mes travaux de la partie classification, et plus particulièrement la classification réalisée à l'aide de GePyR (en section 4), son but étant justement d'identifier des approches comparables. Plusieurs de ces approches proposent de vérifier les propriétés de consentement au niveau du modèle : elles s'appuient sur les contrats intelligents (*smart contracts*) pour les blockchains [18], sur la conception d'une architecture spécifique et sa vérification en utilisant des solveurs SMT [21], sur un langage de politique de respect de la vie privée et un langage de description d'architecture [165], ou encore des traces d'évènements (*logs*) pour vérifier la planification des actions [114]. Parmi ceux-ci, trois s'appuient sur des outils : Bavendiek et al. [21], Ta et Eiza [165] utilisent leur propre outil dédié, tandis que de Montety et al. utilisent Prolog [114]. Cependant, aucune de ces approches ne vérifie une quelconque implémentation.

Certaines autres approches proposent des solutions pour assurer le consentement au niveau du programme, mais elles ne ciblent pas la vérification au niveau du modèle. De plus, elles ne formalisent pas la propriété de consentement vérifiée, mais suivent certains principes de respect de vie privée, typiquement le principe de protection des données dès la conception et par défaut du RGPD (*data protection by design and by default*) [55]. Ces solutions reposent sur l'extension d'un modèle de permission dans une infrastructure logicielle d'application de politique (*Policy enforcement framework*) [121], sur un langage de spécification dédié et une analyse statique [168], sur l'extension d'une API d'application de politique (*Policy enforcement framework*) [65], ou encore sur le contrôle du flot d'information [70]. Tous ces projets reposent sur des outils : Hayati et Abadi [70] proposent de réutiliser un outil existant, qui n'est à l'origine pas conçu pour des problématiques relatives à la vie privée. De même, Nauman et al. [121] et Grünewald et al. [65] étendent des outils existants pour s'attaquer à des problématiques de vie privée. Tokas et al. [168] préfèrent implémenter un outil dédié en partant de zéro.

Pour résumer, ma solution est la seule (à ma connaissance) qui cible à la fois le niveau du modèle et le niveau du code pour vérifier les propriétés de vie privée liées au consentement. C'est également la seule solution qui permet de vérifier une propriété de consentement formellement spécifiée au niveau du code.

10.3 . Travaux futurs

Cette section présente la suite des travaux envisagée.

— Pour l'approche dans son ensemble :

- appliquer l'approche sur des exemples réels, présenté en section 10.3.1 ;
- appliquer l'approche à un autre domaine : le *Privacy Preserving Record Linkage* (PPRL), présenté en section 10.3.2 ;

- ajouter une fonctionnalité de vérification du raffinement, présenté en section 10.3.3.

— Pour le langage de spécification, CSpeL :

- ajouter la possibilité de personnaliser les propriétés à vérifier, présenté en section 10.3.4 ;
- ajouter la possibilité de spécifier des données non personnelles, en section 10.3.5 ;
- implémenter une interface utilisateur pour CSpeL, en section 10.3.6.

— Pour les propriétés de vie privée :

- formaliser et vérifier une propriété d'évolution du consentement, présenté en section 10.3.7 ;
- formaliser et vérifier une propriété de durée de stockage/utilisation des données, en section 10.3.8.

— Pour l'outil de vérification, CASTT :

- implémenter la traduction de la propriété de *conformité à la nécessité des données* pour IAT, présenté en section 10.3.9 ;
- implémenter le choix des propriétés dans les mécanismes de traduction, présenté en section 10.3.10 ;
- utiliser MetACSL pour la traduction des propriétés pour Frama-C, en section 10.3.11 ;
- implémenter un mécanisme de traduction vers JML et utiliser des outils de vérification associés, en section 10.3.12.

Le niveau de détails donné plus bas, est influencé par l'état d'avancement de ces idées. Pour chacune de ces perspectives, j'identifie la limite actuelle, l'idée pour dépasser celle-ci, puis je détaille les pistes de mise en œuvre pour réaliser cette idée.

10.3.1 . Application de l'approche sur des exemples réels

Limite : actuellement les exemples considérés sont assez simples. Même si des versions complexes sont définies, et que des mécanismes de génération ont été implémentés afin de réaliser des tests de charge, elles ne représentent pas des exemples réels.

Idée : l'idée serait d'appliquer mon approche, composée de CSpeL et de CASTT, sur des exemples réels afin de vérifier si elle est immédiatement applicable ou si des changements sont nécessaires.

Pistes de mise en œuvre : le principe serait d'avoir accès à des systèmes réels, à la fois au modèle qui a été défini lors de la phase de conception, et au programme qui a été implémenté. Ces systèmes doivent traiter des données personnelles et utiliser des langages compatibles avec notre approche, pour le moment BPMN (ou

équivalent), en langage d'interaction compréhensible par IAT, ou en C. Si l'on veut considérer des systèmes réels mais dans d'autres langages, il faudrait alors étudier quelles seraient les modifications à apporter. Ces modifications ne devraient être nécessaires que pour le mécanisme de traduction ; le mécanisme de trace étant générique, ne devrait pas être impacté.

10.3.2 . Application de l'approche à un autre domaine - PPRL

Limite : actuellement l'approche a été évaluée sur des exemples concernant deux domaines d'application : les systèmes informatiques des hôpitaux et les sites internets. Il serait intéressant de voir si l'approche pourrait être utilisée dans d'autres domaines et d'identifier si des modifications sont nécessaires. Un de ces domaines, qui semble intéressant, est le *Privacy Preserving Record Linkage* (PPRL). Le *Record Linkage* est un processus permettant d'identifier quels enregistrements dans deux ou plus bases de données correspondent à la même entité. Le PPRL a pour but de garantir que la vie privée et la confidentialité des données sensibles sont respectées lorsque des bases de données sont liées entre organisations [174]. Ce domaine serait intéressant car, contrairement aux domaines d'application étudiés, il a une architecture type. Ainsi, les exemples réels devraient être fondés sur la même architecture, ce qui permettrait d'étudier une architecture réaliste, même sans avoir accès à des vrais systèmes. Cette architecture est illustrée sur la figure 10.1. Elle représente la liaison entre deux bases de données (*Database A* et *Database B*). Les données (di) sont ensuite traitées par une série de processus (Pi), tels que l'indexage, la comparaison et la classification. Ces données sont chiffrées (*encoded/encrypted*) pour être transmises entre ces processus. Le dernier processus permet d'évaluer si les données provenant des deux bases différentes font référence à la même entité.

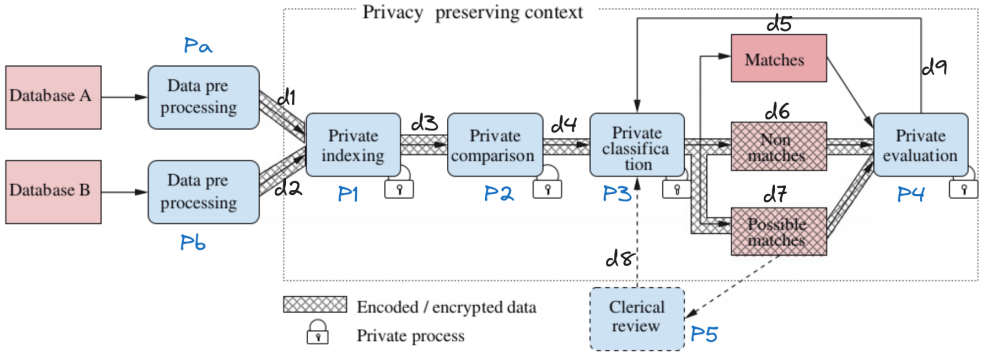


Figure 10.1 – Processus général de *privacy-preserving record linkage* [174] annoté.

Idée : l'idée serait d'identifier une propriété en lien avec ce transfert de données et de la formaliser en utilisant le formalisme de CSpeL. Le but ne serait pas de

proposer une technique issue des mesures classiques de sécurité déjà proposées dans la littérature (tels que la cryptographie ou la *differential privacy*) [174], mais de proposer une approche complémentaire pour s'assurer que les processus ont le droit de traiter les données.

Pistes de mise en œuvre : R. Hall et S. Fienberg [69] indiquent que lorsque deux parties intègrent leur données personnelles, elles n'ont besoin, en résultat, que de l'ensemble des enregistrements qui correspondent à la même entité. Ainsi, d'après eux, le but est de réaliser le *record linkage* sans révéler une quelconque information au sujet des enregistrements non liées. Pour utiliser mon approche dans ce contexte, une première proposition de propriété à vérifier serait : « un processus ne peut traiter une donnée que s'il a le droit d'y accéder ». Par exemple, si on reprend la figure précédente (la figure 10.1), $P1$ a le droit de traiter $d1$ mais pas $d6$. De même $P4$ à le droit de traiter $d5$, $d6$, $d7$ mais pas $d4$. Et seul $d5$ peut être accessible depuis l'extérieur (*i.e.* ceux qui veulent avoir le résultat en liant ces bases de données). Des discussions avec Dinusha Vatsalan de l'université de Macquarie (Australie) ont été initiés sur ce projet en vue d'une collaboration.

10.3.3 . Ajout d'une fonctionnalité de vérification du raffinement

Limite : actuellement une hypothèse forte de mon approche est que le raffinement réalisé entre les différents niveaux d'abstraction est correct. Je ne le vérifie cependant pas, or il est en pratique possible que ce raffinement soit erroné.

Idée : l'idée est de comparer formellement le système spécifié à différents niveaux d'abstraction, afin d'identifier s'ils ont le même comportement.

Pistes de mise en œuvre : pour identifier si le système a un comportement stable durant le processus de raffinement, l'utilisateur fournirait un système spécifié à un niveau d'abstraction (par exemple, au niveau Modèle Graphique) à l'aide d'un ensemble de fichiers CSpeL contenant des spécifications du contexte et des traces d'execution. Il fournirait également un ensemble de fichiers contenant ces spécifications à un autre niveau d'abstraction (par exemple, au niveau Programme). Ces ensembles de fichiers représenteraient ainsi les mêmes comportements du système mais à ces différents niveaux. Ensuite CASTT évaluerait ces traces. Ainsi, si les traces représentant les mêmes traitements sont évaluées avec le même verdict, alors le raffinement est considéré comme correct.

10.3.4 . Enrichissement du langage - Définitions personnalisable des propriétés

Limite : actuellement les propriétés vérifiées sont fixes. Il n'est pas possible pour l'utilisateur de définir lui même la propriété à vérifier. Les deux propriétés liées au consentement, la *conformité aux finalités consenties* et la *conformité à la nécessité des données*, sont fixées par mon approche (ainsi que leur traduction pour pouvoir être vérifiées par IAT ou par Frama-C).

Idée : l'idée serait que l'utilisateur puisse spécifier la propriété à vérifier en utilisant une syntaxe proche de la logique tout en faisant référence aux éléments

définis dans le fichier CSpeL.

Pistes de mise en œuvre : dans un premier temps, on pourrait limiter l'utilisateur à un sous-ensemble de possibilités d'écriture permettant de spécifier les propriétés déjà vérifiées actuellement. Cette fonctionnalité de spécification personnalisable des propriétés impliquerait de grands changements, tant au niveau langage qu'au niveau implémentation dans l'outil. Il faudrait étudier l'impact au niveau des différents mécanismes (vérification de trace et traduction pour IAT et pour Frama-C). Cependant, la traduction pour Frama-C peut être simplifiée par le fait que le mécanisme de traduction de CASTT repose sur ACSL qui est lui même un langage reposant sur une syntaxe proche de la logique (contenant des prédicats, conjonctions, implications, quantificateurs, ...).

10.3.5 . Enrichissement du langage - Données non personnelles

Limite : actuellement, lors de la génération de traces, les évènements possibles composant la trace se réfèrent à la spécification donnée en CSpeL. Or, dans cette spécification, seules les données personnelles sont indiquées, car les données non personnelles ne sont pas nécessaires à la vérification. De plus, leur énumération exhaustive serait fastidieuse (ensemble possiblement beaucoup plus grand que l'ensemble des données personnelles en fonction des cas).

Idée : l'idée serait de permettre également à l'utilisateur de spécifier en CSpeL des données non personnelles. L'ensemble défini ainsi ne serait pas forcément exhaustif, et elles ne seraient pas utilisées dans la vérification des propriétés, mais elles seraient utilisées pour générer des traces d'exécution plus variées.

Pistes de mise en œuvre : le principe serait d'étendre le langage CSpeL pour permettre à l'utilisateur de spécifier des données non personnelles pouvant être présentes dans les traces d'exécution. Pour cela, il faudrait étendre la grammaire, la syntaxe mais également le mécanisme de génération des traces, afin de spécifier l'ensemble supplémentaire pour les données non personnelles pouvant être présentes dans les traces.

10.3.6 . Interface utilisateur d'édition pour CSpeL

Limite : actuellement, les fichiers CSpeL sont écrits manuellement à l'aide d'un éditeur de texte. Cela peut se révéler fastidieux et source d'erreurs, notamment à cause des relations entre les différents éléments.

Idée : l'idée serait d'implémenter une interface graphique afin d'aider l'utilisateur à remplir la spécification, notamment en proposant des éléments déjà préremplis.

Pistes de mise en œuvre : l'utilisateur pourrait simplement remplir les ensembles de base (données personnelles, processus, finalités) via l'interface graphique (sans se préoccuper des mots clés, parenthèses ou accolades). Ensuite, lors de la spécification des fonctions d'association (consentement, finalités des processus et données nécessaires au traitement), l'outil pourrait proposer des valeurs venant de ces ensembles. Ainsi, l'utilisateur n'aurait qu'à sélectionner les valeurs

voulues. À ce stade, aucun langage de programmation n'a encore été identifié pour réaliser l'interface, et aucune définition de l'interface (maquette) n'a été définie.

10.3.7 . Vérification d'une propriété d'évolution de consentement

Limite : actuellement, deux propriétés liées au traitement des données personnelles sont définies et vérifiées dans mon approche : la propriété de *conformité aux finalités consenties* et la propriété de *conformité à la nécessité des données*. Cependant, d'autres notions existantes liées à ce traitement ne sont pas prises en compte dans ces propriétés, notamment l'évolution du consentement (liée au droit de rétractation de la personne concernée (*Data Subject*)).

Idée : l'idée serait de définir ces propriétés formellement et d'implémenter leur vérification.

Pistes de mise en œuvre : le principe de cette propriété serait de considérer que le consentement de la personne concernée peut évoluer dans le temps, c'est-à-dire que la personne concernée peut consentir dans un premier temps à certaines finalités de traitements, puis changer d'avis et se rétracter, et éventuellement par la suite consentir à nouveau. Pour cela, il faut considérer des évènements en plus des évènements existants (correspondant à des traitements de données). Ces évènements correspondraient à des consentements (*Grant*) ou des rétractations (*Withdraw*). Ces évènements influeraient non pas sur la vérification des propriétés mais sur la valeur des éléments du contexte (ce qui pourrait faire changer la validité des traces). Ainsi seules les règles d'inférences sont modifiées.

$$\frac{\mathcal{C}[\gamma(d,p) \leftarrow \top] \vdash_{prop} T}{\mathcal{C} \vdash_{prop} Grant(d,p);T} \qquad \frac{\mathcal{C}[\gamma(d,p) \leftarrow \bot] \vdash_{prop} T}{\mathcal{C} \vdash_{prop} Withdraw(d,p);T}$$

avec $prop \in \{P; N\}$

Figure 10.2 – Règle pour la mise à jour du consentement.

Pour cela il faudrait ajouter les règles présentées sur la figure 10.2.

La première règle signifie que « l'évènement correspondant au consentement d'une donnée d pour la finalité p nécessite que le reste de la trace soit conforme selon ce nouveau consentement ». Plus précisément, lorsque l'évènement correspondant au consentement ($Grant(d,p)$) est rencontré dans la trace, cela met à jour dans le contexte ($\mathcal{C}$) la valeur retournée par la fonction de consentement pour cette donnée et cette finalité (c'est-à-dire que désormais $\gamma(d,p) \rightarrow \top$). Ainsi, la suite de l'évaluation de la trace se fera avec cette nouvelle valeur de consentement.

La seconde règle signifie que « l'évènement correspondant au retrait du consentement d'une donnée d pour la finalité p nécessite que le reste de la trace soit conforme selon ce nouveau consentement ». Plus précisément, lorsque l'évènement correspondant au retrait du consentement ($Withdraw(d,p)$) est rencontré dans la trace, cela met à jour dans le contexte ($\mathcal{C}$) la valeur retournée par la fonction de consentement pour cette donnée et cette finalité (C'est-à-dire que désormais

$\gamma(d, p) \rightarrow \bot$). Ainsi, la suite de l'évaluation de la trace se fera avec cette nouvelle valeur de consentement.

Il reste nécessaire d'en évaluer davantage l'impact sur des exemples avant de pouvoir les implémenter dans le mécanisme de vérification de trace (ce qui semble relativement simple). De plus cela soulève des questions concernant le mécanisme de traduction. En effet, le consentement ne faisant pas partie du système, la question de la correspondence entre ces évènements et le programme (ou le modèle formel) du système demeure ouverte.

10.3.8 . Vérification d'une propriété de durée de conservation et d'utilisation des données

Limite : la limite est la même que pour la propriété précédente (en section 10.3.7) : des notions existantes liées au traitement des données personnelles ne sont pas prises en compte dans les propriétés que j'ai formalisées dans ma thèse, notamment la durée autorisée pour le stockage et l'utilisation des données personnelles.

Idée : de même que précédemment, l'idée serait de définir ces propriétés formellement et d'implémenter leur vérification.

Pistes de mise en œuvre : le principe de cette propriété est de considérer que les données sont collectées et utilisées à différents moments, c'est-à-dire que les traces auraient une notion de temporalité, tout comme le consentement. Pour cela il faut considérer des évènements en plus des évènements existants (correspondant à des traitements de données, ou au consentement si on l'ajoute). Ces évènements correspondraient à la collecte de données personnelles (*Collect*), qui devrait être associée à une temporalité. De plus les événements déjà existants devraient être modifiés pour intégrer des notions de temporalité (afin d'indiquer à quel moment survient l'évènement). Une nouvelle propriété resterait alors définie et vérifiée. Ensuite, au moins deux possibilités existent :

— ne pas ajouter d'éléments dans le contexte CSpeL mais modifier ceux existant pour qu'ils prennent en compte des notions de durée consentie ; ou

— ajouter une nouvelle association représentant la durée autorisée pour le traitement de la donnée personnelle (par exemple sous la forme $DurationOf(data) \rightarrow duration$).

Je n'ai pas encore identifié quelle solution serait la plus pertinente. À noter qu'à l'heure actuelle, il n'y a, à ma connaissance, pas de mécanisme de temporalité dans IAT ou Frama-C.

10.3.9 . Traduction de la propriété de *conformité à la nécessité des données* pour IAT

Limite : actuellement j'ai défini deux propriétés liées au consentement, à savoir la *conformité aux finalités consenties* et la *conformité à la nécessité des données*, qui sont vérifiables à l'aide de CASTT à partir d'une spécification CSpeL. Ces

deux propriétés sont vérifiables pour des traces spécifiées en CSpeL, ainsi que pour des programmes C à l'aide du mécanisme de traduction implémenté dans CASTT. Par contre, ce mécanisme de traduction pour IAT n'est implémenté que pour la première propriété, la *conformité aux finalités consenties*.

Idée : l'idée serait d'implémenter une traduction équivalente à celle existante pour la *conformité aux finalités consenties* (décrite en section 9.3.1) pour cette autre propriété.

