Contribution à la spécification et à l'élaboration d'une plateforme de maintenance orientée connaissances.

Mohamed Hedi Karray

Année : **2012**

THESE

Présentée à

L'UFR des Sciences et Techniques de l'Université de Franche-Comté

Pour obtenir le

GRADE DE DOCTEUR DE L'UNIVERSITE DE FRANCHE-COMTE

en Automatique et Informatique

(Ecole Doctorale Sciences Physiques pour l'Ingénieur et Microtechniques)

Contribution à la spécification et à l'élaboration d'une plateforme de maintenance orientée connaissances

Par

Mohamed Hedi Karray

Soutenue le 09 Mars 2012 devant la Commission d'examen :

Rapporteurs :	Abdelhamid Mellouk	Professeur, Université Paris-Est Créteil Val-de-Marne
	Christophe Roche	Professeur, Université de Savoie
Examinateurs :	Philippe Charbonnaud	Professeur, ENIT de Tarbes
	Farhat Fnaiech	Professeur, Université de Tunis
Directeurs de thèse :	Brigitte Chebel-Morello	Maître de conférences, Université de Franche-Comté
	Noureddine Zerhouni	Professeur, ENSMM de Besançon

A mes chers parents avec tout mon amour et pour tout ce qu'ils m'ont appris.
A ma soeur que j'adore.

Des pensées,

« Et au-dessus de tout homme détenant la science il y a un savant [plus docte que lui] »

Saint Quran [12.76].

« On ne rendra pas la vie supportable par des raisonnements scientifiques ou de bons sentiments mais par des interprétations cohérentes qui peuvent exiger une part de sacrifice »

Jean Zin.

« Le perfectionnisme, en psychologie, consiste à se comporter comme si la perfection pouvait et devait être atteinte. Sous sa forme extrême, pathologique, le perfectionnisme considère toute imperfection comme inacceptable. Mais l'imperfection est un phénomène absolument général. La seule différence entre les êtres est une simple question de degré dans l'imperfection. »

Stanley Cavell.

« Tout ce que les autres ont rêvé d'autres le réaliseront »

Jules Verne.

Remerciements

Je suis très heureux d'écrire ces remerciements réservés de coutume à l'aboutissement de grands projets. Le mien fut ma thèse de doctorat.

Tout d'abord, je tiens à remercier et exprimer toute ma reconnaissance auprès de mon encadrante Brigitte CHEBEL-MORELLO. Elle m'a initié à la recherche dans un domaine qui m'a toujours motivé. Méticuleuse et perfectionniste, toujours disponible, profiler psychologique, elle m'a prodigué des conseils inestimables, dans tous les domaines, tout au long de ma thèse. Je suis très fière de la formation de chercheur acquise sous son encadrement.

Je remercie mon directeur de thèse, Noureddine ZERHOUNI, qui m'a parrainé au commencement de cette thèse. Ses commentaires sur mon travail m'ont chaque fois permis de m'orienter dans les bonnes directions. Je tiens à lui exprimer toute ma gratitude.

Je remercie mon rapporteur Christophe ROCHE pour l'honneur qu'il m'a accordé en acceptant d'évaluer ma thèse. Je suis très heureux que mon travail ait trouvé grâce à ses yeux. Je le remercie également pour les commentaires très constructifs qu'il a formulés.

Je remercie mon rapporteur Abdelhamid MELLOUK pour l'honneur qu'il m'a accordé en acceptant également d'évaluer mon travail, et aussi pour m'avoir donné plus de recul sur certains aspects que j'ai abordés.

J'adresse mes remerciements à mon examinateur Farhat FNAICH pour l'enthousiasme qu'il a manifesté à l'égard de ma thèse, ainsi que pour les commentaires détaillés dont il m'a fait part à propos de mon manuscrit.

Je souhaite remercier et rendre hommage à mon examinateur Philippe CHARBONNAUD pour avoir accepté de manière si chaleureuse de présider mon jury de thèse.

Je remercie très sincèrement, tout les membres de l'équipe PROMI, au sein de laquelle cette thèse est née et a évoluée, m'a procuré les conditions de travail les plus favorables.

Je tiens à remercier aussi tout les membres du département AS2M pour l'ambiance particulièrement agréable qui y règne.

Finalement, je remercie infiniment tous mes amis sans exceptions, tout particulièrement *Dr. Ali SASSANE, Dr. Mohamed KHARBOUTLY, Raed WAFI, Mariem HANNECHI, Jamel et Asma BOUBAKER, Hichem CHALABI, Lisa SERIR*, pour leur soutien, leur présence et leur écoute.

Le sommaire

Liste des figures

Liste des tableaux

Introduction générale

1. Problématique

Le rôle de la fonction de maintenance tient une position stratégique dans le milieu industriel. En effet, au cours des vingt dernières années, le rôle de la maintenance dans les entreprises est devenu de plus en plus important aussi bien sur le plan technologique que sur le plan économique. La maintenance est une fonction importante dans l'entreprise qui s'inscrit dans une logique de développement durable en permettant d'augmenter la disponibilité des systèmes industriels et d'allonger leur cycle de vie. La maintenance peut être définie comme l'ensemble des actions permettant de maintenir ou de rétablir un bien (*Asset*) dans un état spécifié ou en mesure d'assurer un service déterminé.

Par ailleurs, selon les chiffres de l'association française des ingénieurs et responsables de maintenance (AFIM), en France les dépenses annuelles dans l'industrie sont de l'ordre de 21,1 milliards d'euros, équivalent à 2,3 % de la production en valeur, dont 12 milliards d'euros de dépenses en produits et composants industriels en maintenance et travaux neufs. Cette activité génère 450 000 emplois de qualifications élevées dont 12000 cadres, ainsi que 85000 diplômés du CAP au Mastère (Afim, 2011). En Europe, le montant dépensé sur le budget de maintenance est d'environ 1500 milliards d'euros par an (Altmannshoffer, 2006). Les pannes et les temps d'arrêt entraînent une perte de qualité du produit ainsi qu'une perte au niveau de la production. Pallier ces pertes et anticiper les pannes permet d'assurer une continuité des services fournis par l'équipement et par là-même, une augmentation de la qualité de ceux-ci.

Ainsi, tout en ayant cet intérêt dans l'entreprise, l'industrie a vu l'évolution de différentes générations de systèmes informatiques de maintenance. Les nouvelles technologies de l'information et technologies de la communication (TIC) ont contribué à l'élaboration et l'évolution progressive de ces systèmes allant d'un système manuel vers des systèmes de Gestion de Maintenance Assistée par Ordinateur (GMAO), des progiciels de gestion intégré (PGI/ ERP en anglais) et des systèmes intégrant divers types d'application comme les plateformes de e-maintenance.

Ces différents systèmes basés sur l'intégration et la coopération fournissent à ses utilisateurs (les différents acteurs de la maintenance) un ensemble de services permettant une gestion informatisée d'un ensemble d'activités de base appartenant au processus de maintenance (exemple l'intervention, la planification, le diagnostic, etc.). Ils fournissent ainsi un support d'aide à la décision par un ensemble d'indicateurs prédéfinis sur les coûts, les interventions et les équipements.

Néanmoins, les besoins des acteurs de maintenance utilisateurs de ces systèmes évoluent avec le temps et ne peuvent être satisfaits par les services actuellement fournis par les systèmes informatiques d'aide à la maintenance du marché. En effet ces services prennent appui sur la connaissance initialement formalisée mais qui n'est pas systématiquement actualisée, et les services proposés au bout de quelques années ne sont plus en

concordance avec la connaissance du moment. On doit tenir compte de l'aspect dynamique de la connaissance, pour répondre aux besoins des utilisateurs et améliorer les performances des logiciels d'aide proposant ces services. Cette thèse tente de donner un élément de réponse à ce problème de prise en compte de l'évolution de la connaissance qui aura un impact sur les services proposés aux utilisateurs dans un système de gestion de maintenance.

2. Cadre et objectifs de travail

Ce travail rentre dans le cadre du projet transfrontalier SMAC soutenu par le programme Interreg IV France-Suisse. SMAC est un projet entre l'Université de Franche-Comté et l'Ecole Polytechnique Fédérale de Lausanne et de deux PME l'une basée en Jura Suisse TORNOS, l'autre à Besançon EM@SYSTEC.

Ce projet a pour objectif de concevoir et de développer une solution de maintenance intelligente

- assurant le suivi d'un équipement industriel tout au long de son cycle de vie d'allonger sa durée de vie utile par le maintien en bon état de ses composants.
- de concevoir et développer une solution permettant d'une part, d'assurer la traçabilité des informations relatives à l'équipement durant son cycle de vie et d'autre part, d'élaborer et capitaliser la connaissance sur cet équipement.
- de proposer une architecture de s-maintenance (s comme sémantique), basée sur la définition d'un vocabulaire commun (ontologie du domaine de maintenance), et permettant d'assurer un dialogue compréhensible homme-machine et/ou machine-machine

Dans ce contexte, l'objectif de cette thèse est de fournir un système de maintenance assurant le suivi d'un équipement et son maintien en bon état, répondant aux besoins des utilisateurs de maintenance. Ce système sera concrétisé par une plateforme informatique intelligente assurant le retour d'expérience attendue par les utilisateurs, grâce à une démarche de capitalisation des connaissances. Dans cette mouvance, nous proposons d'adopter une approche basée sur l'ingénierie des connaissances pour l'élaboration d'un système de maintenance fournissant des éléments de réponse aux limites des systèmes existants. Ceci permettra d'en tirer profit en assurant une valeur ajoutée par une bonne exploitation des connaissances par des raisonnements logique permettant de faire évoluer le capital de connaissance du système de maintenance et ses applications intégrées. Notre objectif se déclinera en deux grandes étapes :

- La première concerne la spécification et l'élaboration d'une plateforme orientée connaissances permettant de garantir le partage et l'exploitation des connaissances expertes du domaine au profit de la collaboration interopérable sémantiquement et tenant compte de la dynamicité des connaissances et des services fournis aux utilisateurs.
- La deuxième a trait à l'élaboration d'une ontologie du domaine de maintenance couvrant tous les aspects de celui-ci et permettant d'alimenter la base de connaissances sur laquelle prend appui la plateforme pour assurer ses fonctionnalités et services.