Pistes de mise en œuvre : le principe serait d'ajouter automatiquement dans le modèle (fichier .imf) des gardes lors des échanges de messages lorsque ces échanges contiennent une variable considérée comme donnée personnelle. Par exemple, supposons que le modèle d'origine contient l'échange suivant : $\texttt{l}_i$ `--` `m(v)` $\rightarrow$ $\texttt{l}_j$. Si la variable v est une donnée personnelle (c'est-à-dire si elle est spécifiée dans les `\personalData` du fichier CSpeL), alors une garde serait ajoutée à l'envoi du message afin de garantir que le message n'est envoyé que si la garde est respectée, c'est-à-dire que la propriété *conformité à la nécessité des données* est satisfaite. On générerait alors des gardes booléennes spécifiques supplémentaires, de la forme `donnée_processus` (par exemple, v_l_i), qui seraient à ajouter dans le bloc des variables (@variables) et à initialiser (dans le bloc @init). Dans ce cas là, l'échange de message modifié serait : l_i`[v_`l_i`]` `--` `m(v)` $\rightarrow$ l_j.

10.3.10 . Implémentation du choix des propriétés dans les mécanismes de traduction

Limite : actuellement, l'utilisateur peut spécifier quelles propriétés parmi celles définies en section 8.1.3 sont à vérifier (la *conformité aux finalités consenties*, la *conformité à la nécessité des données*, l'une ou l'autre, ou la conjonction des deux). Ce choix n'est pris en compte que pour le mécanisme de vérification de trace. Il n'est pas encore possible d'effectuer un choix dans les propriétés à vérifier pour le mécanisme de traduction. En effet, la *conformité aux finalités consenties* et la *conformité à la nécessité des données*, sont prises en compte toutes les deux pour la traduction vers ACSL pour utiliser Frama-C (fonctionnalité décrite en section 9.4). Le mécanisme de traduction pour IAT, quant à lui, traduit seulement la *conformité aux finalités consenties* (fonctionnalité décrite en section 9.3.1), la seconde propriété n'étant pas encore implémentée.

Idée : l'idée' serait dans un premier temps de laisser l'utilisateur choisir quelles propriétés sont à traduire (et donc à vérifier), comme cela est possible pour la vérification de trace (décrit en section 10.3.10).

Pistes de mise en œuvre : la spécification est déjà possible dans le fichier CSpeL. L'idée serait, lors du mécanisme de traduction, de vérifier quelles propriétés ont été choisies (la *conformité aux finalités consenties*, la *conformité à la nécessité des données*, l'une ou l'autre, ou la conjonction des deux). Il serait alors nécessaire de séparer la génération d'ACSL selon les propriétés et ajouter des conditions (quand une donnée personnelle est utilisée, les assertions générées correspondent aux propriétés choisies).

10.3.11 . Utilisation de MetACSL

Limite : actuellement, une identification de traitement de données personnelles dans le programme C est implémentée dans CASTT, pour générer une annotation ACSL pour vérifier mes propriétés (comme détaillé en section 9.4). Cette identification n'est probablement pas exhaustive, car elle se fonde sur les cas présents dans les exemples définis en section 7.

Idée : l'idée serait d'utiliser MetACSL, le plug-in de Frama-C permettant de générer des annotations ACSL, pouvant identifier le type des variables traitées (lues ou écrites), afin de générer des annotations aux endroits voulus de façon automatique.

Pistes de mise en œuvre : pour cela, au lieu de générer des annotations ACSL en parcourant le programme C pour identifier les traitements de données personnelles, comme c'est actuellement le cas, CASTT pourrait générer des *métapropriétés* MetACSL [141] (aussi appelées *HILARES*) avec des gardes selon les types de données. Une HILARE permet de générer des annotations ACSL selon certains critères, tels que la lecture ou l'écriture d'une variable et selon son type. Ainsi, le mécanisme de traduction de CASTT pourrait générer simplement des HILARES pour chaque type de donnée personnelle identifiée, et déléguer l'identification du traitement de donnée personnelle et l'insertion de l'annotation à MetACSL. Remarque : cette possibilité avait été envisagée lors de l'implémentation du mécanisme de traduction pour Frama-C mais n'avait pas pu être mise en place à cause d'une limite de MetACSL. À ce moment là, MetACSL pouvait identifier le type des variables traitées mais cette identification ne permettait pas d'identifier l'accès à un champ d'une structure correspondant à une donnée personnelle. Par exemple la lecture `int x = patient.id`, avec `patient` de type `Patient`(identifié comme donnée personnelle), n'était pas détectée. Elle n'était pas détectée car le champ lu, `id`, de la structure est un entier (le type entier n'étant pas par nature une donnée personnelle), la lecture n'était donc pas identifiée par MetACSL comme étant la lecture d'une donnée personnelle. Cependant pour traduire mes propriétés cette identification devrait être possible. C'est pour cette raison que je n'avais pas pu utiliser MetACSL. Actuellement, la limite de MetACSL a été levée. Il serait donc possible à court terme de s'appuyer sur MetACSL pour la génération d'annotations ACSL.

10.3.12 . Traduction en JML et utilisation d'outils de vérification associés

Limite : actuellement le mécanisme de traduction pour un outil au niveau programme est réalisée pour permettre une vérification à l'aide de Frama-C. Ainsi, la spécification CSpeL et les propriétés sont traduites en code ghost C et en annotations ACSL. Ceci limite les exemples au niveau programme à des exemples écrits en C.

Idée : l'idée serait de définir et d'implémenter un mécanisme de traduction

équivalent pour le langage JML [97] (un langage de spécification de programme Java) et d'utiliser des outils de vérification existants tels que OpenJML [3] (outil de vérification permettant la vérification statique et dynamique). J'ai choisi ce langage cible car il est très proche de ACSL : le mécanisme de traduction devrait donc être relativement similaire à celui implémenté pour utiliser Frama-C.

Pistes de mise en œuvre : le langage JML semblant proche du langage ACSL (tant au niveau syntaxique qu'au niveau sémantique), il faudrait générer des annotations équivalentes pour des programmes Java, et utiliser ensuite un outil de vérification permettant de vérifier nos propriétés sur des programmes. Bien que n'ayant pas encore utilisé OpenJML, l'outil permet de réaliser des analyses statiques et dynamiques (s'appuyant sur les même principes que les plug-ins WP et E-ACSL de Frama-C). La traduction en elle même devrait être simple, la correspondance étant déjà établie pour ACSL et les deux langages étant proches. La difficulté majeure repose sur le fait qu'actuellement CASTT est un plug-in de Frama-C et qu'à ce titre, il bénéficie de facilités pour parser un programme C en un arbre syntaxique et le parcourir afin de le modifier. Cette facilité a été un gros avantage pour le développement du mécanisme de traduction pour Frama-C. En revanche, comme Frama-C est conçu pour vérifier du code C, il n'offre pas ce genre de possibilités pour Java. Il faudrait donc soit implémenter une fonctionnalité équivalente pour Java dans mon outil CASTT, soit utiliser un outil qui offre cette fonctionnalité et implémenter un mécanisme de communication entre cet outil et le mien. À l'heure actuelle, je n'ai encore pas identifié quelle solution serait la plus pertinente.

10.4 . Conclusion

Lors de la classification relative à la vie privée, dans le chapitre I, GePyR a été défini pour classer les différentes notions de vie privée de l'état de l'art dans des catégories génériques. Pour la partie vérification, je me suis intéressée aux notions de la catégorie *caractérisation du Traitement* de GePyR. En étudiant cette catégorie, et les articles classés dans cette catégorie, j'ai identifié quatre notions importantes à prendre en compte : la finalité des traitements (*purpose*), la nécessité des données personnelles pour les traitements (*necessity*), la durée autorisée pour le traitement des données personnelles (*duration*), la possibilité pour la personne concernée (*Data Subject*) d'accorder ou de retirer son consentement à tout moment (*consent evolution*) En me basant sur ces notions, j'ai pu définir plusieurs propriétés d'intérêt à vérifier.

De façon complémentaire, à l'aide de PyCO, en me focalisant sur le consentement de la personne concernée (*datasubject Consent*) et les éléments lui étant reliés, ainsi que les différents types de relations existantes, j'ai pu identifier les éléments à prendre en compte pour la vérification de ces propriétés.

De plus, en établissant la classification, j'ai identifié différents niveaux d'abs-

3. https://www.openjml.org/

traction pour les systèmes considérés dans l'état de l'art correspondant à des étapes de développement dans le cycle de vie d'un logiciel, notamment ceux utilisés pour les approches formelles : le niveau modèle (utilisant des langages de modélisation) et le niveau programme (utilisant des langages de programmation).

Le but de mes travaux dans cette thèse est d'aider à améliorer le respect de la vie privée. Celui de mes contributions liées à la partie vérification de la vie privée, est de verifier formellement des propriétés de consentement à différents niveaux d'abstraction.

Pour cela, j'ai défini CSpeL, un langage formel permettant à un utilisateur de spécifier un système dans un contexte de vie privée afin de vérifier qu'il respecte des propriétés de consentement. L'utilisateur spécifie différents éléments nécessaires à la vérification, tels que les processus du système, les données personnelles et les finalités de traitement. L'utilisateur spécifie également le consentement accordé par la personne concernée (*Data Subject*), quelles finalités sont associées à chaque processus, et quelles données personnelles sont nécessaires pour l'exécution de chaque processus. L'utilisateur peut également fournir une trace d'exécution, contenant différents traitements, afin de vérifier si cette trace respecte les propriétés de vie privée, selon le contexte spécifié. Les propriétés que j'ai définies dans un premier temps sont : la propriété de *conformité aux finalités consenties*, et la propriété de *conformité à la nécessité des données*. La première formalise le fait que lorsqu'un processus traite une donnée personnelle, la personne concernée (*Data Subject*) doit avoir consenti pour au moins une des finalités de traitement associée à ce processus. La seconde formalise le fait que lorsqu'un processus traite une donnée personnelle, ce processus avait besoin de traiter cette donnée. Un point important de CSpeL est que l'utilisateur peut l'utiliser au niveau modèle mais également au niveau programme et ce, sans à avoir à changer de langage. L'utilisateur peut simplement changer la spécification pour les éléments qui ont été raffinés.

En plus de ce langage, j'ai défini l'outil associé CASTT, permettant de vérifier les propriétés mentionnées ci-dessus. Cet outil permet de vérifier qu'un système respecte les propriétés de *conformité aux finalités consenties* et de *conformité à la nécessité des données*, selon une spécification donnée en CSpeL. Pour cette vérification trois fonctionnalités sont implémentées actuellement :

— L'utilisateur peut spécifier une trace d'exécution du système également à l'aide de CSpeL, ainsi CASTT vérifie si cette trace respecte les propriétés ;

— L'utilisateur peut à la place, au niveau Modèle Formel, fournir le système sous la forme de fichiers .imf et .mtf et demander à CASTT de traduire la spécification CSpeL, ainsi que la propriété de *conformité aux finalités consenties*, pour modifier ces fichiers. Ces nouveaux fichiers peuvent alors être utilisés par IAT pour vérifier que le système respecte cette propriété ;

— L'utilisateur peut, également au niveau Programme, fournir le système sous la forme d'un programme C et demander à CASTT, de traduire la spécification CSpeL, ainsi que les propriétés, en annotation ACSL (et en code *ghost*).

Ce fichier annoté pouvant être utilisé par différents plugins de Frama-C (WP, E-ACSL et Eva) pour vérifier que le système respecte ces propriétés.

Bien que plusieurs approches aient été proposées pour améliorer le respect de la vie privée lors du traitement de données personnelles dans un système, aucun des travaux identifiés dans la littérature ne propose de solutions pour vérifier des propriétés formelles liées au traitement des données personnelles à la fois au niveau modèle et au niveau programme.

Mes contributions répondent à ce besoin mais ont certaines limitations. Ces limitations peuvent être dues à des choix que j'ai faits. Par exemple, je m'intéresse aux notions liées à la catégorie de *caractérisation du Traitement* et pas à celles de *caractérisation de la Visibilité*. Autre exemple, j'ai fait le choix que les notions de vie privées ne soient pas incluses dans le système. De plus, une hypothèse forte est que la spécification fournie par l'utilisateur est correcte. Enfin, j'ai défini moi-même les exemples qui sont utilisés et qui ne représentent pas la complexité d'un système réel.

J'ai identifié plusieurs suites de travaux possibles : traduire la propriété de *conformité à la nécessité des données* pour IAT, permettre à l'utilisateur de choisir ou de définir lui-même les propriétés à vérifier en étendant le langage CSpeL, définir et vérifier d'autres propriétés de consentement, d'utiliser MetACSL pour la génération d'annotations ACSL et également implémenter un mécanisme de traduction pour un autre outil au niveau Programme (par exemple OpenJML) afin de permettre d'étendre les exemples possibles (car étant écrits en Java).

Ces travaux pour la vérification de la vie privée constituent une première étape dans la vérification formelle de propriétés de consentement pour un système à différents niveaux d'abstraction, et complètent ainsi l'ensemble des solutions existantes permettant d'améliorer le respect de la vie privée.

11 - CONCLUSION ET SUITE DES TRAVAUX

11.1 . Conclusion

Il est important de vérifier qu'un système respecte la vie privée mais cela est complexe pour plusieurs raisons. D'une part, la vie privée est une notion complexe qui varie par elle-même selon les points de vue des parties prenantes. D'autre part, la vérification d'un système est complexe. Plusieurs types de méthodes et de techniques de vérification existent dans la littérature, notamment les méthodes liées au cycle de vie d'un logiciel et les méthodes formelles pour la vérification.

Dans ce contexte, l'objectif de cette thèse a été de participer à l'amélioration du respect de la vie privée en informatique, et ce en proposant une solution qui permette de vérifier formellement qu'un système respecte la vie privée. Cette solution devait pouvoir être appliquée au niveau Modèle et au niveau Programme, ces niveaux correspondant à des étapes de développement du système.

J'ai divisé cet objectif en deux problématiques : qu'est-ce que la vie privée en informatique et comment vérifier formellement qu'un système la respecte ? Ainsi, mes contributions ont été divisées en deux parties : une partie classification et une partie vérification. Chacune de ces parties répond à une de ces problématiques.

11.1.1 . Partie classification

Cette partie a permis de répondre à la première problématique : qu'est-ce que la vie privée en informatique ? Dans la littérature, un grand nombre d'articles traitent du respect de la vie privée mais n'ont pas de structure commune. Chacun adopte un certain point de vue, une façon de représenter la vie privée, et ce, sans qu'il y ait de liens évidents entre les différentes représentations. Ainsi, il est difficile de classer les différentes approches afin d'identifier les solutions comparables, c'est-à-dire des solutions répondant à des problématiques similaires. Pour répondre à ce besoin, j'ai réalisé les contributions suivantes :

Représentations de la vie privée - À partir d'une analyse de l'état de l'art, j'ai identifié différents types de représentations existantes de la vie privée en informatique. Il en existe six : la représentation *via* des Principes, celle *via* des Propriétés, ou bien encore celle *via* des Buts, mais également les représentations *via* des Menaces, *via* des Activités Nocives ou *via* des Vulnérabilités. À partir de cette identification des différentes représentations existantes, j'ai pu classer dans une SMS (*Systematic Mapping Study*) un certain nombre d'articles proposant une solution pour améliorer le respect de la vie privée selon le type de représentation considéré. Cependant, cela ne suffit pas pour déterminer si ces articles répondent à des problématiques proches reposant sur des notions de vie privée similaires.

GePyR - J'ai aussi défini une taxonomie constituée de catégories reposant sur des notions génériques liées à la vie privée : les catégories de *caractérisation du Traitement* (P), *de la Visibilité* (V), *de la Transparence* (T), *de l'Exactitude des données* (D), et celle *de la Responsabilité* (A). Cette taxonomie permet de regrouper les représentations précédentes dans ces catégories, selon les notions qu'elles considèrent. J'ai utilisé GePyR et ces représentations dans la SMS que j'ai réalisée afin d'étudier et classer 79 articles.

PyCO - Je me suis également fondée sur un sous-ensemble des catégories précédentes (P,T et A) pour définir une ontologie permettant d'identifier les éléments à prendre en compte pour vérifier qu'un système respecte la vie privée et leurs relations. Ces éléments sont : les Données personnelles, la Personne concernée, le Responsable du traitement, le Sous-traitant, le Système et ses processus, l'Utilisateur, les Finalités de traitement, les Durées de conservation, le Consentement de la personne concernée, la Notice et l'Autorité de contrôle. Les types de liens identifiés sont : des activités entre des entités, des activités de définition et de traitement, et également des relations de composition et d'autorité. J'ai appliqué cette ontologie sur des articles concernant différents domaines d'application, et à différents niveaux d'abstraction, afin d'identifier comment ces éléments étaient représentés dans ces articles. Cette ontologie et cette identification des représentations des éléments ont servi de support pour la définition du langage formel de la partie vérification.

11.1.2 . Partie vérification

Cette partie a permis de répondre à la seconde problématique : comment vérifier formellement qu'un système respecte la vie privée ? Pour cela, il a fallu pouvoir formaliser un système et les notions de vie privée nécessaires à la vérification. Il a fallu également définir les propriétés que le système doit respecter dans le but d'utiliser des méthodes formelles. De plus, durant son cycle de vie, un système change de formalisme. Par exemple, au niveau Modèle, il sera modélisé à l'aide d'un langage de modélisation (tel que BPMN), tandis qu'au niveau Programme, il sera écrit à l'aide d'un langage de programmation (tel que C). Il a donc fallu pouvoir s'adapter à ces différents formalismes pour pouvoir vérifier le système tout au long de son cycle de vie. Pour répondre à ces besoins, j'ai ainsi réalisé les contributions suivantes :

CSpeL - Dans un premier temps, j'ai défini le langage CSpeL (Context Specification Language) qui permet à l'utilisateur de spécifier formellement le système et le contexte de vie privée associé, c'est-à-dire les éléments à prendre en compte pour vérifier le respect de la vie privée. Pour ce faire, je me suis restreinte aux notions liées à la catégorie *caractérisation du Traitement* définie dans GePyR. L'utilisateur de CSpeL va spécifier le contenu d'un modèle formel en renseignant : les Processus du système, les Données personnelles qui sont traitées, les Finalités de traitement, le Consentement (pour quelles finalités le consentement au traitement

des données a été accordé), la Signification des traitements (c'est-à-dire pour les finalités pour lesquelles a été défini chaque processus), et la Nécessité des données (c'est-à-dire les données nécessaires à chacun des processus). En plus de ce modèle formel, CSpeL permet également à l'utilisateur de spécifier une trace d'exécution à vérifier. Cette trace représente une séquence d'évènements, où chaque évènement correspond au traitement d'une donnée par un processus. Tous les éléments nécessaires à la vérification peuvent donc être spécifiés formellement. Ce langage est suffisamment générique pour permettre de spécifier un système au niveau Modèle mais également au niveau Programme.

Formalisation de propriétés de vie privée - Pour vérifier que les traces d'exécution spécifiées en CSpeL respectent la vie privée selon le contexte spécifié en CSpeL, j'ai formalisé les propriétés suivantes de respect de la vie privée : la propriété de *conformité aux finalités consenties*, qui assure qu'un processus ne traite une donnée personnelle que pour une finalité de traitement consentie, et la propriété de *conformité à la nécessité des données*, qui assure qu'un processus ne traite une donnée personnelle que si cela lui est nécessaire. Ces propriétés faisant référence aux éléments spécifiés à l'aide de CSpeL, elles ne sont pas impactées par le niveau d'abstraction du système. En effet, ces propriétés peuvent être vérifiées sur des systèmes au niveau Modèle comme au niveau Programme. Des exemples de preuves manuelles ont été donnés dans ce manuscrit.

CASTT - Vérifier manuellement le respect des propriétés précédentes peut être fastidieux et source d'erreurs. J'ai donc implémenté l'outil CASTT qui permet de vérifier automatiquement qu'un système respecte ces propriétés. Actuellement CASTT a trois mécanismes de vérification : un mécanisme de vérification de trace, un mécanisme de traduction pour un outil au niveau Modèle, et un mécanisme de traduction pour un outil au niveau Programme.

Pour utiliser le premier mécanisme, l'utilisateur spécifie un contexte en CSpeL et une trace d'exécution à vérifier. À partir de ces éléments, l'outil vérifie que la trace respecte les propriétés précédentes de *conformité aux finalités consenties* et de *conformité à la nécessité des données*. Pour l'évaluation, j'ai testé l'outil sur des exemples issus de différents domaines d'application, à la fois au niveau Modèle et au niveau Programme. Cela m'a permis de vérifier que la validité ou l'invalidité des traces était correctement détectée. Les traces spécifiées (manuellement) de façon à être valides étaient bien détectées comme valides, et réciproquement pour les traces invalides. J'ai également implémenté un générateur de traces afin d'évaluer mon outil sur des traces composées d'un grand nombre d'évènements, allant jusqu'à un million d'évènements. Cette vérification prenait moins de quatre secondes et le temps de vérification était linéaire par rapport à la taille de la trace.

L'objectif des mécanismes de traduction est de ne plus avoir besoin de spécifier une trace à vérifier mais de vérifier directement le système. L'utilisateur spécifie alors un contexte en CSpeL et fournit le fichier source contenant le système à

CASTT. Mon outil traduit ensuite cette spécification et les propriétés à vérifier dans un formalisme compréhensible par un outil spécifique de vérification correspondant au niveau d'abstraction du système, et génère un nouveau fichier correspondant au fichier source modifié selon cette traduction. De cette façon, le fichier généré peut être utilisé par l'outil spécifique de vérification pour vérifier le respect des propriétés. Pour le mécanisme de traduction vers un outil au niveau Modèle, l'outil sélectionné est IAT, tandis que pour le niveau Programme, l'outil sélectionné est Frama-C. Le mécanisme de traduction pour le niveau Modèle a été évalué sur un petit ensemble d'exemples simples, afin de vérifier si les résultats obtenus correspondaient bien aux verdicts attendus. Quant au mécanisme de traduction pour le niveau Programme, il a été évalué selon le même principe, mais j'ai également implémenté un générateur d'appels de fonction C afin de pouvoir évaluer le mécanisme sur des programmes de grande taille (allant jusqu'à cent mille appels de fonctions). Pour cette évaluation, je me suis intéressée au surcoût généré par la vérification des propriétés de respect de la vie privée précédentes. J'ai pu observer que ce surcoût était négligeable, et que plus le programme était grand, plus ce surcoût diminuait.

11.2 . Suite des travaux

Plusieurs travaux futurs sont détaillés dans la section 6.3 pour la partie classification, et dans la section 10.3 pour la partie vérification. Je les résume ici, mais plus de détails sont disponibles dans ces sections.

11.2.1 . Partie classification

Pour la partie classification, des travaux futurs ont été identifiés pour chacune des contributions. Pour les représentations de la vie privée, il serait intéressant d'étudier d'autres lois et règlements que le RGPD et d'observer les changements que cela impliquerait dans les types de représentations identifiées.

Pour la taxonomie GePyR, il y a plusieurs pistes. Une première piste serait d'étendre l'application de la SMS réalisée en interrogeant d'autres bases de données ; ceci permettrait d'étudier des articles supplémentaires et de voir si cela change les observations obtenues avec la SMS déjà réalisée. Une autre piste serait d'implémenter un outil avec une interface graphique pour permettre à un utilisateur d'utiliser automatiquement et facilement la classification afin d'obtenir un sous-ensemble d'articles pertinents. Une dernière piste serait d'ajouter les catégories de GePyR dans un outil existant lié aux exigences (par exemple, MaatREQ [17]).

Pour l'ontologie PyCO, deux perspectives ont été étudiées. Une première serait la réalisation d'une SMS, de façon similaire à ce qui a été réalisé avec GePyR, mais en identifiant cette fois les éléments de PyCO. Une autre piste serait l'implémentation d'un outil avec interface graphique pour permettre à un utilisateur d'utiliser automatiquement et simplement PyCO pour identifier les représentations des éléments de l'ontologie dans un ensemble d'articles sélectionnés.

11.2.2 . Partie vérification

Pour la partie classification, des travaux futurs ont été identifiés pour chacune des contributions et également pour l'ensemble d'entre elles. Une piste serait d'évaluer ces contributions sur des exemples plus complexes, notamment des systèmes existants. Une autre serait d'utiliser cette approche dans le domaine des *Privacy-Preserving Record Linkage* [174] et d'identifier les extensions nécessaires à apporter aux travaux de cette thèse. Actuellement, une hypothèse forte a été établie pour mes contributions : le fait que le raffinement lors du passage entre les différents niveaux d'abstractions est correct. Une idée serait d'identifier et d'implémenter un mécanisme de vérification pour garantir que le raffinement est réellement correct.

Deux autres extensions seraient intéressantes pour le langage CSpeL : permettre à l'utilisateur de définir de façon personnalisée les propriétés à vérifier d'une part, et permettre de spécifier, en plus des données personnelles, des données non personnelles utilisées dans le système d'autre part.

Aussi actuellement, deux propriétés sont vérifiées à l'aide des contributions, il serait pertinent de s'intéresser à d'autres propriétés comme une propriété liée à l'évolution du consentement et une propriété concernant la durée d'utilisation des données personnelles.

Enfin, plusieurs perspectives ont été identifiées pour l'outil CASTT. Une première perspective serait d'implémenter la traduction de la propriété *conformité à la nécessité des données* pour l'outil IAT [109], comme cela a été fait pour Frama-C [19] (l'utilisateur peut actuellement choisir les propriétés à vérifier avec le mécanisme de vérification de trace). Une autre perspective serait de permettre ce choix également avec le mécanisme de traduction. Par ailleurs, la traduction de spécification CSpeL en annotations ACSL pour Frama-C est faite pour chaque traitement de donnée identifié dans le programme. Il serait possible de simplifier cette traduction en utilisant le plug-in MetACSL [142] qui, à partir d'une méta-propriété, générerait automatiquement les assertions aux endroits pertinents.

Bibliographie

[1] Abdul Ghafoor Abbasi et Zaheer Khan. “VeidBlock : Verifiable Identity Using Blockchain and Ledger in a Software Defined Network”. In : *Companion Proceedings of The10th International Conference on Utility and Cloud Computing*. Association for Computing Machinery, 2017.

[2] Jean-Raymond Abrial. *The B-Book, assigning programs to meaning*. Cambridge University Press, 1996.

[3] Amir Shayan Ahmadian, Daniel Strüber et Jan Jürjens. “Privacy-Enhanced System Design Modeling Based on Privacy Features”. In : *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. Association for Computing Machinery, 2019.

[4] Amirshayan Ahmadian. “Model-based privacy by design”. Thèse de doct. 2020.

[5] Hussain Mutlaq Alnajrani, Azah Anir Norman et Babiker Hussien Ahmed. “Privacy and data protection in mobile cloud computing : A systematic mapping study”. In : *Plos one* (2020).

[6] Saeed Ibrahim Alqahtani et Shujun Li. “PPAndroid-Benchmarker : Benchmarking Privacy Protection Systems on Android Devices”. In : *Proceedings of the 12th International Conference on Availability, Reliability and Security*. Association for Computing Machinery, 2017.

[7] Majed Alshammari et Andrew Simpson. “Privacy Architectural Strategies : An Approach for Achieving Various Levels of Privacy Protection”. In : *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*. Association for Computing Machinery, 2018.

[8] Aaesha Alzaabi, Ayesha Aldoobi, Deena Alnuaimi, Latifa Alserkal, Mahra Alsuwaidi et Nedal Ababneh. “Grid-Based Source Location Privacy Protection Schemes in IoT Wireless Sensor Networks”. In : *2021 4th International Conference on Data Storage and Data Engineering*. Association for Computing Machinery, 2021.

[9] Saule Amanzholova, Darya Akhmetova et Azhar Sagymbekova. “Development of a Web-Resources Testing System for Compliance with GDPR Regulation”. In : *The 7th International Conference on Engineering & MIS 2021*. Association for Computing Machinery, 2021.

[10] Mahmoud Ammar et Bruno Crispo. “Verify&Revive : Secure Detection and Recovery of Compromised Low-End Embedded Devices”. In : *Annual Computer Security Applications Conference*. Association for Computing Machinery, 2020.

[11] Annie I. Antón et Julia B. Earp. “A requirements taxonomy for reducing web site privacy vulnerabilities”. In : *Requirements engineering* (2004).

[12] Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers et Matthias Schunter. “Enterprise privacy authorization language (EPAL)”. In : *IBM Research* (2003).

[13] Australian Government. *The Privacy Act*. 1988.

[14] Jos CM Baeten. “A brief history of process algebra”. In : *Theoretical Computer Science* (2005).

[15] Ranjbar A. Balisane, Ravishankar Borgaonkar, Ahmad Atamli-Reineh et Andrew Martin. “Architectures for Enhancing Authentication Privacy and Security Using Trusted Computing”. In : *Proceedings of the 2nd Workshop on System Software for Trusted Execution*. Association for Computing Machinery, 2017.

[16] Ntsako Baloyi et Paula Kotzé. “Guidelines for Data Privacy Compliance : A Focus on Cyber-Physical Systems and Internet of Things”. In : *Proceedings of the South African Institute of Computer Scientists and Information Technologists 2019*. Association for Computing Machinery, 2019.

[17] Boutheina Bannour, Arnault Lapitre, Patrick Tessier et Guillaume Giraud. “Demonstrating The MAAT REQ Tool : Using Algebraic Process Models ToSupport Time-Sensitive Requirements Design”. In : *IEEE Real-Time Systems Symposium (RTSS)*. 2022.

[18] Masoud Barati, Omer Rana, Ioan Petri et George Theodorakopoulos. “GDPR Compliance Verification in Internet of Things”. In : *IEEE Access* (2020).

[19] Patrick Baudin, François Bobot, David Bühler, Loïc Correnson, Florent Kirchner, Nikolai Kosmatov, André Maroneze, Valentin Perrelle, Virgile Prevosto, Julien Signoles et Nicky Williams. “The dogged pursuit of bug-free C programs : the Frama-C software analysis platform”. In : *Communications of the ACM* (2021).

[20] Patrick Baudin, Jean-Christophe Filliâtre, Claude Marché, Benjamin Monate, Yannick Moy et Virgile Prevosto. *ACSL : ANSI/ISO C Specification Language*. Rapp. tech.

[21] Kai Bavendiek, Tobias Mueller, Florian Wittner, Thea Schwaneberg, Christian-Alexander Behrendt, Wolfgang Schulz, Hannes Federrath et Sibylle Schupp. “Automatically proving purpose limitation in software architectures”. In : *IFIP International Conference on ICT Systems Security and Privacy Protection*. 2019.

[22] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira et Bogdan Warinschi. “Sok : A comprehensive analysis of game-based ballot privacy definitions”. In : *2015 IEEE Symposium on Security and Privacy*. 2015.

[23] Pascal Birnstill et Jürgen Beyerer. “Building Blocks for Identity Management and Protection for Smart Environments and Interactive Assistance Systems”. In : *Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference*. Association for Computing Machinery, 2018.

[24] Per Bjesse. “What is Formal Verification ?” In : (2005).

[25] Allan Blanchard. *Introduction to C program proof with Frama-C and its WP plugin*. Tutorial. 2020.

[26] Sandrine Blazy, David Bühler et Boris Yakobowski. “Structuring abstract interpreters through state and value abstractions”. In : *Int. Conf. on Verification, Model Checking, and Abstract Interpretation*. 2017.

[27] Louis Brandeis et Samuel Warren. “The right to privacy”. In : *Harvard law review* (1890).

[28] Pearl Brereton, Barbara A Kitchenham, David Budgen, Mark Turner et Mohamed Khalil. “Lessons from applying the systematic literature review process within the software engineering domain”. In : *Journal of systems and software* (2007).

[29] Anton Bykov, Vladimir Ivanov, Alan Rogers, Alexandr Shunevich, Alberto Sillitti, Giancarlo Succi, Alexander Tormasov, Jooyong Yi, Albert Zabirov et Denis Zaplatnikov. “A New Architecture and Implementation Strategy for Non-Invasive Software Measurement Systems”. In : *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. Association for Computing Machinery, 2018.

[30] Michele C. et Alberto T. “BPMN : An introduction to the standard”. In : *Computer Standards & Interfaces* (2012).

[31] Elmer C. Matel, Ariel M. Sison et Ruji P. Medina. “Implementation of GA-IFS-Based Network Intrusion Detection System : A Comparative Analysis”. In : *2020 The 4th International Conference on E-Society, E-Education and E-Technology*. 2020.

[32] Julio C. Caiza, Jose M. Del Alamo et Danny S. Guamán. “A Framework and Roadmap for Enhancing the Application of Privacy Design Patterns”. In : *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. Association for Computing Machinery, 2020.

[33] Sébastien Canard, Aïda Diop, Nizar Kheir, Marie Paindavoine et Mohamed Sabt. “BlindIDS : Market-Compliant and Privacy-Friendly Intrusion Detection System over Encrypted Traffic”. In : *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. Association for Computing Machinery, 2017.

[34] Melissa Chase, Trevor Perrin et Greg Zaverucha. “The Signal Private Group System and Anonymous Credentials Supporting Efficient Verifiable Encryption”. In : *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, 2020.

[35] Yoonsik Cheon et Gary T Leavens. “A simple and practical approach to unit testing : The JML and JUnit way”. In : *ECOOP 2002—Object-Oriented Programming : 16th European Conference Màlaga, Spain, June 10–14, 2002 Proceedings 16*. 2002.

[36] Jesùs Mauricio Chimento, Wolfgang Ahrendt et Gerardo Schneider. “Testing meets static and runtime verification”. In : *Proceedings of the 6th Conference on Formal Methods in Software Engineering*. 2018.

[37] Edmund M Clarke. “Model checking”. In : *Foundations of Software Technology and Theoretical Computer Science : 17th Conference Kharagpur, India, December 18–20, 1997 Proceedings 17*. 1997.

[38] Lori A. Clarke et David S. Rosenblum. “A Historical Perspective on Runtime Assertion Checking in Software Development”. In : *SIGSOFT Softw. Eng. Notes* (2006).

[39] Michael R. Clarkson et Fred B. Schneider. “Hyperproperties”. In : *Journal of Computer Security* (2010).

[40] Myriam Clouet, Thibaud Antignac, Mathilde Arnaud, Gabriel Pedroza et Julien Signoles. “A New Generic Representation for Modeling Privacy”. In : *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 2022.

[41] Myriam Clouet, Thibaud Antignac, Mathilde Arnaud et Julien Signoles. “Context Specification Language for Formally Verifying Consent Properties on Models and Code”. In : *TAP 2023 - 17th International Conference on Tests and Proofs*. 2023.

[42] European Comission. *Proposal for a Regulation of the European Parliament and of the Council. On the protection of individuals with regard to the processing of personal data and on the free movement of such data*. 2012.

[43] Commission Nationale Informatique et Libertés. *Exploration des enjeux éthiques, techniques et juridiques des assistants vocaux*. Rapp. tech. 2020.

[44] Sylvain Conchon, Albin Coquereau, Mohamed Iguernlala et Alain Mebsout. “Alt-Ergo 2.2”. In : *SMT Workshop : International Workshop on Satisfiability Modulo Theories*. 2018.

[45] Patrick Cousot. “Abstract Interpretation : From 0, 1, To ∞”. In : *Challenges of Software Verification*. 2023.

[46] Patrick Cousot et Radhia Cousot. “Abstract interpretation : a unified lattice model for static analysis of programs by construction or approximation of fixpoints”. In : *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. 1977.

[47] Tran Khanh Dang, Chan Nam Ngo, Trong Nhan Phan et Nguyen Nhat Minh Ngo. “An Open Design Privacy-Enhancing Platform Supporting Location-Based Applications”. In : *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*. Association for Computing Machinery, 2012.

[48] Jose M Del Alamo, Danny S Guaman, Boni García et Ana Diez. “A systematic mapping study on automated analysis of privacy policies”. In : *Computing* (2022).

[49] Stéphanie Delaune, Mark Ryan et Ben Smyth. “Automatic verification of privacy properties in the applied pi calculus”. In : *IFIP International Conference on Trust Management*. 2008.

[50] Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel et Wouter Joosen. “A privacy threat analysis framework : supporting the elicitation and fulfillment of privacy requirements”. In : *Requirements Engineering* (2011).

[51] Adnesh Dhamangaonkar, Prajwal Adsul, Rohini Sarode et Sunil Mane. “Secure, Decentralized, Privacy Preserving Machine Learning System Implementation over Blockchain”. In : *Proceedings of the 2021 3rd Blockchain and Internet of Things Conference*. Association for Computing Machinery, 2021.

[52] Roel van Dijk, Christophe Creeten, Jeroen van der Ham et Jeroen van den Bos. “Model-Driven Software Engineering in Practice : Privacy-Enhanced Filtering of Network Traffic”. In : *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. Association for Computing Machinery, 2017.

[53] Guillaume Dufay, Amy Felty et Stan Matwin. “Privacy-sensitive information flow with JML”. In : *International Conference on Automated Deduction*. 2005.

[54] Fahimeh Ebrahimi, Miroslav Tushev et Anas Mahmoud. “Mobile app privacy in software engineering research : A systematic mapping study”. In : *Information and Software Technology* (2021).

[55] European Parliament, Council of the European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. Rapp. tech. 2016.

[56] Nicola Fabiano. “Robotics, Intelligent Systems, Ethics and Data Protection : The New Challenge”. In : *Proceedings of the 2nd International Conference on Applications of Intelligent Systems*. Association for Computing Machinery, 2019.

[57] Yliès Falcone, Klaus Havelund et Giles Reger. “A tutorial on runtime verification”. In : *Engineering dependable software systems* (2013).

[58] Yliès Falcone, Klaus Havelund et Giles Reger. “A tutorial on runtime verification”. In : *Engineering dependable software systems* (2013).

[59] Jean-Christophe Filliâtre, Léon Gondelman et Andrei Paskevich. “The spirit of ghost code”. In : *Formal Methods in System Design* (2016).

[60] Simone Fischer-Hubner et Amon Ott. “From a formal privacy model to its implementation”. In : *Proceedings of the 21st National Information Systems Security Conference*. 1998.

[61] Damjan Fujs, Simon Vrhovec et Anže Mihelič. “What Drives the Motivation to Self-Protect on Social Networks ? The Role of Privacy Concerns and Perceived Threats”. In : *Proceedings of the Central European Cybersecurity Conference 2018*. Association for Computing Machinery, 2018.

[62] Raphaël Gellert et Serge Gutwirth. “The legal construction of privacy and data protection”. In : *Computer Law & Security Review* (2013).

[63] Mohamad Gharib, Paolo Giorgini et John Mylopoulos. “COPri v. 2—A core ontology for privacy requirements”. In : *Data & Knowledge Engineering* (2021).