3. Contribution et démarche de travail

Afin d'élaborer une plateforme de maintenance orientée connaissance répondant aux besoins des utilisateurs, nous suivons une démarche qui consiste :

- à répondre aux nouveaux besoins des utilisateurs et aux enjeux des nouveaux systèmes de maintenance.
- à passer en revues les générations des systèmes de maintenance afin d'identifier les causes d'inadéquation entre services fournis et besoins des utilisateurs.
- à se positionner par rapport à la dernière génération de ces systèmes, la e-maintenance, dans l'objectif si possible de l'adapter aux nouveaux besoins ou de créer une nouvelle génération de systèmes.
- de proposer une nouvelle génération de systèmes de maintenance (système de s-maintenance) répondant à des concepts se différenciant de la e-maintenance.
- à proposer une architecture de plateforme assurant les caractéristiques des systèmes de la nouvelle génération en se basant sur des composants logiciels intelligents et un système à base de connaissances.
- à développer une ontologie du domaine de maintenance à l'aide des standards du domaine comme MIMOSA-CRIS et les ontologies issues de différents projets reliés à la maintenance comme les projets PROTEUS, PROMISE.
- à développer les fonctionnalités permettant à cette plateforme d'assurer la dynamicité de ses connaissances et de ses services grâce à l'exploitation de l'ingénierie des traces d'interactions.

4. Organisation du mémoire

Cette démarche est mise en place et détaillée dans les différents chapitres de ce mémoire que nous organisons comme suit :

Chapitre 1 : Après avoir rappelé les notions de base d'un processus de maintenance, nous avons étudié les nouveaux besoins et enjeux des systèmes intelligents de maintenance en nous basant sur les recommandations des projets avant-gardistes. A la lumière des enjeux identifiés, nous classons les différentes générations de systèmes de maintenance, et nous définissons un système de cinquième génération répondant aux enjeux identifiés que nous appelons s-maintenance. Afin de mieux positionner cette nouvelle génération, nous étudions les définitions du concept et des plateformes de e-maintenance et nous proposons une définition en tenant compte des différents travaux dans le domaine, du concept de e-maintenance et de s-maintenance prenant appui sur les besoins non couverts et une réponse possible pour les satisfaire.

Chapitre 2 : Afin de mettre en place une plateforme de s-maintenance, nous nous sommes intéressés aux plateformes existantes classées sous la dénomination e-maintenance, mais pouvant dépasser ce concept et répondre au concept de s-maintenance.

Nous avons analysé les différentes plateformes de maintenance suivant quatre critères principaux dans la s-maintenance à savoir : 1) le type de maintenance traité, 2) l'interopérabilité entre les applications de la plateforme, 3) l'exploitation de l'ingénierie et le management des connaissances au cœur de la plateforme et 4) le degré d'intégration (coopération et collaboration).

Chapitre 3 : le troisième chapitre portera sur la spécification et l'élaboration d'une architecture d'une plateforme répondant aux caractéristiques de la s-maintenance et assurant ses fonctionnalités. Parmi les différentes méthodes d'ingénierie logicielle nous choisissons l'architecture à base de composants (appelé en anglais Components Based System) pour mettre en place cette architecture. Dans cette architecture chaque composant répond à un ou plusieurs de ces exigences comme l'interopérabilité sémantique, l'inférence de connaissances, la génération de services à la demande, l'auto-apprentissage, l'autogestion, etc. Nous présentons donc les composants, leurs processus de fonctionnements, et l'architecture interne de quelque uns.

Chapitre 4 : L'architecture de la plateforme de s-maintenance orientée connaissance prend appui sur un système à base de connaissance et notamment une ontologie du domaine de maintenance. Nous présentons dans ce chapitre le processus de construction d'une ontologie du domaine de maintenance que nous appelons IMAMO (Industrial MAintenance Management Ontolgie) en prenant appui sur la méthodologie METHONTOLOGY et le langage de description PowerLoom. Ainsi, nous développons l'activité d'évaluation par l'application de différentes approches permettant de mettre en évidence la valeur ajoutée de IMAMO.

Chapitre 5 : le cinquième chapitre se focalise sur le développement et la validation des fonctionnalités assurées par la plateforme de s-maintenance comme l'auto-traçabilité, l'auto-apprentissage et l'autogestion. A ces fins, nous proposons un système à base de traces (SBT) exploitant l'ontologie IMAMO comme modèle associé aux traces numériques d'interaction entre utilisateurs et plateforme, et des modules d'apprentissage.

Nous expliquons donc dans ce chapitre la mise en place et le fonctionnement des processus de ce SBT. D'autre part, nous évaluons l'impact de ces fonctionnalités assurant la dynamicité de la connaissance dans la plateforme sur les performances de maintenance en coûts et en temps grâce à une simulation de la gestion de maintenance sur un cas d'utilisation.

Chapitre 1 Evolution des systèmes de maintenance et enjeux des systèmes du futur

1. Introduction

Au cours des vingt dernières années, le rôle de la maintenance dans les entreprises est devenu de plus en plus important aussi bien sur le plan technologique que sur le plan économique. Elle s'inscrit dans une logique de développement durable et permet d'augmenter la disponibilité des systèmes industriels d'une part et d'allonger leur cycle de vie d'autre part. De plus, l'anticipation des défaillances sur les éléments critiques de systèmes, grâce à la maintenance prévisionnelle (le pronostic) (Muller A. , 2005) permet de prévenir les risques industriels (nucléaire, plateforme pétrolière…) et d'assurer la sécurité des personnes et des biens. Enfin, la mise en œuvre de la maintenance nécessite une qualification et participe à la valorisation du personnel technique de maintenance. Ce processus de maintenance sera plus détaillé dans la section 2.

Par ailleurs, le développement des systèmes informatiques d'aide à la gestion de la maintenance industrielle a commencé lorsque la maintenance a été reconnue comme fonction fondamentale dans l'entreprise et un accent particulier a été mis sur le développement des procédures de cette fonction. L'industrie a vu l'évolution de différentes générations de systèmes de maintenance passant d'un système de maintenance local au système de GMAO (Gestion de la Maintenance Assisté par Ordinateur) et arrivant à une plateforme de e-maintenance.

En effet, la tendance actuelle des entreprises à se recentrer sur leur corps de métier en externalisant tout ou en partie, la fonction de maintenance, combinée aux avancées dans le domaine technique et aux nouvelles technologies de l'information et de la communication ont induit une évolution des systèmes de maintenance vers des systèmes intégrateurs de modules intelligents, communiquant et collaborant entre eux.

Ces différents systèmes fournissent à ses utilisateurs (les différents acteurs de la maintenance) un ensemble de service permettant une gestion informatisée d'un ensemble d'activités de base appartenant au processus de maintenance (exemple l'intervention, la planification, le diagnostic, etc.). Ils fournissent ainsi un support d'aide à la décision par un ensemble d'indicateurs prédéfinis sur les coûts, les interventions et les équipements.

Les besoins des utilisateurs évoluent dans le temps en fonction de nouvelles contraintes ou de l'expertise de ceux-ci, et les services fournis par ces systèmes d'aide ne sont plus en concordance et nécessite une réactualisation.

Quoique, les services et les indicateurs fournis par de tels systèmes répondent à un moment donné aux attentes et besoins de l'utilisateur, ces services peuvent être insuffisants à un autre moment ou d'autres besoins.

Soit l'exemple d'un expert de maintenance utilisant un système de GMAO qui fourni le MTBF[1] et MTTR[2] comme indicateurs. Le directeur de l'entreprise étudie une possibilité d'investir dans des nouveaux équipements et a besoin d'obtenir des informations sur le rendement et l'efficacité des équipements possédés. Pour répondre à cette requête, l'expert a besoin de l'indicateur TRG[3] (Taux de rendement global), mais le système de GMAO qu'il possède ne fournit pas cet indicateur. Dans ce cas, il doit fouiller la base de données manuellement,

[1] *Mean Time Between Failures* (*MTBF*): Le temps moyen entre pannes
[2] Mean Time To Repair *(*MTTR*): temps moyen pour réparer,*
[3] *TRG* : indicateur sur les performances des machines qui utilise le temps des arrêts des machines comme paramètres. Ce facteur est le rapport entre le temps de fonctionnement de la machine et la durée de vie de la machine.

chercher les facteurs de temps de l'historique s'il peut les localiser dans la base de données et faire les calculs à la main en se référant à tous les types d'arrêts.

La réactualisation doit se faire manuellement, ou être intégrée dans le service fournissant les indicateurs, soit en faisant redévelopper ce service par un informaticien qui intégrera ce nouvel indicateur, soit en achetant une nouvelle version de GMAO si elle existe, ou dans le cas d'une plateforme de e-maintenance intégratrice de services, intégrer un service au calcul de cet indicateur. Les services et les indicateurs fournis par ces systèmes ne sont pas dynamiques et n'évoluent pas au fur et à mesure de manière automatique avec les besoins des utilisateurs. L'idéal serait d'avoir un service à la demande de calcul d'indicateurs qui suivrait l'évolution des besoins.

Dans un autre registre, quand l'utilisateur a acquis une nouvelle expertise sur l'équipement grâce aux nouvelles réparations faites, il est frustrant de ne pas pouvoir réutiliser cette expertise dans le système d'aide, qui ne s'est pas remis à jour, faute de réactualisation. En effet, l'utilisateur une fois la réparation effectuée omet d'intégrer ces informations dans le système d'aide, quand celui-ci le lui permet. Les autres utilisateurs ne peuvent donc pas bénéficier de cette expérience. La capitalisation des connaissances n'est pas effective, et devrait s'effectuer automatiquement en minimisant l'impact utilisateurs. Il serait intéressant d'avoir un service de retour d'expérience apprenant par lui-même, d'avoir des traces de toute opération ou événement se déroulant dans le système d'aide et les rendre disponibles à l'utilisateur d'une manière ergonomique, et les exploiter pour développer des services se remettant à jour par eux même.

Sur la base de ces deux exemples simples, on peut constater que ces systèmes actuels d'aide à la maintenance ne satisfont pas toujours les attentes des acteurs de maintenance. Insatisfaction due à l'absence d'évolution automatique de l'intelligence des services qui ne s'adaptent pas aux besoins des utilisateurs.

Les limites des systèmes de maintenance, par rapport aux besoins des utilisateurs nous amènent à étudier les différents aspects que doit présenter la future maintenance. La section 3 prend appui sur les travaux de IMS-Center sur la future maintenance et le projet avant-gardiste TATEM pour définir les caractéristiques des ces systèmes de maintenance.

Par rapport à ces enjeux, la section 4 évaluera les générations de systèmes de maintenance selon leurs caractéristiques pour identifier à quels besoins ils répondent, leurs limites et pourquoi ils ne répondent pas aux attentes évolutives des utilisateurs.