[64] David Greenaway, June Andronick et Gerwin Klein. “Bridging the Gap : Automatic Verified Abstraction of C”. In : 2012.

[65] E. Grünewald, Paul Wille, Frank Pallas, Maria C Borges et Max-R Ulbricht. “TIRA : An OpenAPI Extension and Toolbox for GDPR Transparency in RESTful Architectures”. In : *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS & PW)*. 2021.

[66] Danny S Guaman, Jose M Del Alamo et Julio C Caiza. “A systematic mapping study on software quality control techniques for assessing privacy in information systems”. In : *IEEE access* (2020).

[67] Chaoran Guo, Haiyan Zheng et Jindong Guo. “A Brief Analysis of Privacy Protection Strategy for Block Chain-Based Internet of Things System”. In : *Proceedings of the 3rd International Conference on Big Data Research*. Association for Computing Machinery, 2020.

[68] Reiner Hähnle et Marieke Huisman. “Deductive Software Verification : From Pen-and-Paper Proofs to Industrial Tools”. In : *Computing and Software Science : State of the Art and Perspectives*. 2019.

[69] Rob Hall et Stephen E Fienberg. “Privacy-preserving record linkage”. In : *International conference on privacy in statistical databases*. 2010.

[70] Katia Hayati et Martin Abadi. “Language-based enforcement of privacy policies”. In : *International Workshop on Privacy Enhancing Technologies*. 2004.

[71] Lucca Hirschi, David Baelde et Stéphanie Delaune. “A method for unbounded verification of privacy-type properties”. In : *Journal of Computer Security* (2019).

[72] Christoph Hochreiner, Markus Huber, Georg Merzdovnik et Edgar Weippl. “Towards Practical Methods to Protect the Privacy of Location Information with Mobile Devices”. In : *Proceedings of the 7th International Conference on Security of Information and Networks*. Association for Computing Machinery, 2014.

[73] Tore Hoel, Dai Griffiths et Weiqin Chen. “The Influence of Data Protection and Privacy Frameworks on the Design of Learning Analytics Systems”. In : *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*. Association for Computing Machinery, 2017.

[74] Jaap-Henk Hoepman. “Privacy design strategies”. In : *IFIP International Information Security Conference*. 2014.

[75] Martin Horák, Václav Stupka et Martin Husák. “GDPR Compliance in Cybersecurity Software : A Case Study of DPIA in Information Sharing Platform”. In : *Proceedings of the 14th International Conference on Availability, Reliability and Security*. Association for Computing Machinery, 2019.

[76] Meriem Houmer et Moulay Lahcen Hasnaoui. “Enhancing Vehicular Ad-Hoc Networks Security Using Intrusion Detection System Techniques”. In : *Proceedings of the 4th International Conference on Smart City Applications*. Association for Computing Machinery, 2019.

[77] Baohua Huang, Fengwei Jia, Jiguo Yu et Wei Cheng. “A Transparent Framework Based on Accessing Bridge and Mobile App for Protecting Database Privacy with PKI”. In : *Proceedings of the 1st ACM Workshop on Privacy-Aware Mobile Computing*. Association for Computing Machinery, 2016.

[78] Gérard Huet, Gilles Kahn et Christine Paulin-Mohring. “The coq proof assistant a tutorial”. In : *Rapport Technique* (1997).

[79] R.L. David Hughes. “Two concepts of privacy”. In : *Computer Law & Security Review* (2015).

[80] Tahani Hussain et Ranya Alawadhi. “A Privacy Protection System in Context-Aware Environment The Privacy Controller Module”. In : *Proceedings of the 22nd International Conference on Information Integration and Web-Based Applications & Services*. Association for Computing Machinery, 2021.

[81] Michael Huth et Mark Ryan. *Logic in Computer Science : Modelling and Reasoning about Systems*. Cambridge University Press ; 2nd edition, 2004.

[82] International Organization for Standardization. *Information technology - Security techniques - Privacy framework*. Rapp. tech. 2011.

[83] Leonardo Horn Iwaya, Aakash Ahmad et M Ali Babar. "Security and privacy for mHealth and uHealth systems : a systematic mapping study". In : *IEEE Access* (2020).

[84] Ruoxi Jia, Roy Dong, S. Shankar Sastry et Costas J. Spanos. "Privacy-Enhanced Architecture for Occupancy-Based HVAC Control". In : *Proceedings of the 8th International Conference on Cyber-Physical Systems*. Association for Computing Machinery, 2017.

[85] Yusheng Jiang, Tamotsu Noguchi, Nobuyuki Kanno, Yoshiko Yasumura, Takuya Suzuki, Yu Ishimaki et Hayato Yamana. "A Privacy-Preserving Query System Using Fully Homomorphic Encryption with Real-World Implementation for Medicine-Side Effect Search". In : *Proceedings of the 21st International Conference on Information Integration and Web-Based Applications & Services*. Association for Computing Machinery, 2020.

[86] Claiborne Johnson, Thomas MacGahan, John Heaps, Kevin Baldor, Jeffery von Ronne et Jianwei Niu. "Verifiable Assume-Guarantee Privacy Specifications for Actor Component Architectures". In : *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies*. Association for Computing Machinery, 2017.

[87] Rezvan Joshaghani, Stacy Black, Elena Sherman et Hoda Mehrpouyan. "Formal Specification and Verification of User-Centric Privacy Policies for Ubiquitous Systems". In : *Proceedings of the 23rd International Database Applications & Engineering Symposium*. Association for Computing Machinery, 2019.

[88] Mohsin Junaid, Jiang Ming et David Kung. "StateDroid : Stateful Detection of Stealthy Attacks in Android Apps via Horn-Clause Verification". In : *Proceedings of the 34th Annual Computer Security Applications Conference*. Association for Computing Machinery, 2018.

[89] Kentaro Kita, Yoshiki Kurihara, Yuki Koizumi et Toru Hasegawa. "Location Privacy Protection with a Semi-Honest Anonymizer in Information Centric Networking". In : *Proceedings of the 5th ACM Conference on Information-Centric Networking*. Association for Computing Machinery, 2018.

[90] Barbara Kitchenham, O Pearl Brereton, David Budgen, Mark Turner, John Bailey et Stephen Linkman. “Systematic literature reviews in software engineering–a systematic literature review”. In : *Information and software technology* (2009).

[91] Barbara Kitchenham, Stuart Charters et al. *Guidelines for performing systematic literature reviews in software engineering*. 2007.

[92] Barbara Kitchenham, Emilia Mendes et Guilherme H Travassos. “A systematic review of cross-vs. within-company cost estimation studies”. In : *10th International Conference on Evaluation and Assessment in Software Engineering (EASE) 10*. 2006.

[93] Georgios Kontaxis et Angelos D. Keromytis. “Protecting Insecure Communications with Topology-Aware Network Tunnels”. In : *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, 2016.

[94] Karel Kubicek, Jakob Merane, Carlos Cotrini, Alexander Stremitzer, Stefan Bechtold et David Basin. “Checking Websites’ GDPR Consent Compliance for Marketing Emails”. In : *Proceedings on Privacy Enhancing Technologies* (2022).

[95] Anelia Kurteva, Tek Raj Chhetri, Harshvardhan J Pandit et Anna Fensel. “Consent through the lens of semantics : State of the art survey and best practices”. In : *Semantic Web* (2021).

[96] Jussi Laakkonen, Salla Annala et Pekka Jappinen. “Abstracted architecture for smart grid privacy analysis”. In : *2013 International Conference on Social Computing*. 2013.

[97] Gary T Leavens et Yoonsik Cheon. *Design by Contract with JML*. 2006.

[98] Valentina Leone, Luigi Di Caro et Serena Villata. “Taking stock of legal ontologies : a feature-based comparative analysis”. In : *Artificial Intelligence and Law* (2020).

[99] Chao Li et Balaji Palanisamy. “ReverseCloak : Protecting Multi-Level Location Privacy over Road Networks”. In : *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. Association for Computing Machinery, 2015.

[100] Qinan Li et Zhihao Xue. “A Privacy-Protecting Authorization System Based on Blockchain and Zk-SNARK”. In : *Proceedings of the 2020 International Conference on Cyberspace Innovation of Advanced Technologies*. Association for Computing Machinery, 2021.

[101] Ye Li et Fumio Teraoka. “Privacy Protection for Low-Cost RFID Tags in IoT Systems”. In : *Proceedings of the 7th International Conference on Future Internet Technologies*. Association for Computing Machinery, 2012.

[102] Chuan-Gang Liu, Ya-Ming Shiue et Meng-Yao Wu. “Constructing a Secure Pharmaceutical Care System for the Elderly with a Mutual Authentication Protection”. In : *Proceedings of the 5th International Conference on Medical and Health Informatics*. Association for Computing Machinery, 2021.

[103] Hui Liu, Changyu Li, Xuancheng Jin, Juanru Li, Yuanyuan Zhang et Dawu Gu. “Smart Solution, Poor Protection : An Empirical Study of Security and Privacy Issues in Developing and Deploying Smart Home Devices”. In : *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*. Association for Computing Machinery, 2017.

[104] Tom Lodge, Andy Crabtree et Anthony Brown. “IoT App Development : Supporting Data Protection by Design and Default”. In : *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. Association for Computing Machinery, 2018.

[105] Edoardo Longo, Alessandro E. C. Redondi et Matteo Cesana. “Design and Implementation of a Privacy-Aware Smart Mirror System”. In : *Proceedings of the 1st Workshop on Experiences with the Design and Implementation of Frugal Smart Objects*. Association for Computing Machinery, 2020.

[106] Faiza Loukil. “Towards a new data privacy-based approach for IoT”. Thèse de doct. Université Jean Moulin Lyon 3, 2019.

[107] Mengwei Lu, Patrick P. C. Lee et John C. S. Lui. “Identity Attack and Anonymity Protection for P2P-VoD Systems”. In : *Proceedings of the Nineteenth International Workshop on Quality of Service*. IEEE Press, 2011.

[108] Lin Ma, Jinyan Xu, Jiadong Sun, Yajin Zhou, Xun Xie, Wenbo Shen, Rui Chang et Kui Ren. “Revisiting Challenges for Selective Data Protection of Real Applications”. In : *Proceedings of the 12th ACM SIGOPS Asia-Pacific Workshop on Systems*. 2021.

[109] Erwan Mahe. “An operational semantics of interactions for verifying partially observed executions of distributed systems”. PhD Thesis. Université Paris-Saclay, 2021.

[110] Shinichi Matsumoto et Kouichi Sakurai. “A Proposal for the Privacy Leakage Verification Tool for Android Application Developers”. In : *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*. Association for Computing Machinery, 2013.

[111] Joseph Maxwell. “Understanding and validity in qualitative research”. In : *Harvard educational review* (1992).

[112] Yannic Meier, Johanna Schäwel, Elias Kyewski et Nicole C. Krämer. “Applying Protection Motivation Theory to Predict Facebook Users’ Withdrawal and Disclosure Intentions”. In : *International Conference on Social Media and Society*. Association for Computing Machinery, 2020.

[113] Saeed Mirzamohammadi et Ardalan Amiri Sani. “Viola : Trustworthy Sensor Notifications for Enhanced Privacy on Mobile Systems”. In : *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. Association for Computing Machinery, 2016.

[114] Colombe de Montety, Thibaud Antignac et Christophe Slim. “GDPR Modelling for Log-Based Compliance Checking”. In : *Trust Management XIII*. Sous la dir. de Weizhi Meng, Piotr Cofta, Christian Damsgaard Jensen et Tyrone Grandison. 2019.

[115] Miguel Ehécatl Morales-Trujillo, Gabriel Alberto García-Mireles, Erick Orlando Matla-Cruz et Mario Piattini. “A systematic mapping study on privacy by design in software engineering”. In : *CLEI electronic journal* (2019).

[116] Miguel Ehécatl Morales-Trujillo, Erick Orlando Matla-Cruz, Gabriel Alberto García-Mireles et Mario Piattini. “Privacy by Design in Software Engineering : a Systematic Mapping Study.” In : *CIbSE* (2018).

[117] Nabil Mohammed Ali Munassar et A Govardhan. “A comparison between five models of software engineering”. In : *International Journal of Computer Science Issues (IJCSI)* (2010).

[118] Andrew C Myers. “JFlow : Practical mostly-static information flow control”. In : *Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. 1999.

[119] Ben Nassi, Ron Bitton, Ryusuke Masuoka, Asaf Shabtai et Yuval Elovici. “SoK : Security and privacy in the age of commercial drones”. In : *Proc. IEEE Symp. Security Privacy (SP)*. 2021.

[120] Anirudh Sunil Nath, Revathi Venkataraman, M. Pushpalatha et D. Vanusha. “Implementation of Data Privacy between Nodes Using AES in Wireless Ad Hoc Networks”. In : *Proceedings of the Second ACM MobiHoc Workshop on Airborne Networks and Communications*. Association for Computing Machinery, 2013.

[121] Mohammad Nauman, Sohail Khan et Xinwen Zhang. “Apex : extending android permission model and enforcement with user-defined runtime constraints”. In : *Proceedings of the 5th ACM symposium on information, computer and communications security*. 2010.

[122] Dorgival Netto, Carla Silva et João Araújo. “Identifying How the Brazilian Software Industry Specifies Legal Requirements”. In : *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*. Association for Computing Machinery, 2019.

[123] Chris Norval, Heleen Janssen, Jennifer Cobbe et Jatinder Singh. “RECLAIMING Data : Overcoming App Identification Barriers for Exercising Data Protection Rights”. In : *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. Association for Computing Machinery, 2018.

[124] Ian Oliver. *Privacy engineering : A dataflow and ontological approach*. CreateSpace Independent Publishing Platform, 2014.

[125] Alessandro Oltramari, Dhivya Piraviperumal, Florian Schaub, Shomir Wilson, Sushain Cherivirala, Thomas B Norton, N Cameron Russell, Peter Story, Joel Reidenberg et Norman Sadeh. “PrivOnto : A semantic framework for the analysis of privacy policies”. In : *Semantic Web* (2018).

[126] Harshvardhan J. Pandit, Declan O’Sullivan et Dave Lewis. “An Ontology Design Pattern for Describing Personal Data in Privacy Policies.” In : *WOP@ ISWC*. 2018.

[127] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha et Michael P Wellman. “Sok : Security and privacy in machine learning”. In : *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2018.

[128] Personal Information Protection Commission. *Act on the Protection of Personal Information*. 2016.

[129] Kai Petersen, Robert Feldt, Shahid Mujtaba et Michael Mattsson. “Systematic mapping studies in software engineering”. In : *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*. 2008.

[130] Kai Petersen et Cigdem Gencel. “Worldviews, research methods, and their relationship to validity in empirical software engineering research”. In : *2013 joint conference of the 23rd international workshop on software measurement and the 8th international conference on software process and product measurement*. 2013.

[131] Kai Petersen, Sairam Vakkalanka et Ludwik Kuzniarz. “Guidelines for conducting systematic mapping studies in software engineering : An update”. In : *Information and software technology* (2015).

[132] Milan Petković, Davide Prandi et Nicola Zannone. “Purpose control : Did you process the data for the intended purpose ?” In : *Workshop on Secure Data Management*. 2011.

[133] Andreas Pfitzmann et Marit Hansen. “A terminology for talking about privacy by data minimization : Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management”. In : (2010).

[134] Hoang Pham. *Software reliability*. Springer Science & Business Media, 2000.

[135] Benjamin C Pierce. *Types and programming languages*. 2002.

[136] *Privacy Online : Fair Information Practices in the Electronic Marketplace : A Federal Trade Commission Report to Congress*. Rapp. tech. US Federal Trade Commission, 2000.

[137] Chenxi Qiu, Anna Squicciarini, Zhuozhao Li, Ce Pang et Li Yan. “Time-Efficient Geo-Obfuscation to Protect Worker Location Privacy over Road Networks in Spatial Crowdsourcing”. In : *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. Association for Computing Machinery, 2020.

[138] Farzana Rahman, Ivor D. Addo et Sheikh I. Ahamed. “PriSN : A Privacy Protection Framework for Healthcare Social Networking Sites”. In : *Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems*. Association for Computing Machinery, 2014.

[139] Mariana Raykova, Hasnain Lakhani, Hasanat Kazmi et Ashish Gehani. “Decentralized Authorization and Privacy-Enhanced Routing for Information-Centric Networks”. In : *Proceedings of the 31st Annual Computer Security Applications Conference*. Association for Computing Machinery, 2015.

[140] Henrique C. Carvalho de Resende, Joao Paulo de Brito Gonçalves, Cristiano B. Both et Johann M. Marquez-Barja. “Enabling QoS-Secured Enhanced Non-Public Network Slices for Health Environments”. In : *Proceedings of the 6th EAI International Conference on Smart Objects and Technologies for Social Good*. Association for Computing Machinery, 2020.

[141] Virgile Robles. “Specifying and verifying high-level requirements on large programs : application to security of C programs”. Thèse de doct. Université Paris-Saclay, 2022.

[142] Virgile Robles, Nikolai Kosmatov, Virgile Prevosto, Louis Rilling et Pascale Le Gall. “MetAcsl : specification and verification of high-level properties”. In : *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. 2019.

[143] Marco Robol, Travis D Breaux, Elda Paja et Paolo Giorgini. “Consent Verification Under Evolving Privacy Policies”. In : *2019 IEEE 27th International Requirements Engineering Conference (RE)*. 2019.

[144] Michael Rushanan, Aviel D Rubin, Denis Foo Kune et Colleen M Swanson. “Sok : Security and privacy in implantable medical devices and body area networks”. In : *2014 IEEE symposium on security and privacy*. 2014.

[145] Eunsu Ryu, Yao Rong, Jie Li et Ashwin Machanavajjhala. “Curso : Protect Yourself from Curse of Attribute Inference : A Social Network Privacy-Analyzer”. In : *Proceedings of the ACM SIGMOD Workshop on Databases and Social Networks*. Association for Computing Machinery, 2013.

[146] Wannida Sae-Tang, Masaaki Fujiyoshi et Hitoshi Kiya. “Encryption-Then-Compression-Based Copyright- and Privacy-Protected Image Trading System”. In : *Proceedings of the International Conference on Advances in Image Processing*. Association for Computing Machinery, 2017.

[147] Wannida Sae-Tang et Hitoshi Kiya. “A Generation of Meaningfully Secured Images for Copyright- and Privacy-Protected Image Trading Systems Using Singular Component Interchange”. In : *Proceedings of the International Conference on Advances in Image Processing*. Association for Computing Machinery, 2017.

[148] S Madhukar Salve, S Neha Samreen et Neha Khatri-Valmik. “A Comparative Study on Software Development Life Cycle Models”. In : *International Research Journal of Engineering and Technology (IRJET)* (2018), p. 696-700.

[149] Kiavash Satvat, Maliheh Shirvanian, Mahshid Hosseini et Nitesh Saxena. “CREPE : A Privacy-Enhanced Crash Reporting System”. In : *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*. Association for Computing Machinery, 2020.

[150] Ashutosh Saxena, Ina Jain et M. Choudary Gorantla. “An Integrated Framework for Enhancing Privacy in Online Social Networks”. In : *Proceedings of the 6th ACM India Computing Convention*. Association for Computing Machinery, 2013.

[151] Zifei Shan, Haowen Cao, Jason Lv, Cong Yan et Annie Liu. “Enhancing and Identifying Cloning Attacks in Online Social Networks”. In : *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*. Association for Computing Machinery, 2013.

[152] S Shylesh. “A study of software development life cycle process models”. In : *National Conference on Reinventing Opportunities in Management, IT, and Social Sciences*. 2017.

[153] Julien Signoles. “The E-ACSL perspective on runtime assertion checking”. In : *Proceedings of the 5th ACM International Workshop on Verification and mOnitoring at Runtime EXecution*. 2021.

[154] Julien Signoles, Nikolai Kosmatov et Kostyantyn Vorobyov. “E-ACSL, a Runtime Verification Tool for Safety and Security of C Programs. Tool Paper”. In : *Int. Workshop on Competitions, Usability, Benchmarks, Evaluation, and Standardisation for Runtime Verification Tools (RV-CuBES)*. 2017.

[155] Jefferson Silva, Newton Calegari et Eduardo Gomes. “After Brazil’s General Data Protection Law : Authorization in Decentralized Web Applications”. In : *Companion Proceedings of The 2019 World Wide Web Conference*. Association for Computing Machinery, 2019.

[156] Manya Sleeper, Divya Sharma et Lorrie Faith Cranor. “I Know Where You Live : Analyzing Privacy Protection in Public Databases”. In : *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*. Association for Computing Machinery, 2011.

[157] Joshua B Smith. *Practical OCaml*. 2006.

[158] Ben Smyth et David Bernhard. “Ballot Secrecy and Ballot Independence Coincide”. In : *Computer Security – ESORICS 2013*. Springer Berlin Heidelberg, 2013.

[159] Daniel J. Solove. “A taxonomy of privacy”. In : (2005).

[160] Ha Xuan Son et Nguyen Minh Hoang. “A Novel Attribute-Based Access Control System for Fine-Grained Privacy Protection”. In : *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy*. Association for Computing Machinery, 2019.

[161] Fei Song, Wei Quan, Tianming Zhao, Hongke Zhang, Ziwei Hu et Ilsun You. “Ports Distribution Management for Privacy Protection inside Local Domain Name System”. In : *Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats*. Association for Computing Machinery, 2016.

[162] Anna Cinzia Squicciarini, Giuseppe Petracca et Elisa Bertino. “Adaptive Data Protection in Distributed Systems”. In : *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*. Association for Computing Machinery, 2013.

[163] Degang Sun, Chenyang Guo, Dali Zhu et Weimiao Feng. “Secure HybridApp : A detection method on the risk of privacy leakage in HTML5 hybrid applications based on dynamic taint tracking”. In : *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. 2016.

[164] Mingshen Sun, Min Zheng, John C. S. Lui et Xuxian Jiang. “Design and Implementation of an Android Host-Based Intrusion Prevention System”. In : *Proceedings of the 30th Annual Computer Security Applications Conference*. Association for Computing Machinery, 2014.

[165] Vinh Thong Ta et Max Hashem Eiza. “DataProVe : Fully Automated Conformance Verification Between Data Protection Policies and System Architectures”. In : *Proceedings on Privacy Enhancing Technologies* (2022).

[166] Naseeb Thapaliya, Lavanya Goluguri et Shan Suthaharan. “Asymptotically Stable Privacy Protection Technique for FMRI Shared Data over Distributed Computer Networks”. In : *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. Association for Computing Machinery, 2020.

[167] Cristina Timón López, Ignacio Alamillo Domingo et Julián Valero Torrijos. “Approaching the Data Protection Impact Assessment as a Legal Methodology to Evaluate the Degree of Privacy by Design Achieved in Technological Proposals. A Special Reference to Identity Management Systems”. In : *Proceedings of the 16th In-*

ternational Conference on Availability, Reliability and Security. Association for Computing Machinery, 2021.

[168] Shukun Tokas, Olaf Owe et Toktam Ramezanifarkhani. “Language-based mechanisms for privacy-by-design”. In : *IFIP International Summer School on Privacy and Identity Management*. 2019.

[169] Casey Tran, Reza Tourani, Satyajayant Misra, Travis Machacek et Gaurav Panwar. “Analyzing GDPR Compliance of Named Data Networking”. In : *Proceedings of the 8th ACM Conference on Information-Centric Networking*. Association for Computing Machinery, 2021.

[170] Michael Carl Tschantz, Anupam Datta et Jeannette M. Wing. “Formalizing and enforcing purpose restrictions in privacy policies”. In : *IEEE Symposium on Security and Privacy*. 2012.

[171] Michael Carl Tschantz, Shayak Sen et Anupam Datta. “Sok : Differential privacy as a causal property”. In : *2020 IEEE Symposium on Security and Privacy (SP)*. 2020.

[172] Michael Carl Tschantz et Jeannette M. Wing. “Formal methods for privacy”. In : *International Symposium on Formal Methods*. 2009.

[173] Denis Ulybyshev, Ibrahim Yilmaz, Bradley Northern, Vadim Kholodilo et Michael Rogers. “Trustworthy Data Analysis and Sensor Data Protection in Cyber-Physical Systems”. In : *Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*. Association for Computing Machinery, 2021.

[174] Dinusha Vatsalan, Peter Christen et Vassilios S Verykios. “A taxonomy of privacy-preserving record linkage techniques”. In : *Information Systems* (2013).

[175] Stef Verreydt, Koen Yskout et Wouter Joosen. “Security and Privacy Requirements for Electronic Consent : A Systematic Literature Review”. In : *ACM Transactions on Computing for Healthcare* (2021).

[176] Alan R. Wagner. “An Autonomous Architecture That Protects the Right to Privacy”. In : *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. Association for Computing Machinery, 2018.

[177] Dolores R. Wallace et Roger U. Fujii. “Software verification and validation : an overview”. In : *IEEE Software* (1989).

[178] Yingjie Wang, Xing Liu, Weixuan Mao et Wei Wang. “DCDroid : Automated Detection of SSL/TLS Certificate Verification Vulnerabilities in Android Apps”. In : *Proceedings of the ACM Turing Celebration Conference - China*. Association for Computing Machinery, 2019.

[179] Yuanfeng Wen, JongHyuk Lee, Ziyi Liu, Qingji Zheng, Weidong Shi, Shouhuai Xu et Taeweon Suh. “Multi-Processor Architectural Support for Protecting Virtual Machine Privacy in Untrusted Cloud Environment”. In : *Proceedings of the ACM International Conference on Computing Frontiers*. Association for Computing Machinery, 2013.

[180] Claes Wohlin. “Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering”. In : *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. 2014.

[181] Working Party 29. *ARTICLE 29 DATA PROTECTION WORKING PARTY - Opinion 03/2013 on purpose limitation*. Rapp. tech. 2013.

[182] Guowen Xu, Hongwei Li, Shengmin Xu, Hao Ren, Yinghui Zhang, Jianfei Sun et Robert H. Deng. “Catch You If You Deceive Me : Verifiable and Privacy-Aware Truth Discovery in Crowdsensing Systems”. In : *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. Association for Computing Machinery, 2020.

[183] Mohammed Amine Yagoub, Abdelkader Laouid, Ahcène Bounceur et Muath Alshaikh. “An Intelligent Cloud Data Protection Technique Based on Multi Agent System Using Advanced Cryptographic Algorithms”. In : *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems*. Association for Computing Machinery, 2019.

[184] Junpei Yamaguchi et Ryuya Uda. “Proposal of Privacy Protection System for Web Forms Using Bloom Filter”. In : *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*. Association for Computing Machinery, 2012.

[185] Cheng Yong, Wu Jiangjiang, Mei Songzhu, Wang Zhiying, Jun Ma, Ren Jiangchun et Yan Ke. “Mailbook : Privacy-Protecting Social Networking via Email”. In : *Proceedings of the Third International Conference on Internet Multimedia Computing and Service*. Association for Computing Machinery, 2011.

[186] Deng Yun, Li Zonglin et Chen Shouxue. “The Methods of Range Query and Access Control on Multidimensional Data of Two-Layer Wireless Sensor Network Facing Privacy Protection”. In : *Proceedings of the 4th International Conference on Intelligent Information Processing*. Association for Computing Machinery, 2020.

[187] Aiqing Zhang, Abel Bacchus et Xiaodong Lin. “Consent-based access control for secure and privacy-preserving health information exchange”. In : *Security and Communication Networks* (2016).

[188] Xinwen Zhang, Francesco Parisi-Presicce, Ravi Sandhu et Jaehong Park. “Formal model and policy specification of usage control”. In : *ACM Transactions on Information and System Security (TISSEC)* (2005).

[189] Zhe Zhou, Di Tang, Wenhao Wang, Xiaofeng Wang, Zhou Li et Kehuan Zhang. “Beware of Your Screen : Anonymous Fingerprinting of Device Screens for Off-Line Payment Protection”. In : *Proceedings of the 34th Annual Computer Security Applications Conference*. Association for Computing Machinery, 2018.

[190] Xingzhen Zhu, Peng Zhu et Yanmei Qiu. “Research on the Influence of Fear Appeals on APP Users’ Privacy Protection Behavior”. In : *Proceedings of the 1st International Conference on Information Management and Management Science*. Association for Computing Machinery, 2018.

A - ANNEXES

A.1 . Exemple Gestion Données Médicales - version Simple

Cette section regroupe les annexes concernant l'exemple Gestion Données Médicales - version Simple, défini en section 7.3.1.

A.1.1 . CSpeL - Traces d'exécution

Cette section présente des formalisations en CSpeL de traces d'exécution possibles pour la réalisation de l'essai clinique.

Au niveau Modèle Graphique

La trace

$$Handle(P2, EPR); \epsilon$$

dénote le traitement de EPR par le processus $P2$, ce qui correspond, à ce niveau-là, à la réalisation de l'essai clinique.

Au niveau Modèle Formel

La trace

$$Handle(T22, £epr); Handle(T23, £epr); \epsilon$$

dénote la réalisation du traitement médical au niveau Modèle Formel. Les lignes de vie S2, T21 et E2 ne traitant aucune données, elles ne sont pas présentes dans les traces.

A.1.2 . Fichier pour IAT

La figure A.1 et la figure A.2 correspondent au fichier complet de spécification pour IAT, correspondant au système de l'exemple GDM$_{Spl}$.

```
@signature{
  @lifeline{
    S1; T11; T12; T13; T14; E1; S2; T21; T22; T23; E2
  }
  @message{
    START;
    collectSymptoms(Integer);
    makeDiagnosis(Integer);
    prescribeTreatment(Integer);
    dischargePatient;
    prepareTrial;
    performTrial(Integer);
    analyzeResults(Integer);
    END
  }
  @variable{
    epr : Integer;
  }
  @init{
    &epr = 4;
  }
}
@option{
  /* Declaring options for various processes proposed by the tool */
  @explore_option{
    /* Options for the exploration process which
    *  explores the semantics of the model
    *  following certain heuristics */
    @general_option{
      loggers = [graphic[svg,vertical]]; strategy = BreadthFS
    }
  }
  @analyze_option{
    /* Options for the analysis process which
    *  evaluates the conformity of a (multi-)trace
    *  against the model */
    @general_option{
      loggers = [graphic[svg,vertical]]; strategy = DepthFS
    };
    analysis_kind = prefix; local_analysis = true;
    goal = WeakPass
  }
}
```

Figure A.1 – Fichier de spécification pour IAT pour GDM$_{Spl}$ (partie 1).

```
@model{
  loopH(
    alt(
      seq(
        S1 -- START -> T11,
        T10 -- next -> T11, /* check medical equipment */
        T11 -- send(&epr) -> T12, /* collectSymptoms */
        T12 -- send(&epr) -> T13, /* makeDiagnosis */
        T13 -- send(&epr) -> T14, /* prescribeTreatment */
        T14 -- next -> E1, /* dischargePatient */
        E1 -- END -> |
      ),
      seq(
        S2 -- START -> T21,
        T21 -- next -> T22, /* prepareTrial */
        T22 -- send(&epr) -> T23, /* performTrial */
        T23 -- send(&epr) -> E2, /* analyzeResults */
        E2 -- END -> |
```

Figure A.2 – Fichier de spécification pour IAT pour GDM$_{Spl}$ (partie 2).

A.1.3 . Fichier instrumenté pour IAT

La figure A.3 et la figure A.4 présentent le fichier de l'exemple GDM$_{Spl}$ instrumenté pour la vérification des propriétés de consentement avec IAT (propriétés définies section 8.1.3), pour l'exemple GDM$_{Spl}$. Cette instrumentation suit le principe de traduction expliqué en section 9.3.1.

```
@signature{
  @lifeline{
    S1; T11; T12; T13; T14; E1; S2; T21; T22; T23; E2
  }
  @message{
    START;
    collectSymptoms(Integer);
    makeDiagnosis(Integer);
    prescribeTreatment(Integer);
    dischargePatient;
    prepareTrial;
    performTrial(Integer);
    analyzeResults(Integer);
    END }
  @variable{
    epr : Integer; epr_treatment: Bool; epr_research: Bool
  }
  @init{
    &epr = 4; &epr_treatment = ⊤; &epr_research = ⊥
  }
}
@option{
  /* Declaring options for various processes proposed by the tool */
  @explore_option{
    /* Options for the exploration process which
    *  explores the semantics of the model
    *  following certain heuristics */
    @general_option{
      loggers = [graphic[svg,vertical]];
      strategy = BreadthFS }
  }
  @analyze_option{
    /* Options for the analysis process which
    *  evaluates the conformity of a (multi-)trace
    *  against the model */
    @general_option{ strategy = DepthFS };
    analysis_kind = prefix; local_analysis = true; goal = WeakPass
  }
}
```

Figure A.3 – Fichier IAT pour l'exemple GDM$_{Spl}$ instrumenté avec le consentement (partie 1).

```
@model{
  loopH(
    alt(
      seq(
        S1 -- START -> T11,
        T11[£epr_treatment] -- collectSymptoms(£epr) -> T12,
        T12[£epr_treatment] -- makeDiagnosis(£epr) -> T13,
        T13[£epr_treatment] -- prescribeTreatment(£epr) -> T14,
        T14 -- dischargePatient -> E1,
        E1 -- END -> |
      ),
      seq(
        S2 -- START ->T21,
        T21 -- prepareTrial -> T22,
        T22[£epr_research] -- performTrial(£epr) -> T23,
        T23[£epr_research] -- analyzeResults(£epr) -> E2,
        E2 -- END -> |
      )
    )
  )
}
```

Figure A.4 – Fichier IAT pour l'exemple GDM$_{Spl}$ instrumenté avec le consentement (partie 2).

A.1.4 . Code source C

La figure A.5 présente un fichier source complet (données, fonctions, main) pour notre exemple GDM$_{Spl}$.

```c
#define NB_PATIENT 20

/*--  DATA --*/
typedef struct {
    char date[S_SIZE];
    char medecine[S_SIZE];
} Dos;

typedef struct {
    int id;
    char name[S_SIZE];
    char lastname[S_SIZE];
    char birthdate[S_SIZE];
    char address[S_SIZE];
    char sex[S_SIZE];
    Dos dosList[NB_DOS];
} Patient;

typedef struct {
    char birthdateList[NB_PATIENT][S_SIZE];
    char sexList[NB_PATIENT][S_SIZE];
    Dos dosList[NB_DOS];
} TrialData;

/*--  FUNCTIONS --*/
Patient makePrescr(Dos newd, Patient p){
    ...
}

TrialData getData(Patient patientList[NB_PATIENT]){
    ...
}

int computeStats(TrialData data){
    ...
}

/*--  MAIN --*/
int main(){
    ...
    return 0;
}
```

Figure A.5 – Fichier source pour GDM$_{Spl}$.

A.1.5 . Code généré

Fig. A.6 correspond à un example de fichier C généré par CASTT, à partir d'une spécification de context écrit en CSpeL et d'un fichier source écrit en C. Pour des raisons de lisibilités, certaines parties dans le code ont été remplacées par "..." dans l'exemple (cela ne concerne que la génération de code C par by Frama-C, et non les informations relatives à notre approche).

```
#include "stdio.h"
enum _Purposes {RESEARCH = 0,TREATMENT = 1};
typedef enum _Purposes Purposes;
enum _PersonalData {DATA = 0,PATIENT = 1};
typedef enum _PersonalData PersonalData;
/*@ ghost int \ghost Consent[2][2]; */

struct __anonstruct_Dos_1 {char date[20] ;char medecine[20] ;};
typedef struct __anonstruct_Dos_1 Dos;
struct __anonstruct_Patient_2 {
  int id ;char name[20] ;char lastname[20] ;char birthdate[20] ;
  char address[20] ;char gender[20] ;Dos dosList[20] ;};
typedef struct __anonstruct_Patient_2 Patient;
struct __anonstruct_Data_3 {
  char birthdateList[20][20] ;
  char genderList[20][20] ;Dos dosList[20] ;};
typedef struct __anonstruct_Data_3 Data;

/*@ requires Consent[PATIENT][TREATMENT] ≡ 1; */
Patient makePrescr(Dos newd, Patient p){
  return p;
}

/*@ requires
      (Consent[PATIENT][TREATMENT] ≡ 1 ∧ Consent[DATA][TREATMENT] ≡ 1) ∨
      (Consent[PATIENT][RESEARCH] ≡ 1 ∧ Consent[DATA][RESEARCH] ≡ 1);
 */
Data getData(Patient patientList[20]){
  Data d;
  Patient p = *(patientList + 0);
  return d;
}

/*@ requires Consent[DATA][RESEARCH] ≡ 1; */
int computeStats(Data data){
  int __retres;
  __retres = 0;
  return __retres;
}

int main(void){
  int __retres;
  /*@ ghost Consent[PATIENT][RESEARCH] = 0; */
  /*@ ghost Consent[PATIENT][TREATMENT] = 1; */
  /*@ ghost Consent[DATA][RESEARCH] = 0; */
  /*@ ghost Consent[DATA][TREATMENT] = 0; */
  Patient patient =
    {.id = 0,
     .name = {(char)'J', (char)'o', (char)'h', (char)'n', (char)'\000'},
     .lastname = {(char)'D', (char)'o', (char)'e', (char)'\000'},
     .birthdate = {(char)0, ..., (char)0},
     .address = {(char)0, ..., (char)0},
     .gender = {(char)0, ..., (char)0},
     .dosList = {...}};
  Dos newd =
    {.date = {(char)'0',...,(char)'\000'},
     .medecine = {(char)'T',...,(char)'\000'}};
  patient = makePrescr(newd,patient);
  __retres = 0;
  return __retres;
}
```

Figure A.6 – Exemple de fichier généré pour GDM$_{Cpx}$.

A.2 . Exemple Gestion Données Médicales - version Complexe

Cette section regroupe les annexes concernant l'exemple Gestion Données Médicales - version Complexe, défini en section 7.3.2.

A.2.1 . CSpeL - Formalisation de l'exemple

Il est possible de formaliser l'exemple GDM_{Cpx} décrit section 7.3.2, d'une façon équivalente à celle réalisée pour l'exemple GDM_{Spl}.

Au niveau Modèle Graphique (BPMN)

$$
\begin{aligned}
S &\triangleq \{HealthcareTreatementProcess; ClinicalTrialProcess\} \\
D &\triangleq \{EPR\} \\
P &\triangleq \{Traitement; Recherche\} \\
\gamma &\triangleq \begin{cases} (EPR, Traitement) \mapsto \top \\ (EPR, Recherche) \mapsto \bot \end{cases} \\
\pi &\triangleq \begin{cases} HealthcareTreatementProcess \mapsto \{Traitement\} \\ ClinicalTrialProcess \mapsto \{Recherche\} \end{cases} \\
\nu &\triangleq \begin{cases} HealthcareTreatementProcess \mapsto \{EPR\} \\ ClinicalTrialProcess \mapsto \{EPR\} \end{cases}
\end{aligned}
$$

En effet, à ce niveau d'abstraction, deux processus ont été définis : *HealthcareTreatementProcess* pour la réalisation du traitement médical (Traitement) et *ClinicalTrialProcess* pour la réalisation de l'essai clinique (Recherche). Les données personnelles sont contenues dans le dossier électronique du patient (EPR). Les deux processus auront besoin d'avoir accès à ces données. Dans notre exemple, seule la réalisation du traitement médical a été consenties par la personne concernée (*Data Subject*).