On fera apparaître une cinquième génération initialement proposée par Rasovska (Rasovska I. , 2006) et devant résoudre le problème d'interopérabilité sémantique d'une plateforme de e-maintenance.

Nous étudions une nouvelle génération de système faisant l'objet de travaux de recherche actuels dans le domaine ayant comme objectif de répondre aux enjeux identifiés. Nous consacrerons donc la suite de ce chapitre à une proposition d'évolution entre le concept de e-maintenance (présentant la dernière génération de système actuel) vers le concept de s-maintenance répondant aux enjeux de demain et formant la nouvelle génération. Afin de mieux expliquer ce passage entre générations, la section 5 sera consacrée à étudier le concept de e-

maintenance, et à proposer une définition fédérant les travaux dans le domaine. La section 6 sera consacrée à la définition du concept de s-maintenance basée sur un système d'information orienté connaissance.

2. Processus de maintenance

Les normes NF X 60-010 et 60 011 définissent la maintenance comme l'ensemble des actions permettant de maintenir ou de rétablir un bien (*Asset*) dans un état spécifié ou en mesure d'assurer un service déterminé. Dans cette définition, nous trouvons deux mots-clés à savoir maintenir et rétablir, qui font référence aux différents types de maintenance dont nous rappelons les définitions dans l'*annexe 1 : types de maintenance.* Le premier mot clé a trait à une action préventive, que la prévention soit systématique conditionnelle ou prévisionnelle tandis que le deuxième est lié à l'aspect correctif.

Une autre définition en concordance avec la première définition est fournie par le ministère du Commerce et de l'Industrie (DTI) Britannique, présentant la maintenance comme « la gestion, le contrôle, l'exécution et la qualité des activités qui feront en sorte que les niveaux optimaux de la disponibilité et la performance globale d'un bien sont atteints, afin de répondre aux objectifs de l'entreprise » (Labib, 2006).

Ainsi, la norme AFNOR définie la maintenance comme «l'ensemble de toutes les actions techniques, administratives et de gestion durant le cycle de vie d'un bien, destinées à le maintenir ou à le rétablir dans un état dans lequel il peut accomplir la fonction requise ».

La Figure 1-1 issue de Retour et al. présente la fonction maintenance comme un ensemble d'activités regroupées en deux sous-ensembles : les activités à dominante technique et les activités à dominante gestion (Retour, Bouche, & Plauchu, 1990).

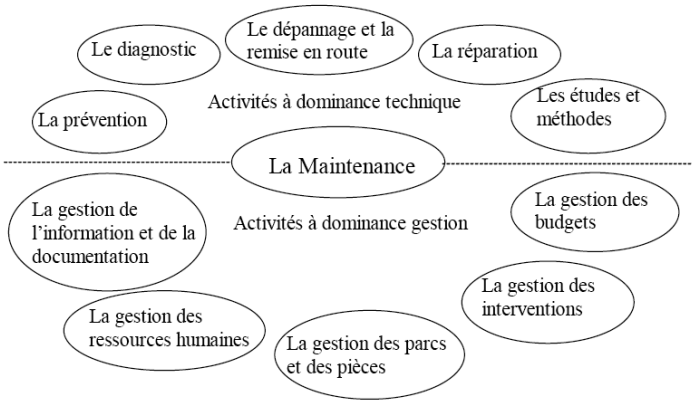


Figure 1-1 La fonction maintenance (Retour, Bouche, & Plauchu, 1990)

Afin de proposer un système de gestion et d'aide à la maintenance, Rasovska et al (Rasovska, Morello-Chebel, & Zerhouni, 2007) partant de ces définitions et du processus de maintenance[4] ont identifiés quatre principaux domaines, deux techniques et deux de gestion dont les activités s'emboitent les unes dans les autres, et qui catégorisent toutes les tâches du processus global de maintenance (voir Figure 1-2).

[4] Un processus de maintenance selon (Spadoni, 2004) est un enchaînement d'activités contrôlées ou interactives.

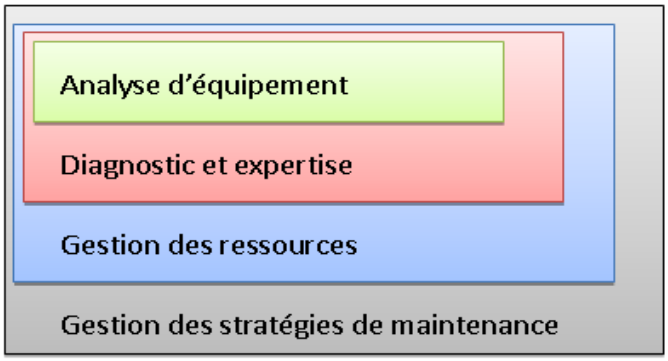


Figure 1-2 Les différentes couches du processus de maintenance

Les deux premières couches dédiées au domaine technique, comportent :

- L'analyse de l'équipement, domaine qui englobe le recensement des données et information ainsi que les différentes analyses de l'équipement (structurelle, fonctionnelle/ dysfonctionnelle, événementielle etc..), les connaissances concernant le plan de l'expertise associé.
- Les applications de surveillance, de détection de défaillance, de diagnostic de pannes, de pronostic des dégradations et d'aide à la décision concernant des mesures de réparation, et de maintien en conditions opérationnelles.

Les deux couches suivantes traitent de la gestion des ressources et des stratégies de maintenance (contrats). Ces deux couches utilisent des indicateurs techniques et financiers. Ces indicateurs proviennent des rapports d'intervention édités dans les phases de diagnostic de panne et de réparation, mais gérés dans la phase de gestion des interventions, phase faisant partie de la gestion des ressources. La première couche permet de gérer les ressources humaines et matérielles, les plans des interventions de maintenance préventive et le stockage avec des pièces de rechange, des outils utilisés pour les interventions.

La couche de gestion des stratégies de maintenance et de contrat gère les décisions concernant la stratégie générale de maintenance appliquée dans un parc de machines, et la gestion des contrats.

Maints travaux développent un aspect ou quelques aspects concernant la maintenance, soit les deux premières couches dans les travaux sur le CBM (Tsang A. H., 1995) soit la troisième couche sur la gestion des ressources humaines (Shenoy & Bhadury, 1998). Nous considérons comme Karim et al (Karim, Kajko-Mattsson, & Söderholm, 2008) que le processus de maintenance doit être pris en considération dans son ensemble, y compris dans ses phases de gestion, de planification, de préparation, d'exécution, sans oublier l'évaluation et l'amélioration pour fournir aux opérateurs un système complet d'aide à la maintenance.

3. Les nouveaux besoins et enjeux des systèmes intelligents de maintenance

3.1- Les nouveaux besoins de maintenance

La maintenance de nos jours combinée avec les pertes en coûts et en temps imposent une plus grande efficacité au processus de maintenance dans les entreprises. Par conséquent, les nouveaux systèmes de maintenance doivent prendre ceci en considération et fournir une valeur ajoutée dans le but d'améliorer cette efficacité et de faire évoluer le rôle de la maintenance. A ces fins, le besoin de mettre en place de plus en plus de services intelligents s'est imposé pour répondre à ces exigences ; Services d'aides à la décision pour la réutilisation des composants dans le but de diminuer les coûts, Services de retour d'expérience pour améliorer les activités d'expertises, de diagnostic, d'intervention, ainsi que d'autres services intelligents.

Ainsi, Lee identifie les services nécessaires pour demain comme (Lee, Liao, Lapira, Ni, & Li, 2009):

- un système orienté client (Customer-Intensive System) dans lequel les utilisateurs bénéficiaires des résultats du système participent au processus d'élaboration du système en fournissant les entrées nécessaire pour cette élaboration. (Pinhanez, 2008).
- un système instrumenté (Smart Agent),
- un système d'analyse intelligente,
- un système de gestion des connaissances métier,
- un système orienté clients pour éviter d'éventuels problèmes.

Ainsi, ce que nous constatons en étudiant les travaux actuels en maintenance qu'on a besoin de plus de fonctionnalités à valeur ajoutée, comme les services de prédiction temporelle telles que des fonctions d'anticipation de dégradation, des services à la demande édités par l'utilisateur exprimant ses besoins (les services d'aide à la décision) et ceux basés sur l'utilisation (c.à.d. le retour d'expérience).

Le centre IMS a fourni un schéma des besoins communs non satisfaits pour les futurs services de maintenance [*"Common Unmet Needs in Future Maintenance and Service"*] (voir Figure 1-3).

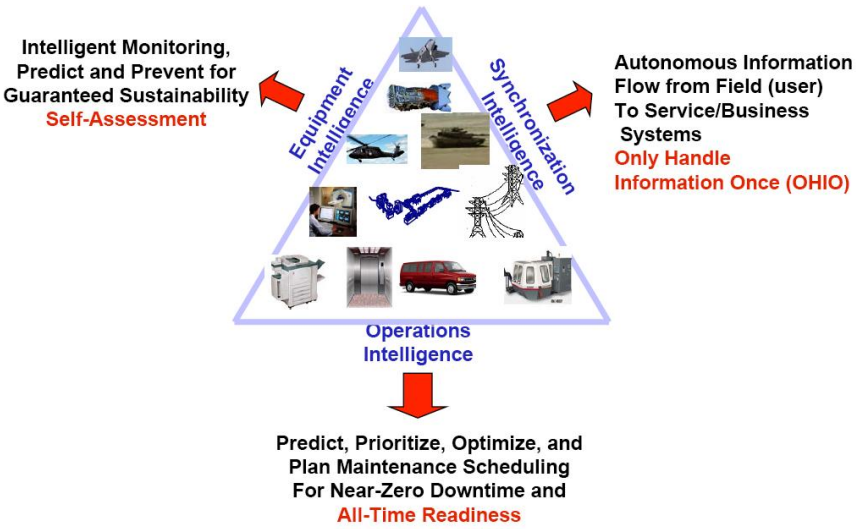


Figure 1-3 CuNeMas : Common Unmet Needs in Future Maintenance and Service (Lee, Liao, Lapira, Ni, & Li, 2009)

La Figure 1.3 présente 3 axes sur l'intelligence (mot clé des services de demain) qui se recoupent en triangle ; intelligence appliquée à l'équipement, aux opérations de maintenance et à l'information et sa synchronisation.