Au niveau Modèle Formel (IAT)

$$
\begin{aligned}
S &\triangleq \{S1; S2; G1; T01; T02; T03; T04; E1; T05; E2; S3; S4; T06; G2; T07; E3; \\
&\quad G3; T08; T09; E4; E5; S5; T10; T11; T12; E6; S6; T13; T14; T15; E7; S91; \\
&\quad T91; T92; T93; T94; T95; E91\}; \\
D &\triangleq \{£epr\}; \\
P &\triangleq \{Traitement, Recherche\}; \\
\gamma &\triangleq \begin{cases} (£epr, Traitement) \mapsto \top \\ (£epr, Recherche) \mapsto \bot \end{cases} \\
\pi &\triangleq \begin{cases}
S1 \mapsto \{Traitement\} & E4 \mapsto \{Traitement\} \\
S2 \mapsto \{Traitement\} & E5 \mapsto \{Traitement\} \\
G1 \mapsto \{Traitement\} & S5 \mapsto \{Traitement\} \\
T01 \mapsto \{Traitement\} & T10 \mapsto \{Traitement\} \\
T02 \mapsto \{Traitement\} & T11 \mapsto \{Traitement\} \\
T03 \mapsto \{Traitement\} & T12 \mapsto \{Traitement\} \\
T04 \mapsto \{Traitement\} & E6 \mapsto \{Traitement\} \\
E1 \mapsto \{Traitement\} & S6 \mapsto \{Traitement\} \\
T05 \mapsto \{Traitement\} & T13 \mapsto \{Traitement\} \\
E2 \mapsto \{Traitement\} & T14 \mapsto \{Traitement\} \\
S3 \mapsto \{Traitement\} & T15 \mapsto \{Traitement\} \\
S4 \mapsto \{Traitement\} & E7 \mapsto \{Traitement\} \\
T06 \mapsto \{Traitement\} & S91 \mapsto \{Recherche\} \\
G2 \mapsto \{Traitement\} & T91 \mapsto \{Recherche\} \\
T07 \mapsto \{Traitement\} & T92 \mapsto \{Recherche\} \\
E3 \mapsto \{Traitement\} & T93 \mapsto \{Recherche\} \\
G3 \mapsto \{Traitement\} & T94 \mapsto \{Recherche\} \\
T08 \mapsto \{Traitement\} & T95 \mapsto \{Recherche\} \\
T09 \mapsto \{Traitement\} & E91 \mapsto \{Recherche\}
\end{cases}
\end{aligned}
$$

$$
\nu \triangleq \begin{cases}
S1 \mapsto \{\pounds epr\} & E4 \mapsto \{\pounds epr\} \\
S2 \mapsto \{\pounds epr\} & E5 \mapsto \{\pounds epr\} \\
G1 \mapsto \{\pounds epr\} & S5 \mapsto \{\pounds epr\} \\
T01 \mapsto \{\pounds epr\} & T10 \mapsto \{\pounds epr\} \\
T02 \mapsto \{\pounds epr\} & T11 \mapsto \{\pounds epr\} \\
T03 \mapsto \{\pounds epr\} & T12 \mapsto \{\pounds epr\} \\
T04 \mapsto \{\pounds epr\} & E6 \mapsto \{\pounds epr\} \\
E1 \mapsto \{\pounds epr\} & S6 \mapsto \{\pounds epr\} \\
T05 \mapsto \{\pounds epr\} & T13 \mapsto \{\pounds epr\} \\
E2 \mapsto \{\pounds epr\} & T14 \mapsto \{\pounds epr\} \\
S3 \mapsto \{\pounds epr\} & T15 \mapsto \{\pounds epr\} \\
S4 \mapsto \{\pounds epr\} & E7 \mapsto \{\pounds epr\} \\
T06 \mapsto \{\pounds epr\} & S91 \mapsto \{\pounds epr\} \\
G2 \mapsto \{\pounds epr\} & T91 \mapsto \{\pounds epr\} \\
T07 \mapsto \{\pounds epr\} & T92 \mapsto \{\pounds epr\} \\
E3 \mapsto \{\pounds epr\} & T93 \mapsto \{\pounds epr\} \\
G3 \mapsto \{\pounds epr\} & T94 \mapsto \{\pounds epr\} \\
T08 \mapsto \{\pounds epr\} & T95 \mapsto \{\pounds epr\} \\
T09 \mapsto \{\pounds epr\} & E91 \mapsto \{\pounds epr\}
\end{cases}
$$

À ce niveau d'abstraction, les processus *HealthcareTreatementProcess* et *ClinicalTrialProcess* ont été raffinés en plusieurs processus (appelées lignes de vie dans IAT) : S1, S2, G1, T01, T02, T03, T04, E1, T05, E2 S3, S4, T06, G2, T07, E3, G3, T08, T09, E4, E5, S5, T10, T11, T12, E6, S6, T13, T14, T15 et E7 pour *HealthcareTreatementProcess* et S91, T91, T92, T93, T94, T95, E91 pour *ClinicalTrialProcess*. Les données personnelles correspondent à la variable globale $\pounds epr$. Tous ces processus ont besoin d'utiliser ces données. Les finalités de traitement n'ont pas changé. L'association des finalités est conservée, c'est à dire que les processus raffinant *HealthcareTreatementProcess* gardent la même finalité que P1 (c'est à dire Traitement). Réciproquement pour les processus raffinant *ClinicalTrialProcess*, la même finalité que P2 (c'est à dire Recherche). Le consentement est conservé, c'est à dire que le traitement des données personnelles ($\pounds epr$ à ce niveau d'abstraction) n'est consenti que pour la finalité de traitement médical (Traitement).

A.2.2 . CSpeL - Traces d'exécution

Cette section présente des formalisations en CSpeL de traces d'exécution possibles pour la réalisation du traitement médical et de l'essai clinique.

Traitement médical - Au niveau Modèle Graphique

La trace

$$Handle(HealthcareTreatementProcess, EPR); \epsilon$$

dénote le traitement de EPR par le processus *HealthcareTreatementProcess*, ce qui correspond, à ce niveau-là, à la réalisation du traitement médical.

Traitement médical - Au niveau Modèle Formel

La trace

$Handle(S1, £epr); Handle(S2, £epr); Handle(G1, £epr); Handle(T01, £epr);$
$Handle(T02, £epr); Handle(T03, £epr); Handle(T04, £epr); Handle(E1, £epr);$
$Handle(T05, £epr); Handle(E2, £epr); Handle(S3, £epr); Handle(S4, £epr);$
$Handle(T06, £epr); Handle(G2, £epr); Handle(T07, £epr); Handle(E3, £epr);$
$Handle(G3, £epr); Handle(T08, £epr); Handle(T09, £epr); Handle(E4, £epr);$
$Handle(E5, £epr); Handle(S5, £epr); Handle(T10, £epr); Handle(T11, £epr);$
$Handle(T12, £epr); Handle(E6, £epr); Handle(S6, £epr); Handle(T13, £epr);$
$Handle(T14, £epr); Handle(T15, £epr); Handle(E7, £epr); \epsilon$

dénote la réalisation du traitement médical au niveau Modèle Formel. Comme l'exemple n'explicite pas quels processus traitent des données personnelles, par défaut tous ceux raffinant le processus *HealthcareTreatementProcess* sont considérés comme traitant l'ensemble des données personnelles que traite *HealthcareTreatementProcess* (c'est à dire $£epr$).

Essai Clinique - Au niveau Modèle Graphique

La trace

$$Handle(ClinicalTrialProcess, EPR); \epsilon$$

dénote le traitement de EPR par le processus *ClinicalTrialProcess*, ce qui correspond, à ce niveau-là, à la réalisation de l'essai clinique.

Essai Clinique - Au niveau Modèle Formel

La trace

$Handle(S91, £epr); Handle(T91, £epr); Handle(T92, £epr);$
$Handle(T93, £epr); Handle(T94, £epr); Handle(T95, £epr);$
$Handle(E91, £epr); \epsilon$

dénote la réalisation de l'essai clinique au niveau Modèle Formel. Comme l'exemple n'explicite pas quels processus traitent des données personnelles, par défaut tous ceux raffinant le processus *ClinicalTrialProcess* sont considérés comme traitant l'ensemble des données personnelles que traite *ClinicalTrialProcess* (c'est à dire $\pounds epr$).

A.3 . Exemple Achat en Ligne - version Simple

Cette section regroupe les annexes concernant l'exemple Achat en Ligne - version Simple, défini à la section 7.4.1.

A.3.1 . CSpeL - Formalisation de l'exemple

Il est possible de formaliser l'exemple AL_{Spl} décrit section 7.4.1, d'une façon équivalente à celle réalisée pour l'exemple GDM_{Spl}.

Au niveau Modèle Graphique (BPMN)

$$
\begin{aligned}
S &\triangleq \{P1; P2; P3\} \\
D &\triangleq \{CC\} \\
P &\triangleq \{Administratif; Marketing\} \\
\gamma &\triangleq \begin{cases} (CC, Administratif) & \mapsto & \top \\ (CC, Marketing) & \mapsto & \bot \end{cases} \\
\pi &\triangleq \begin{cases} P1 & \mapsto & \{Administratif\} \\ P2 & \mapsto & \{Administratif\} \\ P3 & \mapsto & \{Marketing\} \end{cases} \qquad
\nu \triangleq \begin{cases} P1 & \mapsto & \{CC\} \\ P2 & \mapsto & \{CC\} \\ P3 & \mapsto & \{CC\} \end{cases}
\end{aligned}
$$

En effet, à ce niveau d'abstraction, trois processus ont été définis : P1 et P2 pour la gestion de l'historique d'achats (Administratif) et P3 pour la gestion d'envoi de publicités ciblées selon l'analyse des achats précédents (Marketing). Les données personnelles sont contenues dans le compte client, CC. Les trois processus auront besoin d'avoir accès à ces données. Dans notre exemple, seules les tâches administratives ont été consenties par la personne concernée (*Data Subject*).

Au niveau Modèle Formel (IAT)

$$
\begin{aligned}
S &\triangleq \{T11; T12; T13; T21; E2; T31; T32\} \\
D &\triangleq \{\pounds CC; \pounds Envoi\} \\
P &\triangleq \{Administratif, Marketing\} \\
\gamma &\triangleq \begin{cases} (\pounds CC, Administratif) & \mapsto \top \\ (\pounds CC, Marketing) & \mapsto \bot \\ (\pounds Envoi, Administratif) & \mapsto \bot \\ (\pounds Envoi, Marketing) & \mapsto \bot \end{cases} \\
\pi &\triangleq \begin{cases} T11 & \mapsto \{Administratif\} \\ T12 & \mapsto \{Administratif\} \\ T13 & \mapsto \{Administratif\} \\ T21 & \mapsto \{Administratif, Marketing\} \\ T31 & \mapsto \{Marketing\} \\ T32 & \mapsto \{Marketing\} \end{cases} \\
\nu &\triangleq \begin{cases} T11 & \mapsto \{\pounds CC\} \\ T12 & \mapsto \{\pounds CC\} \\ T13 & \mapsto \{\pounds CC\} \\ T21 & \mapsto \{\pounds CC\} \\ T31 & \mapsto \{\pounds CC\} \\ T32 & \mapsto \{\pounds Envoi\} \end{cases}
\end{aligned}
$$

À ce niveau d'abstraction, les processus P1, P2 et P3 ont été raffinés en plusieurs processus (appelées lignes de vie dans IAT) : S1, T11, T12, T13, E1 pour P1, S2, T21, T13, E2 pour P2, et S3, T21, T31, T32, E3 pour P3. Or, S1, E1, S2, E2, S3 et E3 ne traitant pas de donnés, ils ne sont pas gardés dans la formalisation. Les données personnelles correspondent aux variables globales $\pounds CC$ et $\pounds Envoi$.

Les processus correspondant au raffinement de P1 et P2, ainsi que le processus T31, ont besoin des données de type $\pounds CC$. Tandis que T32 a besoin des données de type $\pounds Envoi$

Les finalités de traitement n'ont pas changées.

L'association des finalités est conservée, c'est à dire que les processus raffinant P1 gardent la même finalité que P1 (c'est à dire Administratif). Réciproquement les processus raffinant P2, gardent la finalité de P2 (Administratif), et les processus raffinant P3 gardent la finalité de P3 (Marketing).

Il est à noter que T21 fait parti à la fois du raffinement de P2 et de P3, impliquant que T21 soit associé à la fois aux finalités de P2 et P3 (donc Administratif et Marketing). Cela est également le cas pour T13, qui fait parti à la fois du raffinement de P1 et de P2. Cependant, comme P1 et P2 ont la même finalité (Administratif), T21 n'est associé qu'à une seule (Administratif).

Enfin, le consentement est conservé, c'est à dire que le traitement des données personnelles ($\pounds CC$ à ce niveau d'abstraction) n'est consenti que pour la finalité de gestion de l'historique d'achats (Administratif).

Au niveau Programme (C)

$$
\begin{aligned}
S &\triangleq \{\texttt{acheter}; \texttt{enregistrerHistorique}; \texttt{afficher}; \texttt{envoyerPublicité}\};\\
D &\triangleq \{ClientAccount, Client\};\\
P &\triangleq \{Administratif, Marketing\};\\
\gamma &\triangleq \begin{cases}
(ClientAccount, Administratif) \mapsto \top\\
(Client, Administratif) \mapsto \top\\
(ClientAccount, Marketing) \mapsto \bot\\
(Client, Marketing) \mapsto \bot
\end{cases}\\
\pi &\triangleq \begin{cases}
\texttt{acheter} \mapsto \{Administratif\}\\
\texttt{enregistrerHistorique} \mapsto \{Administratif\}\\
\texttt{afficher} \mapsto \{Administratif\}\\
\texttt{accéderCompte} \mapsto \{Administratif; Marketing\}\\
\texttt{accéderHistorique} \mapsto \{Administratif; Marketing\}\\
\texttt{envoyerPublicité} \mapsto \{Marketing\};
\end{cases}\\
\nu &\triangleq \begin{cases}
\texttt{acheter} \mapsto \{ClientAccount, Client\}\\
\texttt{enregistrerHistorique} \mapsto \{ClientAccount\}\\
\texttt{afficher} \mapsto \{ClientAccount\}\\
\texttt{accéderCompte} \mapsto \{ClientAccount, Client\}\\
\texttt{accéderHistorique} \mapsto \{Client\}\\
\texttt{envoyerPublicité} \mapsto \{Client\};
\end{cases}
\end{aligned}
$$

À ce niveau d'abstraction, les processus ont encore été raffinés : T11 en `acheter`, T12 en `enregistrerHistorique`, T13 en `afficher` et `afficherHistorique`, T21 en `accéderCompte` et `accéderHistorique`, T31 en `choisirPublicité`, et T32 en `envoyerPublicité`.

Dans notre exemple simple, seul les processus traitant des données personnelles sont gardés (donc on ne formalisera pas `afficherHistorique` et `choisirPublicité`). Ces processus correspondent à des fonctions dans le code du système.

Les finalités de traitement n'ont pas changées.

Les données personnelles sont raffinées et correspondent aux structures de données $ClientAccount$, $Client$, $PurchaseList$, $Purchase$ et Ad.

Les fonctions `acheter` et `accéderCompte` ont besoin des données de type $ClientAccount$ et de type $Client$, `enregistrerHistorique` et `afficher` des données de type $ClientAccount$, tandis que `accéderHistorique` et `envoyerPublicité` ont besoin des données de type $Client$. L'association des finalités est conservée, c'est à dire que les processus raffinant un autre processus gardent la même finalité. Pour le consentement, dans notre exemple, les traitements des données de type $ClientAccount$ et $Client$ pour la finalité de gestion de l'historique d'achat (Administratif) sont consentis.

A.3.2 . CSpeL - Traces d'exécution

Cette section présente des formalisations en CSpeL de traces d'exécution possibles pour la réalisation d'achats par l'utilisateur, qui sont enregistrés dans son historique (ce qui est lié à la finalité de gestion de l'historique d'achats).

Au niveau Modèle Graphique

La trace

$$Handle(P1, CC); \epsilon$$

dénote le traitement de CC par le processus $P1$, ce qui correspond, à ce niveau-là, à la réalisation d'achats par l'utilisateur, qui sont enregistrés dans son historique.

Au niveau Modèle Formel

La trace

$$Handle(T11, £cc); Handle(T12, £cc); Handle(T13, £cc); \epsilon$$

dénote la réalisation d'achats au niveau Modèle Formel. Les lignes de vie S1, et E1 ne traitant aucune données, elles ne sont pas présentes dans les traces.

Au niveau Programme

La trace

$$Handle(\texttt{acheter}, Client); Handle(\texttt{acheter}, AchatsList);$$
$$Handle(\texttt{acheter}, CompteClient);$$
$$Handle(\texttt{enregistrerHistorique}, CompteClient);$$
$$Handle(\texttt{afficher}, CompteClient);$$
$$Handle(\texttt{afficherHistorique}, AchatsList); \epsilon$$

dénote l'exécution de la fonction `acheter` pour la réalisation d'achats. La fonction `acheter` traite donc trois types de données, $Client$, $AchatsList$ et $CompteClient$. La trace d'exécution de la fonction est composée de trois évènements $Handle(s_i, d_i)$, un pour chaque type de donnée. Pour rappel, nous considérons dans notre exemple que $Client$ et $CompteClient$ sont des données personnelles mais pas $AchatsList$.

A.3.3 . Fichier pour IAT

La figure A.7 et la figure A.8 correspondent au fichier de spécification pour IAT, correspondant au système de l'exemple AL_{Spl} décrit à la section 7.4.1.

```
@signature{
  @lifeline{
    S1; T11; T12; T13; E1; S2; T21; E2; S3; T31; T32; E3
  }
  @message{
    START;
    acheter(Integer);
    enregistrerHistorique(Integer);
    afficherHistorique(Integer);
    accederHistorique(Integer);
    choisirPublicite(Integer);
    envoyerPublicite(Integer);
    END
  }
  @variable{
    CC : Integer; Envoi : Integer;
  }
  @init{
    £CC = 4; £Envoi = 5;
  }
}
@option{
  /* Declaring options for various processes proposed by the tool */
  @explore_option{
    /* Options for the exploration process which
    *  explores the semantics of the model
    *  following certain heuristics */
    @general_option{
      loggers = [graphic[svg,vertical]]; strategy = BreadthFS
    }
  }
  @analyze_option{
    /* Options for the analysis process which
    *  evaluates the conformity of a (multi-)trace
    *  against the model */
    @general_option{
      loggers = [graphic[svg,vertical]];
      strategy = DepthFS
    };
    analysis_kind = prefix; local_analysis = false;
    goal = WeakPass
  }
}
```

Figure A.7 – Fichier de spécification pour IAT pour l'exemple AL_{Spl} (partie 1).

```
@model{
  loopH(
    alt(
      seq(
        S1 -- START -> T11,
        T11 -- send(£CC) -> T12, /* acheter */
        T12 -- send(£CC) -> T13, /* enregistrerHistorique */
        T13 -- send(£CC) -> E1, /* afficherHistorique */
        E1 -- END -> |
      ),
      seq(
        S2 -- START -> T21,
        T21 -- send(£CC) -> T13, /* accederHistorique */
        T13 -- send(£CC) -> E2, /* afficherHistorique */
        E2 -- END -> |
      ),
      seq(
        S3 -- START -> T21,
        T21 -- send(£CC) -> T31, /* accederHistorique */
        T31 -- send(£CC) -> T32, /* choisirPublicite */
        T32 -- send(£Envoi) -> E3, /* envoyerPublicite */
        E3 -- END -> |
      )
    )
  )
}
```

Figure A.8 – Fichier de spécification pour IAT pour l'exemple AL_{Spl} (partie 2).

A.3.4 . Fichier instrumenté pour IAT

La figure A.9 et la figure A.10 présentent le fichier de l'exemple GDM_{Spl} instrumenté pour la vérification des propriétés de consentement avec IAT (propriétés définies section 8.1.3), pour l'exemple GDM_{Spl}. Cette instrumentation suit le principe de traduction expliqué en section 9.3.1.

```
@signature{
  @lifeline{
    S1; T11; T12; T13; E1; S2; T21; E2; S3; T31; T32; E3
  }
  @message{
    START;
    acheter(Integer);
    enregistrerHistorique(Integer);
    afficherHistorique(Integer);
    accederHistorique(Integer);
    choisirPublicite(Integer);
    envoyerPublicite(Integer);
    END
  }
  @variable{
    CC : Integer;
    Envoi : Integer;
    CC_Administratif : Bool;
    CC_Marketing : Bool;
    Envoi_Administratif : Bool;
    Envoi_Marketing : Bool
  }
  @init{
    £CC = 4;
    £Envoi = 5;
    £CC_Administratif = ⊤;
    £CC_Marketing = ⊥;
    £Envoi_Administratif = ⊥;
    £Envoi_Marketing = ⊥
  }
}
@option{
  /* Declaring options for various processes proposed by the tool */
  @explore_option{
    /* Options for the exploration process which
    *  explores the semantics of the model
    *  following certain heuristics */
    @general_option{
      loggers = [graphic[svg,vertical]]; strategy = BreadthFS
    }
  }
  @analyze_option{
    /* Options for the analysis process which
    * evaluates the conformity of a (multi-)trace
    * against the model */
    @general_option{
      loggers = [graphic[svg,vertical]]; strategy = DepthFS
    };
    analysis_kind = prefix; local_analysis = false; goal = WeakPass
  }
}
```

Figure A.9 – Fichier IAT pour l'exemple AL_{Spl} instrumenté avec le consentement (partie 1).

```
@model{
  loopH(
    alt(
      seq(
        S1 -- START -> T11,
        T11[£CC_Administratif] -- acheter(£CC) -> T12,
        T12[£CC_Administratif] -- enregistrerHistorique(£CC) -> T13,
        T13[£CC_Administratif] -- afficherHistorique(£CC) -> E1,
        E1 -- END -> |
        ),
      seq(
        S2 -- START ->T21,
        T21[£CC_Administratif] -- accederHistorique(£CC) -> T13,
        T13[£CC_Administratif] -- afficherHistorique(£CC) -> E2,
        E2 -- END -> |
      ),
      seq(
        S3 -- START ->T21,
        T21[£CC_Marketing] -- accederHistorique(£CC) -> T31,
        T31[£CC_Marketing] -- choisirPublicite(£CC) -> T32,
        T32[£Envoi_Marketing] -- envoyerPublicite(£Envoi) -> E3,
        E3 -- END -> |
      )
    )
  )
}
```

Figure A.10 – Fichier IAT pour l'exemple AL_{Spl} instrumenté avec le consentement (partie 2).

A.3.5 . Code source C

Les figures A.11 et A.12 présentent les différentes parties d'un fichier source complet (données, fonctions, main) pour notre exemple AL_{Spl}.

```c
#include <stdio.h>

#define S_SIZE 20
#define NB_Purchase 20

/*--  DATA --*/
typedef struct {
    int id;
    char content[S_SIZE];
} Ad;

typedef struct {
    char name[S_SIZE];
    int price;
} Purchase;

typedef struct {
    char date[S_SIZE];
    Purchase purchaseList[NB_Purchase];
    int total;
} PurchaseList;

typedef struct {
    int id;
    char name[S_SIZE];
    char lastname[S_SIZE];
    char birthdate[S_SIZE];
    char email[S_SIZE];
} Client;

typedef struct {
    Client client;
    PurchaseList history[NB_Purchase];
} ClientAccount;
```

Figure A.11 – AL_{Spl} - Exemple de fichier source - Données.

```
/*--  FUNCTIONS --*/
ClientAccount buy(Client c, PurchaseList a){  ...  }
void store(ClientAccount cc){  ...  }

void displayHistory(PurchaseList* a){  ... }

void display(ClientAccount cc){  ...  }

ClientAccount getCompte(Client c){  ...  }

PurchaseList* getHistory(Client c){  ...  }

Ad chooseAd(PurchaseList* purchases){  ...  }

void sendAd(Ad p, Client c){  ...  }

/*--  MAIN --*/
int main(){
    Client client = {.name="John", .lastname="Doe"};
    PurchaseList purchases = {.date = ..., ...};
    /*-- P1  --*/
    ClientAccount cc = buy(client, purchases);
    store(cc);
    display(cc);
    return 0;
}
```

Figure A.12 – AL$_{Spl}$ - Exemple de fichier source - Fonctions.

A.3.6 . Code généré

Fig. A.13, Fig. A.14, Fig. A.15 correspond à un example de fichier C généré par CASTT, à partir d'une spécification de context écrit en CSpeL et d'un fichier source écrit en C. Pour des raisons de lisibilités, certaines parties dans le code ont été remplacées par "..." dans l'exemple (cela ne concerne que la génération de code C par by Frama-C, et non les informations relatives à notre approche).

```
/* Generated by Frama-C */
#include ...
...
enum _Purposes {
    MARKETING = 0,
    ADMINISTRATIVE = 1
};
typedef enum _Purposes Purposes;
enum _PersonalData {
    CLIENT = 0,
    CLIENTACCOUNT = 1
};
typedef enum _PersonalData PersonalData;
/*@ ghost int \ghost Consent[2][2]; */

/*@ ghost int Meanings[2]; */

/*@ ghost int nbMeanings; */

struct __anonstruct_Ad_1 {
   int id ;
   char content[20] ;
};
typedef struct __anonstruct_Ad_1 Ad;
struct __anonstruct_Purchase_2 {
   char name[20] ;
   int price ;
};
typedef struct __anonstruct_Purchase_2 Purchase;
struct __anonstruct_PurchaseList_3 {
   char date[20] ;
   Purchase purchaseList[20] ;
   int total ;
};
typedef struct __anonstruct_PurchaseList_3 PurchaseList;
struct __anonstruct_Client_4 {
   int id ;
   char name[20] ;
   ...
};
typedef struct __anonstruct_Client_4 Client;
struct __anonstruct_ClientAccount_5 {
   Client client ;
   PurchaseList history[20] ;
};
typedef struct __anonstruct_ClientAccount_5 ClientAccount;
```

Figure A.13 – AL_{Spl} - Exemple de fichier généré - Données.

```
/*@ requires
      Consent[CLIENTACCOUNT][ADMINISTRATIVE] ≡ 1 ∧
      Consent[CLIENT][ADMINISTRATIVE] ≡ 1;
    assigns \nothing;
 */
ClientAccount buy(Client c, PurchaseList a)
{
  ClientAccount cc = {.client = c, .history = {a}};
  return cc;
}

/*@ requires Consent[CLIENTACCOUNT][ADMINISTRATIVE] ≡ 1;
    assigns \nothing; */
void store(ClientAccount cc)
{
  return;
}

/*@ requires \true;
    assigns \nothing; */
void displayHistory(PurchaseList *a)
{
  return;
}

/*@ requires Consent[CLIENTACCOUNT][ADMINISTRATIVE] ≡ 1;
    assigns \nothing; */
void display(ClientAccount cc)
{
  displayHistory(cc.historique);
  return;
}

/*@ requires
      (Consent[CLIENTACCOUNT][ADMINISTRATIVE] ≡ 1 ∧
       Consent[CLIENT][ADMINISTRATIVE] ≡ 1) ∨
      (Consent[CLIENTACCOUNT][MARKETING] ≡ 1 ∧
       Consent[CLIENT][MARKETING] ≡ 1);
    assigns \nothing;
 */
ClientAccount getCompte(Client c)
{
  ClientAccount cc =
    {.client = c,
     .history = {{.date = {...}, ...}, ...}
    };

  return cc;
}

/*@ requires
      Consent[CLIENT][ADMINISTRATIVE] ≡ 1 ∨
      Consent[CLIENT][MARKETING] ≡ 1;
    assigns \nothing;
 */
PurchaseList *getHistory(Client c)
{
  PurchaseList *__retres;
  ClientAccount cc = getCompte(c);
  __retres = cc.historique;
  return __retres;
}
```

Figure A.14 – AL$_{Spl}$ - Exemple de fichier généré - Fonctions part 1.

```
/*@ requires \true;
    assigns \nothing; */
Ad chooseAd(PurchaseList *purchases)
{
  Ad p = {.id = ..., .content = {...}};
  return p;
}

/*@ requires Consent[CLIENT][MARKETING] == 1;
    assigns \nothing; */
void sendAd(Ad p, Client c)
{
  return;
}

int main(void)
{
  int __retres;
  /*@ ghost Consent[CLIENTACCOUNT][MARKETING] = 0; */
  /*@ ghost Consent[CLIENTACCOUNT][ADMINISTRATIVE] = 1; */
  /*@ ghost Consent[CLIENT][MARKETING] = 0; */
  /*@ ghost Consent[CLIENT][ADMINISTRATIVE] = 1; */
  Client client = {.id = ..., .name = {...}, ...};
  PurchaseList purchases = {.date = {..},...};
  ClientAccount cc = buy(client,purchases);
  store(cc);
  display(cc);
  __retres = 0;
  return __retres;
}
```

Figure A.15 – AL_{Spl} - Exemple de fichier généré - Fonctions part 2.

A.4 . Exemple Achat en Ligne - version Complexe

Cette section regroupe les annexes concernant l'exemple Achat en Ligne - version Complexe, défini à la section 7.4.2.

A.4.1 . CSpeL - Formalisation de l'exemple

Il est possible de formaliser l'exemple AL_{Cpx} décrit section 7.4.2, d'une façon équivalente à celle réalisée pour l'exemple AL_{Spl}. Cet exemple étant une extension de l'exemple Achat en Ligne - version Simple, seul le niveau programme change (comme expliqué en section 7.4.2). La fonction `actionUser` est ajoutée, comme elle englobe les fonctions associées à la finalité *Administratif* et celles associées à la finalité *Marketing*, `actionUser` est associée aux deux.

Au niveau Programme (C)

$$
\begin{aligned}
S &\triangleq \{\texttt{actionUser}; \texttt{buy}; \texttt{store}; \texttt{display}; \texttt{sendAd}\};\\
D &\triangleq \{ClientAccount, Client\};\\
P &\triangleq \{Administratif, Marketing\};\\
\gamma &\triangleq \begin{cases}
(ClientAccount, Administratif) & \mapsto \top\\
(Client, Administratif) & \mapsto \top\\
(ClientAccount, Marketing) & \mapsto \bot\\
(Client, Marketing) & \mapsto \bot
\end{cases}\\
\pi &\triangleq \begin{cases}
\texttt{actionUser} & \mapsto \{Administratif; Marketing\}\\
\texttt{buy} & \mapsto \{Administratif\}\\
\texttt{store} & \mapsto \{Administratif\}\\
\texttt{display} & \mapsto \{Administratif\}\\
\texttt{getAccount} & \mapsto \{Administratif; Marketing\}\\
\texttt{getHistory} & \mapsto \{Administratif; Marketing\}\\
\texttt{sendAd} & \mapsto \{Marketing\};
\end{cases}\\
\nu &\triangleq \begin{cases}
\texttt{actionUser} & \mapsto \{Client\}\\
\texttt{buy} & \mapsto \{ClientAccount, Client\}\\
\texttt{store} & \mapsto \{ClientAccount\}\\
\texttt{display} & \mapsto \{ClientAccount\}\\
\texttt{getAccount} & \mapsto \{ClientAccount, Client\}\\
\texttt{getHistory} & \mapsto \{Client\}\\
\texttt{sendAd} & \mapsto \{Client\};
\end{cases}
\end{aligned}
$$

A.4.2 . CSpeL - Traces d'exécution

Cette section présente des formalisations en CSpeL de traces d'exécution possibles pour la réalisation d'achats par l'utilisateur, qui sont enregistrés dans son historique, pour la consultation de son historique par un utilisateur, et pour l'envoi de publicités ciblées, selon l'analyse des achats précédents.

Trace pour la réalisation d'achats - Au niveau Programme

La trace

$$Handle(\texttt{actionUser}, Client); Handle(\texttt{actionUser}, AchatsList);$$
$$Handle(\texttt{buy}, Client); Handle(\texttt{buy}, AchatsList);$$
$$Handle(\texttt{buy}, CompteClient); Handle(\texttt{store}, CompteClient);$$
$$Handle(\texttt{display}, CompteClient);$$
$$Handle(\texttt{displayHistory}, AchatsList); \epsilon$$

dénote l'exécution des fonctions `actionUser`, `buy`, `display` et `displayHistory` pour la réalisation d'achats.

Trace pour la consultation d'historique - Au niveau Programme

La trace

$$Handle(\texttt{actionUser}, Client); Handle(\texttt{actionUser}, AchatsList);$$
$$Handle(\texttt{getHistory}, Client); Handle(\texttt{getHistory}, AchatsList);$$
$$Handle(\texttt{displayHistory}, AchatsList); \epsilon$$

dénote l'exécution des fonctions `actionUser`, `getHistory`, et `displayHistory` pour la consultation d'historique.

Trace pour l'envoi de publicités ciblées - Au niveau Programme

La trace

$$Handle(\texttt{actionUser}, Client); Handle(\texttt{actionUser}, AchatsList);$$
$$Handle(\texttt{getHistory}, Client); Handle(\texttt{getHistory}, AchatsList);$$
$$Handle(\texttt{chooseAd}, AchatsList); Handle(\texttt{chooseAd}, Publicite); \epsilon$$

dénote l'exécution des fonctions `actionUser`, `getHistory`, et `chooseAd` pour l'envoi de publicités ciblées.

A.5 . Requête pour la Systematic Mapping Study

Cette annexe correspond à la requête, présentée en figure A.16, utilisée pour le *Systematic Mapping Study* que j'ai réalisé à l'aide de GePyR, détaillé en section 4.4.

```
( "software engineering" OR systems OR software OR applications OR
  apps OR database OR architecture OR devices OR network OR
  programs OR code )
AND
( ( privacy OR "data protection" OR GDPR OR "Privacy Act" OR
    privacy-related OR privacy-type OR "privacy by design" OR
    "personal data" )
  OR
  ( ( Linkability OR Identifiability OR Non-Repudiation OR
      Detectability OR "Disclosure of Information" OR
      "Content Unawareness" OR "Policy and Consent Non-Compliance" )
    OR
    ( "Information Monitoring" OR "Information Aggregation" OR
      "Information Storage" OR "Information Transfer" OR
      "Information Collection" OR "Information Personalization" OR
      Solicitation )
    OR
    ( Surveillance OR Interrogation OR Aggregation OR Identification OR
      Insecurity OR "Secondary Use" OR Exclusion OR
      "Breach of Confidentiality" OR Disclosure OR Exposure OR
      "Increased Accessibility" OR Blackmail OR
      Appropriation OR Distortion OR Intrusion OR
      "Decisional Interference" )
    OR
    ( "Lawfulness, Fairness and Transparency" OR
      "Purpose Limitation" OR "Data Minimization" OR Accuracy OR
      "Storage Limitation" OR "Integrity and Confidentiality" OR
      Accountability OR "Data Subject Rights" OR
      "Right to Be Forgotten" OR "Data Quality" OR
      "Adequate protection" OR "Data Portability" OR Transparency OR
      "Data Breach Notification" OR
      "Accountability and (Provable) Compliance" )
    OR
    ( "Notice/Awareness" OR "Enforcement/Redress" OR
      "Access/Participation" OR "Integrity/Security" OR
      "Choice/Consent" OR "Collusion Resistance" OR
      "Mutual Authentication" OR "Consent Revocation" OR
      "Contextual Privacy" OR "Data Confidentiality and Integrity" )
    OR
    ( Anonymity OR Unlinkability OR Undetectability OR
      Unobservability OR "Information-Flow Security Policies" OR
      Non-Interference OR "Purpose Binding" OR
      "Data Subject Consent" OR
      "Necessity of Data Collection and Processing" OR
      "Observational Equivalence" ) ) )
)
AND
( enhancing OR ensuring OR verification OR formalisation OR
  implementation ORspecification OR protection OR compliance )
```

Figure A.16 – Requête utilisée pour ma Systematic Mapping Study.

A.6 . Liens entre les définitions réécrites et les définitions d'origine dans les articles de la littérature

Cette annexe contient des tables présentant les liens entre les définitions réécrites (adaptées) des représentations de la littérature du chapitre 3 et leur définitions originales dans les papiers sources. Au vu de la taille de ces définitions, elles sont séparées en plusieurs tables : le tableau A.1, le tableau A.2, le tableau A.3 et le tableau A.4.

Table A.1 – Liens entre les définitions réécrites et les définitions d'origine dans les articles de la littérature - Partie 1.

		REF	NAME	ADAPTED DEFINITION	ORIGINAL DEFINITION
REPRESENTATION TYPE	Vulnerabilities	[11]	***Information Monitoring***	reflète l'existence d'un suivi des informations par les organisations lorsque les clients visitent leur site web	reflect the occurrence information tracking by organisations when consumers visit their Web site
			Information Aggregation	se réfère à la combinaison d'informations personnelles identifiables (*Personally Identifiable Information*, ou PII) collectées précédemment avec des données provenant d'autres sources	refers to combining previously gathered PII data with data from other sources
			Information Storage	reflète quels enregistrements sont stockés dans la base de données d'une organisation	reflect how and what records are stored in an organisation's database.
			Information Transfer	reflète la pratique consistant à autoriser la transmission d'informations, les raisons pour lesquelles elles peuvent être transférées, et à qui elles le sont	reflect the practice of allowing information to be transmitted, the reason(s) why information may be transferred, and to whom that information is transferred
			Information Personalization	reflète l'adaptation ou la personnalisation d'un site web à un visiteur spécifique, affectant ainsi la fonctionnalité ou le contenu offert aux visiteurs	reflect the tailoring or customisation of a Web site to a specific visitor, thus affecting the functionality or content offered to individual visitors
			Solicitation	reflète comment et dans quel but les organisations contactent les visiteurs ou d'autres personnes	reflect how and for what purpose organisations contact visitors or others
	Harmful Activities	[159]	***Surveillance***	est l'action de regarder, écouter, ou enregistrer les activités d'un individu	is the watching, listening to, or recording of an individual's activities
			Identification	relier des informations aux individus	is linking information to particular individuals

Table A.2 – Liens entre les définitions réécrites et les définitions d'origine dans les articles de la littérature - Partie 2.