Un des projets avant-gardistes de maintenance, le projet international TATEM (Technologies and Techniques for New Maintenance Concepts) définit l'architecture d'une approche de gestion de maintenance intégrée pour la santé des avions. Son objectif est d'élaborer et de valider des philosophies, des technologies et des techniques, qui peuvent être utilisées pour transférer des activités de maintenance imprévues à des activités de maintenance prévisibles.

TATEM a une vision holistique[5] de la maintenance des avions en étudiant tous les aspects des avions et les problèmes de maintenance. Le projet identifie les différents aspects philosophiques, technologiques et techniques de la future maintenance suivants (GE-Aviation, 2007) :

• Les techniques de traitement du signal (basées sur des modèles de raisonnement, de la logique floue, de réseaux neuronaux,), qui peuvent être utilisées pour convertir les données en informations sur la santé des systèmes.

• Les méthodes de diagnostic et de pronostic pour identifier, localiser et anticiper les pannes et les dysfonctionnements et ainsi de réduire l'incidence des fausses alarmes.

• Les nouvelles technologies des capteurs à bord pour recueillir des données de l'avion et ainsi alimenter les systèmes de pronostic ou de diagnostic.

• Les technologies d'interface homme-machine afin de fournir aux acteurs de maintenance les informations, les données et les conseils sur le point de travail.

• Les processus de maintenance qui ne nécessitent aucun travail de maintenance programmée.

• Les techniques d'aide à la décision pour générer des processus orientés information et guider les acteurs de maintenance.

Finalement, nous constatons qu'une forte concordance entre les besoins identifiés par TATEM et les services recommandés par Lee corrobore les enjeux mis en place par Jay Lee. L'objectif commun qui unit ces propositions relatives aux systèmes de maintenance est une efficacité plus élevé du processus de maintenance à travers une meilleure exploitation des connaissances métier et des technologies intelligentes, ce qui induit une réduction des pertes en coûts et en temps.

3.2- La gestion des connaissances

Lee définit la maintenance du future comme: « La bonne information aux bonnes personnes pour faire les bonnes choses au bon moment », définition correspondant au problème de capitalisation des connaissances (Matta, Ermine, Aubertin, & Trivin, 2001). En effet de nombreux travaux sur la connaissance sont développés dans un grand nombre de recherche à savoir les systèmes d'information, le Web Intelligent, les méthodes de retour d'expérience, etc. (Rezgui, Boddy, & Wetherill, 2009) (Mrissa, Benslimane, Ghedira, & Maamar, 2005). La maintenance est un de ces domaines où les connaissances sont développées, que ce soit par une connaissance

[5] Globaliste : qui relève de l'holisme, considérant l'objet comme constituant d'un tout.

formelle ou sous forme de règles de décision extraites à partir des données. On peut identifier les travaux sur la connaissance dans les besoins CuNeMas au niveau de l'axe synchronisation de l'information intelligente.

Concernant la connaissance, l'entreprise a pris conscience de l'importance de son capital intangible à savoir l'activité intellectuelle, la connaissance le savoir en son sein. Stuart en 1996 a dit que notre société transite de l'économie industrielle à l'économie fondée sur une activité de connaissance (Stuart, 1996). La gestion des connaissances devient donc un enjeu stratégique lié à la croissance économique et le moteur de la productivité. Elle répond au nouveau défi de l'environnement concurrentiel : la gestion des connaissances tente de lier la vision classique de production à ces nouveaux besoins.

La gestion des connaissances (KM) a été définie par Grundstein (Grundstein, 2002) comme la gestion du processus de création, de partage, de stockage, de réutilisation et de capitalisation de connaissances, ainsi que ses activités connexes. Aujourd'hui, la connaissance est de plus en plus considérée comme l'atout le plus important des organisations et des entreprises, en particulier dans le secteur des services des industries basé sur la connaissance. Ces connaissances, expériences et savoir-faire des entreprises sont stockées et capitalisées pour tirer profit afin de créer un capital intellectuel défini comme «la somme accumulée ainsi que la valeur des connaissances et de l'expertise partageables de l'entreprise» (Moradi & Vallespir, 2010). Selon la définition ci-dessus, pour que ces connaissances deviennent un capital intellectuel, elles doivent être partagées.

Dans le cadre de la maintenance, Rasovska et al (Rasovska, Morello-Chebel, & Zerhouni, 2007) (Rasovska, Chebel–Morello, & Zerhouni, A mix method of knowledge capitalization in maintenance, 2008) ont construit une mémoire d'entreprise de maintenance grâce au processus de capitalisation des connaissances. Par «mémoire d'entreprise», on entend «un ensemble structuré de connaissances liées à l'expérience de l'entreprise dans un domaine donné». La connaissance doit être identifiée, formalisée et modélisée pour être récupérée, réutilisée et mise à jour par les employés de l'entreprise (Kobbacy & Jeon, 2001).

Dans le même champ d'application, les travaux liés à la maintenance présentés par Labib sur l'auto-maintenance (Labib, 2006) sont basés sur des méthodes d'extraction et de gestion des connaissances en utilisant des systèmes intelligents à base de règles; des systèmes de raisonnement à partir de cas et / ou d'intelligence artificielle à base d'inférence neuro-flou.

3.3- Les fonctionnalités autonomes (self-X)

Ces dernières années, un grand nombre de recherches a été effectué en essayant d'explorer de nouvelles méthodes concernant la gestion des systèmes logiciels complexes (Weiss, Zeller, & Eilers, 2011). Ces recherches ont clairement souligné que la durabilité des systèmes nécessite des niveaux élevés d'autonomie et d'adaptabilité des systèmes (Onori, Semere, & Lindberg, 2011).

En 2001, IBM a introduit le paradigme de « Autonomic Computing (AC) » (Horn, 2001). L'idée principale de ce paradigme est l'adaptation du comportement d'un système complexe qui interagit de manière autonome. La gestion des éléments autonomes est réalisée par un cycle de reconfiguration où chaque élément autonome surveille et analyse l'environnement, les plans de ses prochaines étapes et exécute les actions qui en résulte.

Ainsi, le principe de ce type de système autonome évolutif ne réside pas seulement dans la capacité des composantes du système à s'adapter à l'évolution des conditions de fonctionnement mais aussi à aider à l'évolution de ces composants dans le temps. Ce qui permet ainsi aux processus de devenir auto-X, où le X indique un ou plusieurs fonctionnalités souhaitables d'un système soumis à un état de fonctionnement variables.

Ces fonctionnalités auto-x (comme auto-adaptation, auto-organisation, auto-configuration, d'auto-guérison, d'auto-apprentissage, autogestion, autoprotection et auto-optimisation (Kephart & Chess, 2003)) améliorent l'intelligence, l'évolutivité, la robustesse et la flexibilité du système. Typiquement, les systèmes évolutifs sont composés de modules intégrés intelligents assurant ces propriétés.

Comme le montre la Figure 1-4 que nous adaptons de (Garlan & Schmerl, 2002) et (Weiss, Zeller, & Eilers, 2011), un système assurant des fonctionnalités d'auto-X fonctionne suivant une boucle fermée basée sur son système de gestion de connaissances où chaque étape de cette boucle peut faire état d'une ou plusieurs des propriétés d'auto-X.

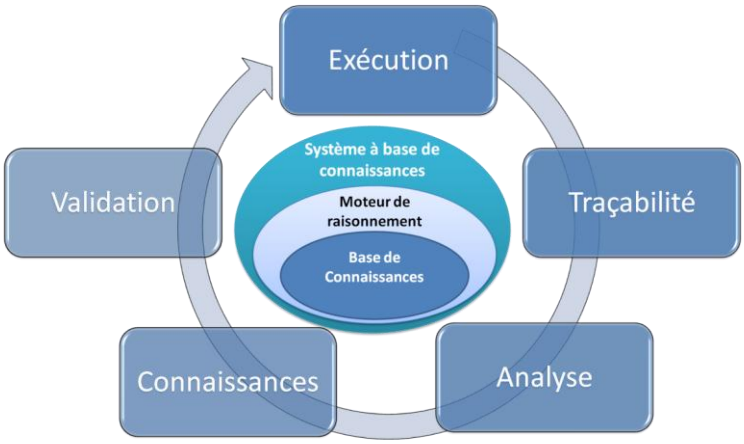


Figure 1-4 Boucle fermée des services auto-X

Nous présenterons dans la prochaine section une analyse des différents systèmes de maintenance d'hier à aujourd'hui afin de pouvoir dégager des spécificités susceptibles de répondre à ces nouveaux enjeux.

4. Les générations des systèmes de Maintenance

4.1- Classification des générations de systèmes

Nous consacrons ce paragraphe à étudier le passé, le présent et l'avenir des systèmes de maintenance. Prenant appui sur l'histoire du développement et de la prévision de la tendance de développement des technologies de maintenance (voir Figure 1-5).

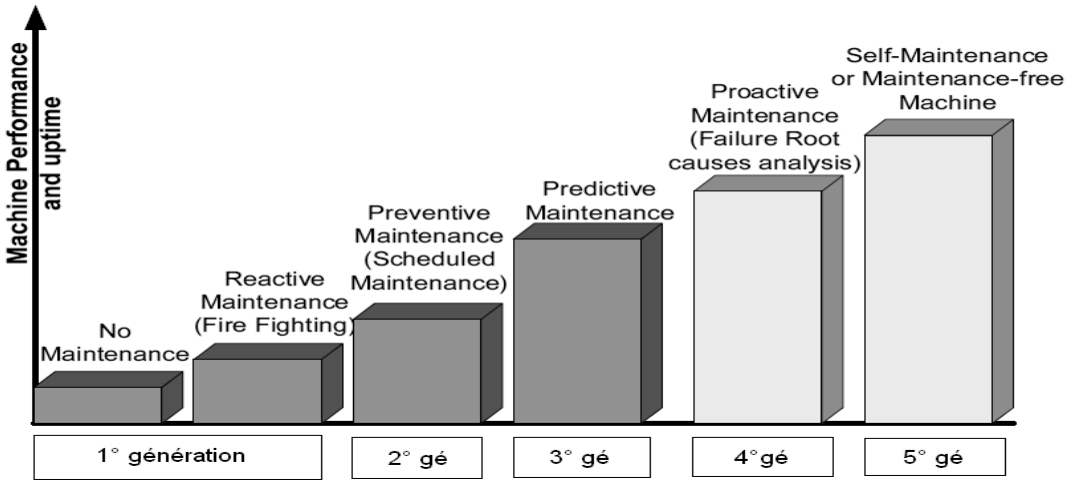


Figure 1-5 Le développement des technologies de maintenance (Lee & Wang, 2008)

Nous constatons qu'il y a une relation de correspondance entre les types de maintenance existants : la maintenance réactive, préventive systématique, prédictive, proactive et la maintenance de demain « l'auto-maintenance » et les systèmes informatiques de maintenance fournissant les services requis par chaque type de maintenance. Sur la base de cette correspondance nous classifions l'évolution des systèmes de maintenance.

Prenant appui sur l'étude faite par Rasovka sur l'évolution des systèmes (Rasovska, Morello-Chebel, & Zerhouni, 2007) et les travaux de Lee (Lee & Wang, 2008), nous définissons cinq générations de systèmes allant d'un système manuel vers les systèmes de Gestion de Maintenance Assistée par Ordinateur (GMAO), les Entreprise Ressource Planning (ERP) et les systèmes intégrant divers types d'application comme les plates-formes e-maintenance. La dernière génération de ces systèmes alliant les nouveaux besoins, et l'évolution de la technologie de 4° génération concerne les plates-formes de s-maintenance intégrant la connaissance dans le cœur de la plateforme, 5° génération qui fait l'objet de cette thèse.

Le tableau 1 caractérise ces générations selon sept critères ; le système informatique de maintenance existant réalisant le type de maintenance, le degré d'informatisation de la maintenance, et les 4 domaines d'application définis au paragraphe 2 à savoir les techniques d'analyse de l'équipement, les technologies de monitoring de l'équipement, les ressources gérer et finalement les stratégies et politiques de maintenance)

La première génération ne s'intéresse qu'à la réparation de l'équipement qui tombe en panne. Il n'y a pas de véritable politique de maintenance permettant d'anticiper toute défaillance.

La deuxième génération a vu l'informatisation des procédures de maintenance, en passant par le développement de fichiers informatiques recensant l'ensemble des équipements, des opérations de maintenance, des plans et des schémas, inventoriant les stocks d'outils et de pièces de rechange, etc. L'intégration de ces fichiers et l'automatisation des activités de maintenance deviennent donc possibles et peuvent être réalisées grâce au système de GMAO (Gestion de Maintenance Assistée par Ordinateur). Les événements de maintenance ont été recensés : la défaillance, la maintenance préventive et la gestion des stocks.

En effet une GMAO est un outil informatique permettant de gérer les ordres d'intervention et contrôler de manière efficace l'inventaire et les données correspondantes. Cet outil inclut les trois tâches de base de gestion de la maintenance, à savoir les tâches essentielles comme la planification et l'ordonnancement, le contrôle, l'exécution des interventions et le suivi, les tâches administratives telles que la gestion de projet et enfin la gestion des données concernant les plans de travail, les documents techniques et les matériaux, etc (Hoeck, H. et al. 2002).

La troisième génération représente une nouvelle étape dans la rationalisation des processus métiers et dans l'intégration de la maintenance avec d'autres fonctions de l'entreprise. Le logiciel de maintenance devait être connecté à d'autres progiciels de gestion intégrée de l'entreprise comme par exemple l'ERP (Enterprise Resource Planning).

De plus, l'émergence des techniques d'analyses modernes telles que l'analyse vibratoire, l'analyse de viscosité, thermographie inférée, couplées à des systèmes experts ont été décrits sous le sigle TTAO (Travaux Techniques Assistés par Ordinateur) ou TMAO (Techniques de Maintenance Assistées par Ordinateur). Ces systèmes d'analyse sont également destinés à fournir le support décisionnel dans le diagnostic, le pronostic et les opérateurs qui effectuent la réparation des équipements, etc. (Rasovska I. , 2006).

Dans la même veine, des systèmes d'acquisition et de contrôle (SCADA), ainsi que les systèmes de stockage de données (entrepôt de données data mining) et de contrôle ont été développés ainsi que des systèmes de gestion des données techniques et de documentation, etc.

Une série de projets a été lancé pour relier les applications dédiées à un type de maintenance par exemple (ICAS : Integrated Condition Assessment System) projet concernant les différentes architectures de maintenance conditionnelle (Muller, Marquez, & Iung, 2008).

Ainsi, dans la quatrième génération s'est imposée l'idée d'intégrer ces différentes briques intelligentes à travers un système d'information concrétisé par des plateformes de transferts et d'échange d'informations.

La Figure 1-6 présentée dans (Karim & al, 2008) illustre la nécessité d'intégrer ces systèmes d'aide à la maintenance.

La partie gauche de cette figure illustre la tâche difficile d'un technicien de maintenance qui doit être connecté à différentes applications pour ensuite accéder à chaque application, récupérer les informations pertinentes et les synthétiser afin de les adapter à l'utilisateur dans le but de lui faciliter la tâche pour faire face aux différentes nécessités de maintenance se présentant à lui et lui permettre d'utiliser ces informations dans une autre application.

Sur la partie droite de la figure, le technicien de maintenance ne doit être connecté qu'à la plateforme de e-maintenance et ensuite avoir une interface unique permettant l'accès et la manipulation de toutes les applications intégrées. Ainsi, l'intégration[6] et la collaboration[7] dans les systèmes sont considérées donc comme les principales

[6] L'intégration de systèmes est le processus qui crée les liens physiques et fonctionnelles entre différents systèmes informatiques et différents applications logicielles hétérogènes afin qu'ils agissent comme un seul système coordonné. (Ruh, Maginnis, & Brown, 2001)

méthodes permettant de conduire le système de maintenance vers l'amélioration de la productivité et de l'efficacité.

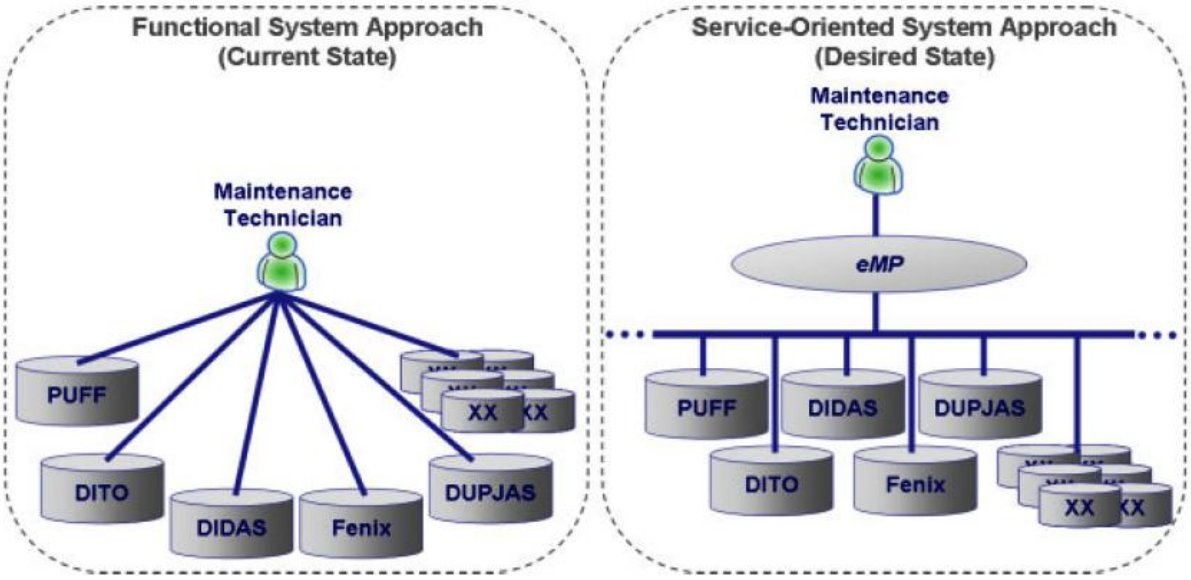


Figure 1-6 Illustration des besoins de technicien en maintenance dans un SI (Karim, Kajko-Mattsson, & Söderholm, 2008)

Les nouvelles technologies de l'information et de la communication (NTIC) ont permis d'asseoir ces nouvelles pratiques et de les faire évoluer. Grâce à l'émergence de ces technologies, la réalisation des services de maintenance et des contrôles peut être effectuée automatiquement, à distance et à l'aide de différents systèmes informatiques implantés au sein des entreprises. Un projet Européen PROTEUS (Bangemann, et al., 2006), a mis en place une première plateforme de e-maintenance proposant des systèmes d'aide à la maintenance à travers des services web.

Parallèlement, les outils de GMAO ont évolué et ont proposé d'avoir des connexions potentielles avec d'autres applications ou systèmes, comme par exemple « les lecteurs de valeurs mesurées », les appareils mobiles, des catalogues externes, des outils de CAO ou encore des sites Web (cf. Liebstückel, K. (2001), p.10).

La nouvelle génération de GMAO fournie sur le marché peut être vue comme un système d'aide à la décision ayant des objectifs de contrôle de coûts de maintenance et d'intervention, d'optimisation des ressources humaines et techniques comme les stocks de pièces de rechanges et les outils, de détails des installations techniques et de documentation associée et de mesurer d'efficacité des activités de maintenance (indicateurs de maintenance).

Dans cette génération des systèmes d'information orientés métier, les données et les informations liées à la maintenance ont été mises en place dans MIMOSA (Kahn, 2003) et les modèles structurés de données sont la référence en maintenance. Ces projets cherchent à intégrer toutes les applications de support de maintenance

[7] La collaboration : travailler ensemble pour atteindre un objectif (Collins English Dictionary, 2011).

grâce à des services web comme PROTEUS, y compris l'intégration de la maintenance proactive (DYNAMITE[8]) et les données du cycle de vie d'un produit (PROMISE[9]).

Une des préoccupations commune de ces projets était l'intégration de toutes ces applications dans un système d'information intelligent coopératif et distribué tout en assurant l'interopérabilité entre ces modules via une plateforme web distribuée basée sur des services Web (Muller, Marquez, & Iung, 2008). Ce type de système d'information de maintenance est appelé généralement plateforme de e-maintenance.

De plus, ces projets ont intégrés des outils de smart technologies dans ses plateformes comme le RFID, des capteurs intelligents, des appareils mobiles intelligents et des outils d'infotronic.

La cinquième génération qui au stade de la recherche est un système informatique qui doit répondre aux enjeux actuels de maintenance présentés dans la section 3.