		REF	NAME	ADAPTED DEFINITION	ORIGINAL DEFINITION
REPRESENTATION TYPE	Threats	[50]	***Linkability***	d'au moins deux éléments d'intérêt (*items of interest*, ou IOIs, par exemple des sujets, des messages, des actions, etc.), permet à un attaquant de distinguer suffisamment quand ces IOIs sont reliés ou non dans le système	of two or more items of interest (IOIs, e.g., subjects, messages, actions, etc.) allows an attacker to sufficiently distinguish whether these IOIs are related or not within the system
	Goals	[74]	***Purpose Limitation***	La finalité des données est précisée et l'utilisation des données est limitée à cette finalité	comprising both specification of the purpose and limiting the use to that stated purpose
			Data Subject Rights	Les personnes concernées ont le droit de consulter, d'effacer et de rectifier leurs données à caractère personnel	in terms of consent, and the right to view, erase, and rectify personal data
			Data Minimization	La collecte de données à caractère personnel doit être réduite au minimum, tant en ce qui concerne la quantité que le nombre d'acteurs qui y ont accès. Par défaut des options qui ne portent pas atteinte à la vie privée doivent être proposées et les données doivent être supprimées une fois qu'elles ne sont plus nécessaires	minimise the amount of personal data collected, minimise the number of actors that have access, offer as default non-privacy invasive options, and delete data once it has become no longer necessary
			Data Portability	Les personnes concernées ont le droit d'obtenir du responsable du traitement une copie des données faisant l'objet du traitement dans un format électronique et structuré, couramment utilisé et permettant leur utilisation ultérieure	allowing them 'to obtain from the controller a copy of data undergoing processing in an electronic and structured format which is commonly used and allows for further use by the data subject'
			Accountability and (Provable) Compliance	Le responsable du traitement des données est tenu de respecter ces principes et doit être en mesure de prouver qu'il respecte la réglementation	A data controller must be accountable for complying with these principles

Table A.3 – Liens entre les définitions réécrites et les définitions d'origine dans les papiers de la littérature - Partie 3.

		REF	NAME	ADAPTED DEFINITION	ORIGINAL DEFINITION
REPRESENTATION TYPE	Goals	[136]	***Right to Be Forgotten***	Les personnes concernées doivent avoir le droit de demander au responsable du traitement la rectification, l'effacement des données à caractère personnel ou la limitation du traitement des données à caractère personnel les concernant, ou encore le droit de s'opposer à un tel traitement	The data subject shall have the right to obtain from the controller the erasure of personal data relating to them and the abstention from further dissemination of such data, especially in relation to personal data which are made available by the data subject while he or she was a child, where one of the following grounds applies
		[187]	***Data Confidentiality and Integrity***	garantit que les données ne peuvent pas être accédées par d'autres entités sans le consentement de l'utilisateur et que les données doivent être protégées des modifications	The data cannot be accessed by other entities without user consent. Additionally, the data should be protected from modification
			Contextual Privacy	permet aux demandeurs de données ("*data requesters*") de demander les informations sur la santé de l'utilisateur au centre de données ("*data center*") avec le consentement de l'utilisateur, alors que le fournisseur de données ("*data provider*") ne sait pas à qui appartiennent les informations demandées et garantit que les observateurs de la transmission des données ne peuvent pas déduire quelles informations demandées sont nécessaires au demandeur de données	On one hand, data requesters are allowed to request the user's health information from the data center with the user's consent while the data provider is unaware of whose information was requested. On the other hand, any observers of the data transmission cannot derive which information is needed by the data requester
			Consent Revocation	affirme que les utilisateurs ont le privilège de mettre fin au consentement à tout moment avant son expiration	Users have the privilege to terminate consent at any time before expiration of the consent

Table A.4 – Liens entre les définitions réécrites et les définitions d'origine dans les articles de la littérature - Partie 4.

		REF	NAME	ADAPTED DEFINITION	ORIGINAL DEFINITION
REPRESENTATION TYPE	Goals	[187]	***Mutual Authentication***	affirme que le fournisseur de données et le centre de données sont en mesure de s'authentifier mutuellement, que l'utilisateur ne peut donner accès qu'à ses propres données et que le centre de données est en mesure d'authentifier le fournisseur de consentement	The data provider and the data center should be able to authenticate each other. Additionally, the user can only provide access to their own data. Consequently, the data center should be able to authenticate the consent provider
			Collusion Resistance	permet au centre de données de re-chiffrer les données de l'utilisateur demandées par le demandeur après avoir reçu d'un utilisateur la clé de re-chiffrement destinée à un demandeur de données et affirme que le demandeur de données ne peut pas décrypter les autres données de l'utilisateur qui sont au-delà du consentement, même s'il est de connivence avec le centre de données	After receiving the re-encryption key intended for a data requester from a user, the data center can re-encrypt the user's data asked by the requester. However, the data requester cannot decrypt user's other data which is beyond the consent even if she colludes with the data center
	Properties	[49]	***Observational Equivalence***	signifie que deux processus ne peuvent pas être distinguables par un quelconque attaquant	a context may represent an attacker, and two processes are observationally equivalent if they cannot be distinguished by any attacker
		[60]	***Necessity of Data Collection and Processing***	n'autorise la collecte et le traitement de données à caractère personnel que lorsqu'ils sont nécessaires à l'accomplissement de tâches relevant de la responsabilité de l'agence de traitement des données	the collection and processing of personal data shall only be allowed, if it is necessary for tasks falling within the responsibility of the data processing agency
		[18]	***Data Protection***	exige le chiffrement des données pour toute activité de traitement des données à caractère personnel dans le cadre d'un relation commerciale	for every processing activity on personal data within a business process, the encryption of data is required

A.7 . Scénarios pour l'évaluation de l'analyse de trace de CASTT

Cette section présente les scénarios utilisés pour évaluer le mécanisme d'analyse de trace de CASTT, sur les exemples GDM$_{Cpx}$ et AL$_{Spl}$. Ces scénarios représentent différentes traces d'exécution du système considéré, dans un contexte correspondant à la définition de l'exemple. Les traces et le contexte sont spécifiés en CSpeL. Ces scénarios sont présentés dans des tables qui présentent, pour chaque exemple, les spécifications de contexte ainsi que la trace T_i utilisée selon le niveau d'abstraction (LVL), puis le nombre total de scénarios utilisés pour l'exemple. Les spécifications de contexte sont communes à chaque niveau d'abstraction (ex : un même contexte pour toutes les traces du niveau Modèle Graphique de l'exemple GDM$_{Spl}$).

A.7.1 . Évaluation sur l'exemple Gestion Données Médicales - version Complexe

Les scénarios détaillés dans le tableau A.5, ont été établis pour l'exemple GDM$_{Cpx}$, défini en section 7.3.2.

Table A.5 – Table des scénarios pour l'exemple GDM$_{Cpx}$.

LVL	\context	\trace	

ML	\process{
	HealthcareTreatmentProcess
	ClinicalTrialProcess},
	\personalData{EPR},
	\purposes{Treatment, Research},
	\isGranted{(EPR : Treatment)},
	\hasPurposes{
	(HealthcareTreatmentProcess :{Treatment}),
	(ClinicalTrialProcess :{Research})},
	\needData{
	(HealthcareTreatmentProcess :{EPR}),
	(ClinicalTrialProcess :{EPR})}
		T1	\handle(HealthcareTreatmentProcess, EPR);
			VOID

		T2	\handle(ClinicalTrialProcess, EPR);VOID
		T3	\handle(HealthcareTreatmentProcess, EPR); \handle(HealthcareTreatmentProcess, EPR); VOID
		T4	\handle(ClinicalTrialProcess, EPR); \handle(HealthcareTreatmentProcess, EPR); VOID
		T5	\handle(HealthcareTreatmentProcess, EPR); \handle(ClinicalTrialProcess, EPR); VOID
ML formel	\process{ S1,S2,G1,T01,T02,T03,T04,E1,T05,E2, S3,S4,T06,G2,T07,E3,G3,T08,T09,E4,E5, S5,T10,T11,T12,E6, S6,T13,T14,T15,E7, S91,T91,T92,T93,T94,T95,E91}, \personalData{EPR}, \purposes{Treatment, Research}, \isGranted{(EPR : Treatment)}, \hasPurposes{(S1 :{Treatment}),(S2 :{Treatment}), (G1 :{Treatment}),(T01 :{Treatment}), (T02 :{Treatment}),(T03 :{Treatment}), (T04 :{Treatment}),(E1 :{Treatment}), (T05 :{Treatment}),(E2 :{Treatment}), (S3 :{Treatment}),(S4 :{Treatment}), (T06 :{Treatment}),(G2 :{Treatment}), (T07 :{Treatment}),(E3 :{Treatment}), (G3 :{Treatment}),(T08 :{Treatment}), (T09 :{Treatment}),(E4 :{Treatment}), (E5 :{Treatment}),(S5 :{Treatment}), (T10 :{Treatment}),(T11 :{Treatment}), (T12 :{Treatment}),(E6 :{Treatment}), (S6 :{Treatment}),(T13 :{Treatment}), (T14 :{Treatment}),(T15 :{Treatment}), (E7 :{Treatment}),(S91 :{Research}), (T91 :{Research}),(T92 :{Research}), (T93 :{Research}),(T94 :{Research}), (T95 :{Research}),(E91 :{Research})}, \needData{(S1 :{EPR}),(S2 :{EPR}),(G1 :{EPR}), (T01 :{EPR}),(T02 :{EPR}),(T03 :{EPR}),(T04 :{EPR}), (E1 :{EPR}),(T05 :{EPR}),(E2 :{EPR}),(S3 :{EPR}), (S4 :{EPR}),(T06 :{EPR}),(G2 :{EPR}),(T07 :{EPR}), (E3 :{EPR}),(G3 :{EPR}),(T08 :{EPR}),(T09 :{EPR}), (E4 :{EPR}),(E5 :{EPR}),(S5 :{EPR}),(T10 :{EPR}), (T11 :{EPR}),(T12 :{EPR}),(E6 :{EPR}),(S6 :{EPR}), (T13 :{EPR}),(T14 :{EPR}),(T15 :{EPR}),(E7 :{EPR}), (S91 :{EPR}),(T91 :{EPR}),(T92 :{EPR}),(T93 :{EPR}), (T94 :{EPR}),(T95 :{EPR}),(E91 :{EPR})}	T1	\handle(S1, EPR); \handle(S2, EPR); \handle(G1, EPR); \handle(T01, EPR); \handle(T02, EPR); \handle(T03, EPR); \handle(T04, EPR); \handle(E1, EPR); \handle(T05, EPR); \handle(E2, EPR); VOID
		T2	\handle(S3, EPR); \handle(S4, EPR); \handle(T06, EPR); \handle(G2, EPR); \handle(T07, EPR); \handle(E3, EPR); \handle(G3, EPR); \handle(T08, EPR); \handle(T09, EPR); \handle(E4, EPR); \handle(E5, EPR); VOID
		T3	\handle(S5, EPR); \handle(T10, EPR); \handle(T11, EPR); \handle(T12, EPR); \handle(E6, EPR); \handle(S91, EPR); \handle(T91, EPR); \handle(T92, EPR); \handle(T93, EPR); \handle(T94, EPR); \handle(T95, EPR); \handle(E91, EPR); VOID
		T4	\handle(S91, EPR); \handle(T91, EPR); \handle(T92, EPR); \handle(T93, EPR); \handle(T94, EPR); \handle(T95, EPR); \handle(E91, EPR); VOID
		T5	\handle(S1, EPR); \handle(S2, EPR); \handle(G1, EPR); \handle(T01, EPR); \handle(T02, EPR); \handle(T03, EPR); \handle(T04, EPR); \handle(E1, EPR); \handle(T05, EPR); \handle(E2, EPR); \handle(S1, EPR); \handle(S2, EPR); \handle(G1, EPR); \handle(T01, EPR); \handle(T02, EPR); \handle(T03, EPR); \handle(T04, EPR); \handle(E1, EPR); \handle(T05, EPR); \handle(E2, EPR); VOID
Total	10		

A.7.2 . Évaluation sur l'exemple Achat en Ligne - version Simple

Les scénarios détaillés dans les tableaux A.6, A.7 et A.8, ont été établis pour l'exemple AL_{Spl}, défini en section 7.4.1.

Pour des raisons de place, les traces utilisées dans les scénarios pour cet exemple sont détaillées dans plusieurs tables selon le niveau d'abstraction considéré. Dans la table A.6 pour le niveau Modèle Graphique et Modèle Formel, dans les tables A.7 et A.8 pour le niveau Programme.

Table A.6 – Table des scénarios pour l'exemple AL_{Spl} - ML & ML formel.

LVL	\context	\trace	
ML	\process{P1,P2,P3}, \personalData{CC,Send}, \purposes{Administrative, Marketing}, \isGranted{(CC : Administrative)}, \hasPurposes{(P1 :{Administrative}), (P2 :{Administrative}), (P3 :{Marketing})}, \needData{(P1 :{CC}), \needData{(P2 :{CC}), (P3 :{CC,Send})}	T1	\handle(P1, CC); VOID
		T2	\handle(P2, CC); VOID
		T3	\handle(P3, CC); \handle(P3, Send); VOID
		T4	\handle(P1, CC); \handle(P2, CC); VOID
		T5	\handle(P1, CC); \handle(P3, CC); \handle(P2, CC); \handle(P3, Send); VOID
ML formel	\process{T11, T12, T13, T21, T31, T32}, \personalData{ CC,Sending}, \purposes{Administratif,Marketing}, \isGranted{(CC : Administratif)}, \hasPurposes{(T11 :{Administratif}), (T12 :{Administratif}), (T13 :{Administratif}), (T21 :{Administratif, Marketing}), (T31 :{Marketing}), (T32 :{Marketing})} \needData{(T11 :{CC}),(T12 :{CC}), (T13 :{CC}),(T21 :{CC}), (T31 :{CC}),(T32 :{Sending})}	T1	\handle(T11, CC); \handle(T12, CC); \handle(T13, CC); VOID
		T2	\handle(T21, CC); \handle(T13, CC); VOID
		T3	\handle(T21, CC); \handle(T31, CC); \handle(T32, Sending); VOID
		T4	\handle(T11, CC); \handle(T12, CC); \handle(T13, CC); \handle(T21, CC); \handle(T13, Sending); VOID
Total	9		

Table A.7 – 1re Table des scénarios pour l'exemple AL_{Spl} - PL

LVL	\context	\trace	
PL	\process{buy,store,display, getAccount,getHistory,chooseAd,sendAd}, \personalData{ClientAccount,Client}, \purposes{Administrative, Marketing}, \isGranted{(ClientAccount : Administrative), (Client : Administrative)} \hasPurposes{(buy :{Administrative}), (store :{Administrative}), (display :{Administrative}), (getAccount :{Administrative,Marketing}), (getHistory :{Administrative,Marketing}) (sendAd :{Marketing})}, \needData{(buy :{ClientAccount,Client}), (store :{ClientAccount}), (display :{ClientAccount}), (getAccount :{ClientAccount,Client}), (getHistory :{Client}),(sendAd :{Client})}	T1	\handle(buy, Client); \handle(buy, PurchaseList); \handle(buy, ClientAccount); \handle(store, ClientAccount); \handle(getHistory, Client); \handle(display, ClientAccount); VOID
		T2	\handle(getAccount, Client); \handle(getAccount, ClientAccount); \handle(getHistory, Client); \handle(getHistory, PurchaseList); \handle(display, ClientAccount); VOID
		T3	\handle(getAccount, Client); \handle(getAccount, ClientAccount); \handle(getHistory, Client); \handle(getHistory, PurchaseList); \handle(chooseAd, PurchaseList); \handle(chooseAd, Ad); \handle(sendAd, Ad); \handle(sendAd, Client); VOID
		T4	\handle(getAccount, Client); \handle(getHistory, Client); \handle(getHistory, PurchaseList); \handle(chooseAd, PurchaseList); \handle(chooseAd, Ad); \handle(sendAd, Ad); \handle(sendAd, Client); \handle(buy, Client); \handle(buy, PurchaseList); \handle(buy, ClientAccount); \handle(store, ClientAccount); \handle(getHistory, Client); \handle(display, ClientAccount); VOID
		T5	\handle(buy, Client); \handle(buy, PurchaseList); \handle(buy, ClientAccount); \handle(store, ClientAccount); \handle(getHistory, Client); \handle(display, ClientAccount); \handle(getAccount, Client); \handle(getAccount, ClientAccount); \handle(getHistory, Client); \handle(getHistory, PurchaseList); \handle(display, ClientAccount); VOID
	\trace		
T6	\handle(getAccount, Client); \handle(getAccount, ClientAccount); \handle(getHistory, Client); \handle(getHistory, PurchaseList); \handle(chooseAd, PurchaseList); \handle(chooseAd, Ad); \handle(sendAd, Ad); \handle(sendAd, Client); \handle(getAccount, Client); \handle(getAccount, ClientAccount); \handle(getHistory, Client); \handle(getHistory, PurchaseList); \handle(chooseAd, PurchaseList); \handle(chooseAd, Ad); \handle(sendAd, Ad); \handle(sendAd, Client);VOID		
T7	\handle(getAccount, Client); \handle(getAccount, ClientAccount); \handle(getHistory, Client); \handle(getHistory, PurchaseList); \handle(display, ClientAccount); \handle(buy, Client); \handle(buy, PurchaseList); \handle(buy, ClientAccount); \handle(store, ClientAccount); \handle(getHistory, Client); \handle(display, ClientAccount);VOID		

Table A.8 – 2de Table des scénarios pour l'exemple AL_{Spl} - PL

LVL	\context	\trace	
PL	\process{buy,store,display, getAccount,getHistory,chooseAd,sendAd}, \personalData{ClientAccount,Client}, \purposes{Administrative, Marketing}, \isGranted{(ClientAccount : Administrative), (Client : Administrative)} \hasPurposes{(buy :{Administrative}), (store :{Administrative}), (display :{Administrative}), (getAccount :{Administrative,Marketing}), (getHistory :{Administrative,Marketing}) (sendAd :{Marketing})}, \needData{(buy :{ClientAccount,Client}), (store :{ClientAccount}), (display :{ClientAccount}), (getAccount :{ClientAccount,Client}), (getHistory :{Client}),(sendAd :{Client})}	T8	\handle(getAccount, Client); \handle(getAccount, ClientAccount); \handle(getHistory, Client); \handle(getHistory, PurchaseList); \handle(display, ClientAccount); \handle(buy, Client); \handle(buy, PurchaseList); \handle(buy, ClientAccount); \handle(store, ClientAccount); \handle(getHistory, Client); \handle(display, ClientAccount); \handle(getAccount, Client); \handle(getAccount, ClientAccount); \handle(getHistory, Client); \handle(getHistory, PurchaseList); \handle(display, ClientAccount); VOID
		T9	\handle(getAccount, Client); \handle(getAccount, ClientAccount); \handle(getHistory, Client); \handle(getHistory, PurchaseList); \handle(display, ClientAccount); \handle(buy, Client); \handle(buy, PurchaseList); \handle(buy, ClientAccount); \handle(store, ClientAccount); \handle(getHistory, Client); \handle(display, ClientAccount); \handle(getAccount, Client); \handle(getAccount, ClientAccount); \handle(getHistory, Client); \handle(getHistory, PurchaseList); \handle(display, ClientAccount); \handle(getAccount, Client); \handle(getAccount, ClientAccount); \handle(getHistory, Client); \handle(getHistory, PurchaseList); \handle(chooseAd, PurchaseList); \handle(chooseAd, Ad); \handle(sendAd, Ad); \handle(sendAd, Client); VOID
Total	9		