En effet la plupart des recherches actuelles dans les systèmes informatiques sont axées sur la connaissance (Ermine, 2000), ceci a conduit les chercheurs de différents domaines à résoudre différents types de problèmes notamment des problèmes d'interopérabilité sémantique, à développer des applications sur la composition de services Web prenant appui sur des modèles de connaissance, à mettre en place des méthodes de self-maintenance (auto-maintenance) (Labib, 2006). Ces dernières applications de self maintenance utilisent des méthodes de raisonnement à base de règles ou de raisonnement floue inductif allié à des méthodes de raisonnement à partir de cas.

Par conséquent, la nouvelle génération de système d'information de maintenance doit prendre en compte non seulement les évolutions actuelles en matière de TIC sur les systèmes d'information orientés métier et utilisateur (le cas de la quatrième génération), mais aussi ceux orientés connaissance et qui seront à la base de cette cinquième génération. Pour cela, nous proposons comme cinquième génération des systèmes orientés connaissance, tout en exploitant les services Web, qui s'appuient sur l'ingénierie ontologique et les connaissances du domaine de maintenance. Ces systèmes peuvent générer de la nouvelle connaissance et sont à même d'exploiter les nouveaux concepts de l'informatique autonome (autonomic computing) (Ganek & Corbi, 2003), les services à la demande et le *« cloud computing »* (Hayes, 2008).

Le cœur de ce système est un système de gestion de connaissance capable d'inférer de la nouvelle connaissance et de fournir des services dynamiques grâce à des systèmes d'auto-apprentissage ; les systèmes d'auto-maintenance (self-maintenance), de retour d'expériences comme sources de génération de nouvelles connaissances.

[8] Dynamic Decisions in *Maintenance* : http://dynamite.vtt.fi/
[9] PROduct lifecycle management and Information tracking using Smart Embedded systems http://www.promise.no/

Tableau 1-1 Différentes générations de systèmes de gestion de maintenance

	1° génération	**2° génération**	**3° génération**	**4°génération**	**5°génération**
Système de Maintenance	Compétences de réparation manuelle	Gestion de Maintenance Assistée par ordinateur)	TTAO10 ; TMAO11 ; Systèmes Experts ; GMAO + ERP ; architecture CBM	e-maintenance ; plateforme coopérative distribuée de maintenance	s-maintenance ; systèmes d'auto-maintenance ;
Informatisation	Pas d'informatisation. Arrêter l'équipement quand il se casse	L'informatisation des procédures de maintenance. Suivi des interventions et des coûts de maintenance	Intégrer d'autres fonctions dans la fonction de maintenance ; Evolution dans les différents champs de la maintenance	Intégration de modules intelligents dans un système d'information orienté métier	Système d'information orienté connaissance utilisateur
Type de Maintenance	Exécution jusqu'à la défaillance ; Maintenance Réactive	Maintenance Réactive et Préventive	Surveillance conditionnée (condition monitoring) ; Maintenance Prédictive	Maintenance Proactive	Self maintenance (auto maintenance)
Analyse d'équipement		Composition de l'équipement ; Les plans et dessins des équipements	Analyse vibratoire, oil analysis, analyse d'huile, thermographie IR, ultrasons à chaud, Contrôle non destructif ;	AMDEC; analyse par arbre de défaillances; des technologies intelligentes	
Surveillance de l'équipement			Système d'acquisition, de supervision et de contrôle (SCADA)	Infotronique ; système de pronostics ; système de diagnostic ; capitalisation des connaissances ;	Infotronique, Retour d'expérience ;
Gestion des ressources		Gestion des interventions (panne et inspection), Révisions planifiées ; Systèmes de planification et de contrôle (diagrammes de PERT et de Gantt) ; Inventaire des pièces de rechange.	Planification de la maintenance ; Gestion des ressources humaines ; Gestion de logistique ;	Planification commune ; Production de la maintenance ;	Gestion intéractive des ressources et réutilisation des planifications passées.
Politiques et stratégies de maintenance	Maintenance centrée sur la fiabilité (RCM)	Coûts de maintenance réduits (Lower maintenance costs) ; Coûts de cycle de vie (Life Cycle Costing LCC) ; Indicateurs de maintenance (MBR, MBF) ;	Disponibilité supérieur de l'équipement ; Durée de vie prolongée de l'équipement ; Gestion de la qualité totale ; Maintenance en Condition Opérationnelle ;	Plus grande disponibilité, fiabilité et sécurité; Produit de meilleure qualité, vie plus longue ; Plus grande efficacité des coûts ; Pas de dommages à l'environnement ; Design pour la maintenabilité et la fiabilité	Système avec une redondance autorisée pour l'exécution jusqu' à l'échec ; Gestion du cycle de vie ; Combiner les outils de conception et les outils de maintenance ;

[10] Travaux Techniques Assistés par Ordinateur

[11] Techniques de Maintenance Assistées par ordinateur.

4.2- Caractérisation des systèmes de maintenance

Rasovska et al dans (Rasovska, Morello-Chebel, & Zerhouni, 2007) ont listé, caractérisé et classifié les différentes architectures des systèmes informatiques de maintenance existants suivant deux critères : l'évolution de l'information utilisée et la relation entre les systèmes intégrés dans les architectures. Quatre architectures génériques ont été identifiées, à savoir maintenance, télémaintenance, e-maintenance et s-maintenance figure1.7.

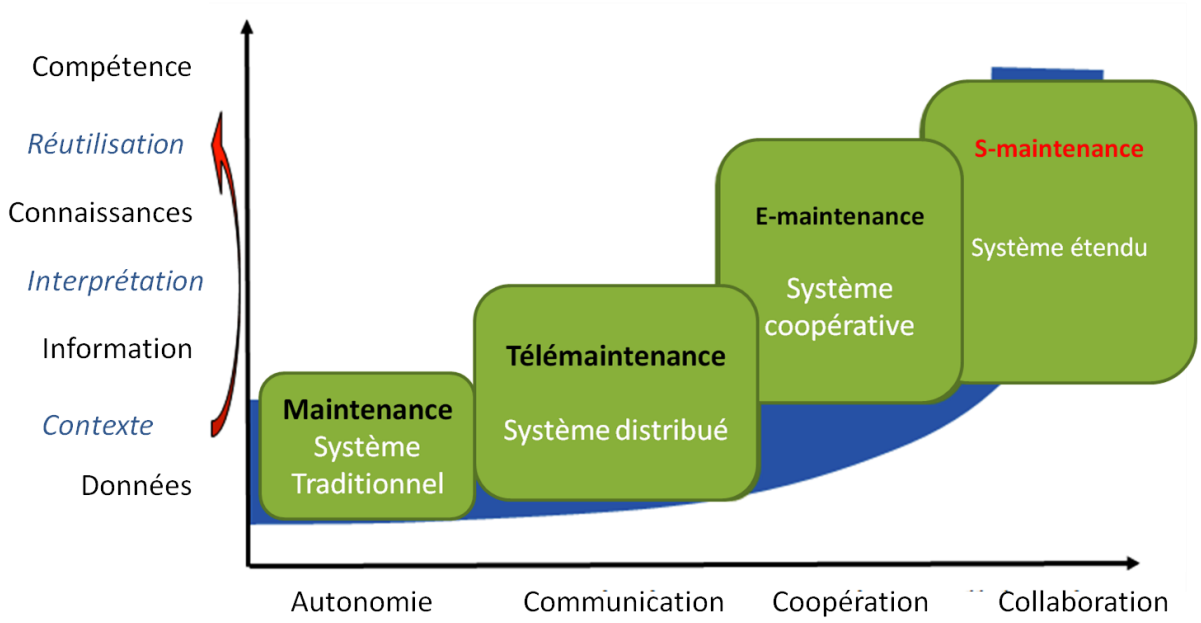


Figure 1-7 Classification des architectures des systèmes de maintenance

La figure 1-7 illustre le positionnement des architectures de maintenance en fonction de deux axes : le type d'information échangée et l'intensité des relations entre les systèmes intégrés dans la plateforme.

Les définitions relatives aux notions d'autonomie de communication, de coopération, de collaboration, ainsi que les données, information, connaissance et compétences sont fournies en annexe 2. Les architectures de maintenance sont classées suivant une exponentielle car la collaboration entre ces systèmes est atteinte plus tôt que le niveau de compétence partagée. Le volume des informations gérées automatiquement se concrétise par la surface de la place de chaque système et augmente avec l'intensité de la collaboration et à la complexité de l'information partagée (Rasovska, Morello-Chebel, & Zerhouni, 2007).

Ainsi, le système de s-maintenance prend appui sur le concept d'e-maintenance avec un échange d'informations non plus seulement sur les web services mais nécessitant des contraintes supplémentaires basées sur la sémantique de l'information échangée. Cette sémantique s'appuie sur une ontologie du domaine de maintenance commune aux différents systèmes. Elle permet d'utiliser et de créer des connaissances et des compétences qui aboutissent à la capitalisation des connaissances. Les systèmes collaborent, ce qui suppose un effort coordonné pour résoudre ensemble des problèmes.

5. La e-maintenance

L'objectif principal d'une plateforme d'intégration comme une plateforme de e-maintenance est de passer de la coexistence à la coopération des modules intelligents de maintenance dans le même environnement et de fournir les bonnes conditions pour l'orchestration de composants afin de fournir un service global intégré pour l'utilisateur de la plateforme.

5.1- Les définitions de la e-maintenance

5.1.1- définitions existantes

Le terme e-maintenance a vu le jour depuis le début des années 2000 et il est maintenant un terme très commun dans la littérature lié à une maintenance. Ingénieurs ou scientifiques peuvent considérer la e-maintenance en tant que concept, ou comme une philosophie, ou comme un phénomène, etc.

Selon Baldwin (Baldwin, 2004), le « e » dans la e-maintenance signifie « excellente » et présente la e-maintenance comme une agrégation de différents aspects à savoir :

E-Maintenance = Excellent-Maintenance = Efficient maintenance (faire plus avec moins de gens et moins d'argent) + Effective maintenance (amélioration des mesures FDMS) + Enterprise maintenance (contribuer directement à la performance de l'entreprise) (Baldwin, 2004).

Tout d'abord, un point crucial doit être clarifié, par analogie avec la programmation orientée objets nous tenons à dissocier le concept de e-maintenance (correspondant à une classe abstraite) de la notion de plate-forme de e-maintenance (correspondant à une classe), et une plateforme implémentée via des technologies spécifiques (correspond à l'objet) chacun ayant sa propre définition. Par conséquent, nous considérons que le concept de e-maintenance est un concept abstrait qui doit avoir une définition conceptuelle plus générale que la définition de la plateforme qui représente un modèle réalisant ce concept abstrait. De plus, la définition de la plateforme doit être aussi indépendante des technologies spécifiques utilisées dans une plateforme, car il peut y avoir différentes plateformes suivant le modèle de plateforme de e-maintenance utilisant des technologies différentes.

Ainsi, un des problèmes dans les travaux actuels est qu'il existe différentes définitions d'une plateforme de e-maintenance, définition dépendante du point de vue des auteurs de leurs expertises et de l'objectif visé.

Muller et al identifient en (Muller, Marquez, & Iung, 2008), des points de vue différents comme les stratégies de maintenance, les plans de maintenance, le type de maintenance et les supports de maintenance.

Certains considèrent la e-maintenance comme une stratégie de maintenance, où les tâches sont gérées électroniquement par l'utilisation de données temps réelles obtenues grâce aux technologies numériques et des technologies Internet (Tsang A. , 2002).

D'autres considèrent que la e-maintenance est un plan de maintenance exploitant des approches de CBM[12] (Tsang A. H., 1995), la maintenance proactive, la maintenance collaborative, la maintenance à distance (télémaintenance) et les services de support, approvisionner l'accès à l'information en temps réel, et l'intégration de la production à la maintenance (Ucar & Qiu, 2005). Un autre point de vue présenté par Koc et Lee (Koc & Lee, 2001) identifie le système de e-maintenance à un système de maintenance prévisionnelle, qui ne prévoit que la surveillance et les fonctions d'anticipation tel que le pronostic. Par contre Zhang et al (Zhang, Halang, & Diedrich, 2003) considèrent la e-maintenance comme la combinaison de technologie de services Web et la technologie des agents fournissant un moyen de réaliser des fonctions intelligentes et de coopération pour les systèmes d'automatisation industrielle. Crespo Marquez et Gupta (Crespo Marquez & Gupta, 2006) définissent la e-maintenance comme un environnement d'intelligence artificielle distribuée comprenant les capacités de traitement de l'information, d'aide à la décision et des outils de communication et de collaboration entre les processus de maintenance et les systèmes experts.

Nous pouvons remarquer par conséquent que chaque définition propose une vue partielle de la maintenance (par exemple maintenance prévisionnelle) et ne différencie nullement le concept de la plateforme. Une confusion s'instaure quand on relie le concept à un ensemble de technologies bien particulières comme les services Web ou les agents intelligents.

D'un point de vue métier, et plus particulièrement point de vue orienté intervention, la e-maintenance est définie par Karim (Karim, 2008) comme la part du support de maintenance qui assure que le processus de maintenance soit aligné avec les processus de fonctionnement et de modification afin d'obtenir les objectifs métiers exigés par les intervenants, grâce à une logistique d'information[13] (Willems, 2008) appropriée par l'utilisation des technologies d'information et de communication et la fourniture de services d'information.

D'un point de vue plus étendu, Moore et Starr (Moore & Starr, 2006) définissent la e-maintenance comme un réseau d'information de gestion des biens, qui intègre et synchronise diverses applications de maintenance et de fiabilité dans un seul système et permet de fournir des informations sur les biens là où c'est nécessaire quand c'est nécessaire.

Devant cette diversité de définitions et de points de vue une définition générique est fortement recommandée. Nous devons adopter une définition qui enveloppe le processus global de maintenance. À cette fin Muller et al dans (Muller, Marquez, & Iung, 2008) présentent une définition prenant en compte les différents points de vue et adoptent que la e-maintenance est une composante du concept d'e-manufacturing. Ils définissent ce concept comme un support de maintenance qui comprend les ressources, les services et la gestion nécessaire permettant une exécution proactive du processus de décision. Ce support comprend les e-technologies (les TIC, le Web, les réseaux sans fil, technologies d'Infotronics) mais aussi, les activités de e-maintenance (opérations ou de processus) telles que la e-surveillance, le e-diagnostic, le e-pronostic, etc.

[12] Condition Based Maintenance

[13] Information Logistics (IL) peut être définie comme «gérer et contrôler les processus de traitement des informations de manière optimale par rapport au temps (temps d'écoulement et la capacité), le stockage, la distribution et la présentation de telle manière qu'elle contribue aux résultats de l'entreprise en concurrence avec les coûts de la capture (création, recherche, etc entretien).

Comme le montre la Figure 1-8, Lee et al concluent que la e-maintenance est un pilier majeur dans les industries modernes qui prend en charge la réussite de l'intégration du e-manufacturing et e-business (Koc, Ni, Lee, & Bandyopadhyay, 2003).

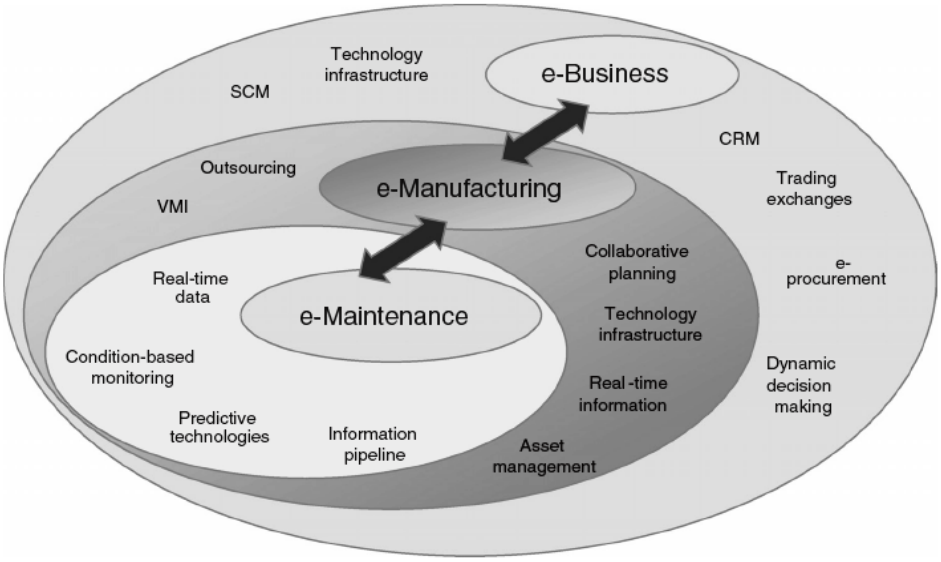


Figure 1-8 Inclusion de la e-maintenance dans le e-manufacturing et le e-business (Koc, Ni, Lee, & Bandyopadhyay, 2003)

Ces définitions sont insatisfaisantes et mélangent dans la plupart des cas concepts et technologies. Nous proposons une définition qui a comme ambition de tenir compte du processus global de maintenance, de resituer ce concept par rapport au business et de le rendre indépendant de la technologie afin de proposer une définition générique.

5.1.2- Proposition d'une définition de e-maintenance

Tant que la e-maintenance est inspirée du e-business elle peut être considérée comme l'un de ses concepts dérivés. En effet, nous adhérons au fait que la définition de la e-maintenance doit être inspirée de la définition du commerce électronique tout en respectant la définition de la maintenance. À cette fin, nous nous référons aux définitions de la maintenance et du e-business.

Diverses définitions du e-business sont présentées dans la littérature, nous adoptons celle présentée par Jones et al qui définissent le e-business comme *la réalisation des activités qui conduisent à un échange de valeur, où les parties interagissent électroniquement, en utilisant les technologies de réseau ou de télécommunications* (Jones, Wilikens, Morris, & Masera, 2000).

Sur la base de cette définition et celle de la maintenance, nous définissons la e-maintenance comme suit:

«La e-maintenance est la réalisation de la maintenance, où toutes ses actions ou activités techniques, administratives et de gestion interagissent et coopèrent par voie électronique, en utilisant les technologies de réseau ou de télécommunications ».

Cette définition prend en compte le processus de maintenance global présenté par Karim (Karim, 2008) (voir Figure I.5) pour inclure les différents points de vue de la maintenance et les différentes expertises (c.à.d. techniques, administrative et de gestion). Nous mettons en évidence l'interaction électronique et la coopération entre les actions et les activités assurées par les différents acteurs (humain ou logiciel) impliqués dans ce processus.

5.1.3- définition de la plateforme de e-maintenance

Nous définissons la plateforme de e-maintenance à partir du concept de maintenance. Sachant que ce dernier met le point sur l'aspect coopération, nous devons donc prendre en considération la définition d'une plateforme de coopération.

En effet, Les plateformes de coopération sont définies comme un environnement de travail logiciel composé de différents composants logiciels à partir desquels ceux-ci agissent ensemble pour atteindre des propriétés globales. Saint-Voirin affirme qu'une plateforme coopérative est vue comme un ensemble de collecticiels présentant des fonctions spécifiques regroupées dans une plateforme intégrée (Saint-Voirin, 2006).

Par conséquent, nous définissons une plateforme de e-maintenance comme :

« Une plateforme de e-maintenance est une plateforme de coopération offrant un ensemble de composants logiciels et de services logiciels intelligents d'aide à la maintenance (applications intégrées et / ou à distances) qui permettra aux acteurs de maintenance de communiquer et de travailler ensemble pour réaliser le processus complet de maintenance. ».

Dans cette définition, nous avons souligné les points d'intégration et de distributivité qui caractérisent la plupart des plateformes existantes et imposées par le monde réel de l'industrie. Nous nous sommes intéressés aux composants logiciels qui présentent les diverses applications de diagnostic, de pronostic, de surveillance, etc. à la communication et l'interopérabilité entre ces éléments.

5.1.4- Discussion et analyse des définitions

Dans les deux définitions que nous avons présentées, nous avons respecté la séparation et la différentiation entre le concept et la plateforme de e-maintenance.

Par rapport aux définitions existantes que nous avons passées en revue dans les sections précédentes, nous pouvons constater la généricité de notre définition qui se base sur la définition de l'AFNOR de la maintenance. En effet, cette définition met le point sur l'objectif de la e-maintenance qui est *la réalisation de la maintenance* en prenant en considération tout le processus de maintenance sans se limiter à un type ou une stratégie de maintenance *aussi performante soit elle,* ainsi que la prise en considération de l'aspect coopération et interactions électroniques. Ceci n'est pas vraiment le cas avec les définitions plus restreintes de Tsang (Tsang A. , 2002) qui limite la e-maintenance à une stratégie de maintenance et néglige l'aspect interactions en parlant juste d'utilisation de données en temps réel. De même Ucar et Qiu (Ucar & Qiu, 2005) se limitent à des types et stratégies de maintenance et en focalisant sur l'accès temps réel à l'information. De plus la définition fournie par Karim (Karim, 2008) n'est concentré que sur la partie intervention et néglige le reste du processus de

maintenance en se basant sur le contrôle, le stockage et la distribution de l'information qui est un aspect à traiter dans la définition de la plateforme et non pas celle du concept.

D'autre part, deux définitions sont proches de notre définition comme celle de Moore et Starr et celle de Muller et al. La définition fournie par Muller et al considère que la e-maintenance est un support de maintenance permettant une exécution proactive ce qui limite l'objectif de ce concept par rapport à la réalisation du processus de maintenance. En effet, on vise à faire de la maintenance proactive mais cela ne nous dispense pas de proposer d'autres types de maintenance et d'en tenir compte dans la plateforme. De plus, cette définition met le point sur des aspects technologiques qui doivent être placés dans la définition de la plateforme et non pas dans celle du concept. Elle se contente de parler d'activités de e-maintenance sans mentionner la relation et les interactions entre ces activités sachant que la définition fournie ne mentionne nulle part que la e-maintenance est un processus composé d'activités.

En ce qui concerne la définition de Moor et Starr, nous pouvons constater qu'il y a une séparation entre concept et plateforme. Mais ce qui limite cette définition c'est son orientation vers la gestion des biens (Asset Management) en considérant la e-maintenance comme un réseau d'information dans la gestion des biens. De plus, en ce qui concerne l'intégration, Moor et Starr parlent de synchronisation plus limitée que la coopération existante entre les applications intégrées dans la e-maintenance.

En ce qui concerne la définition de la plateforme, nous considérons que notre définition est aussi générique dans le but de couvrir les plateformes existantes utilisant telles ou telles technologies. Nous avons mis en évidence l'intégration en utilisant l'expression *«un ensemble de composants logiciels et de services logiciels intelligents* ».Ceci inclut aussi l'aspect intelligence que peut proposer des méthodes à base d'agents intelligents ou d'intelligence artificielle. Ainsi, nous avons mis en évidence l'aspect distributivité *(intégrés et / ou à distance)* ainsi que la coopération et la communication ce qui permet d'intégrer l'utilisation de n'importes quelles technologies réseaux.

En passant en revue les définitions existantes de e-maintenance, nous avons pu classer deux définitions comme définition de plateforme et non pas de concept de e-maintenance. En effet, Zhang et al (Zhang & al, 2003) spécifie l'aspect de coopération mais limite la plateforme à une combinaison des technologies de services web avec les technologies agents. Par contre la définition présentée par Crespo-Marquez et Gupta (Crespo-Marquez & Gupta, 2006) est plus évoluée que la première mais elle situe la plateforme dans un environnement d'intelligence artificielle distribuée, ce qui est spécifique à cette plateforme. Ainsi, nous remarquons une ambigüité dans cette définition quand les auteurs mentionnent que c'est un environnement *« Comprenant les capacités de,… et de collaboration entre les processus de maintenance et les systèmes experts ».* Ces deux définitions s'orientent vers un aspect particulier de la technologie employée et excluent un grand nombre de plateformes de e-maintenance existantes qui ne sont pas basées ni sur les services Web ni sur des méthodes d'intelligences artificielles.

5.2- Limites de la e-maintenance

L'intégration dans la e-maintenance est nécessairement faite par l'échange des données entre les composants et les applications intégrées. Ceci montre que celles-ci doivent faire face aux problèmes d'échange de données

entre sites, et de prendre en compte la diversité des formats des données et de grands volumes de données produites par les différentes applications. Ce qui induit des problèmes d'interopérabilité, de coopération et de partage de connaissances entre les applications, ayant une grande diversité de modèles d'exécution: temps réel pour SCADA[14], transactionnelles pour les données de documentation et l'ERP[15], interactif pour les dialogues homme machine et décisionnel pour les applications d'aide.

Les systèmes proposant différents formats d'information ne sont pas toujours compatibles pour le partage de données et de connaissances ce qui nécessite coordination et coopération entre ces systèmes pour les rendre interopérables. A ces fins, la plupart des plateformes existantes utilisent les Web services pour garantir l'interopérabilité technique mais pas sémantique entre ses différentes applications intégrées. La mise en place d'adaptateurs entre ces services web et la normalisation des données échangées sont toujours des tâches très compliquées, et ne traitent pas de la sémantique des données échangées.

L'interopérabilité est définie par le *« IEEE Computer Standard Dictionary »* comme «la capacité de deux ou plusieurs systèmes ou composants à échanger des informations et à utiliser les informations qui ont été échangées» (IEEE Computer Dictionary, 1990). A partir de cette définition, il est possible de décomposer l'interopérabilité en deux composantes distinctes : la capacité à échanger des informations, et la capacité à utiliser l'information une fois qu'elle a été reçue. Le premier processus est appelé «l'interopérabilité syntaxique», quant au deuxième, il est appelé «l'interopérabilité sémantique».

En effet, l'interopérabilité syntaxique assurée aujourd'hui par les plateformes de e-maintenance solutionne à moitié le problème, et doivent être orientée vers une interopérabilité sémantique assurant un niveau élevé d'échange d'information et même de connaissances, (Shedrof, 1999) pour garantir un échange efficace de l'information entre les applications hétérogènes de maintenance. Si l'interopérabilité sémantique n'est pas définie il peut survenir un conflit sémantique (Pollock, 2001).

Par ailleurs, concernant le partage de connaissances, la e-maintenance s'est penchée jusqu'à présent sur l'intégration et la coopération, ce qui a orienté les plateformes de e-maintenance vers l'échange de données et d'information et ne s'intéressent pas à cet aspect de partage de connaissances.

Pour répondre aux nouveaux besoins des acteurs de maintenance, nous proposons d'élaborer une plateforme devant assurer une bonne exploitation des connaissances par des raisonnements logiques permettant de faire évoluer le capital de connaissance du système de maintenance et de ses applications intégrées.

6. La s-maintenance : réponse aux nouveaux enjeux de la maintenance

Dans le but d'améliorer les performances des processus de maintenance fournis par la e-maintenance au niveau de la communication et de l'échange des données et de connaissances entre les systèmes ainsi qu'au niveau de l'exploitation de ces connaissances, nous proposons le concept de s-maintenance. Ce concept crée

[14] Supervisory Control And Data Acquisition
[15] Enterprise Resource Planning

un environnement d'information sémantiquement compatible, basé sur des concepts convenus entre différentes entités dans la maintenance. Dans ce cadre, il donne une solution pour la réalisation de l'interopérabilité sémantique au niveau de la plateforme de maintenance. Ainsi, l'objectif principal de ce concept est de répondre aux besoins évolutifs, aux exigences des acteurs de maintenance qui sont les utilisateurs réels de ces systèmes et de remédier aux limites de la e-maintenance. Le besoin principal peut se résumer par «Avoir la bonne information au bon format pour les bonnes personnes pour faire les bonnes choses au bon moment » (Lee & Wang, 2008). Par conséquent, ce concept ne s'arrête pas à la mise à disposition des services intégrés dans la plateforme comme le cas de la e-maintenance, mais il va plus loin en fournissant des services adaptés dynamiquement à la demande, et en assurant des services d'auto-X sans interventions humaines.

Afin de faire face aux enjeux de demain, et de proposer un système capable de donner une solution durable, la gestion de maintenance à besoin d'un système dynamique, évolutif (l'intelligence du système évolue au fur et à mesure de son exécution), capable d'analyser ses comportements et d'en tirer une expérience.

6.1- La définition du concept de s-maintenance

Nous rappelons que la définition du concept de s-maintenance prend appui sur la définition de la e-maintenance qui est générale et qui dépend elle-même de la définition de la maintenance. Mais des contraintes sont imposées orientant ainsi la réalisation de ce concept vers la connaissance.

La s-maintenance « est la réalisation de la maintenance basée sur la connaissance experte du domaine, où les systèmes dans le réseau gèrent ces connaissances et partagent la sémantique faisant émerger de nouvelles génération de services et offrant des services à la demande, grâce à des fonctionnalités adaptatives et autonomes ».

Gérer la connaissance ceci entend la formalisation, l'acquisition, le raisonnement, lamaintenance, l'exploitation et la réutilisation de la connaissance ainsi que sa réalimentation (Menzies, 2000). Par nouveaux services de maintenances, nous entendons des services dynamiques (ayant des comportements évolutifs) adaptables aux besoins des utilisateurs grâce à des fonctionnalités d'auto-X [self-x] comme l'auto-apprentissage, l'autogestion, etc.

Par service à la demande, nous entendons nouveau service dont le résultat généré est par des activités entre composants du système et utilisateur, par des activités internes au système, destiné à répondre à une demande, non fournie par les services existants et exprimée par un utilisateur.

6.2- Définition des fonctionnalités d'auto apprentissage et d'autogestion

La fonctionnalité d'auto-apprentissage (self-learning) peut être définie comme *« la fonctionnalité qui permet au système tout au long de son exécution et grâce à des connaissances de départ, d'apprendre automatiquement, de créer ainsi de nouvelles connaissances et de les exploiter lors de futures exécutions »*. Par conséquent, cette fonctionnalité permettra au système concerné de faire évoluer son degré d'intelligence.

D'autre part, la fonctionnalité d'autogestion (self-management) est définie comme *« la fonctionnalité qui permet au système grâce à des connaissances sur ses comportements, de gérer automatiquement et sans intervention humaine quelques activités appartenant au processus métier gérer par celui-ci »*. Ceci permettra donc au système concerné de faire évoluer, réorganiser et adapter ses comportements (les processus) lors de son exécution.

6.3- Définition d'une plateforme de s-maintenance

La collaboration est l'une des caractéristiques de base de la s-maintenance en sachant que tout système dans le réseau agit collectivement dans un même objectif en utilisant et en partageant les ressources communes à savoir les connaissances expertes du domaine dans ce cas. Par conséquent, pour définir la plateforme de s-maintenance nous devons prendre en considération la définition de la plateforme de collaboration.

En effet, les plateformes de collaboration sont des plateformes électroniques unifiées qui facilitent la communication synchrone et asynchrone à travers une variété de dispositifs. Les plateformes de collaboration offrent un ensemble de composants logiciels et de services logiciels permettant aux individus de trouver les uns et les autres, les informations dont ils ont besoin et d'être capable de communiquer et de travailler ensemble pour atteindre des objectifs métiers communs.

D'où, la définition d'une plateforme de s-maintenance concrétisant le concept de s-maintenance:

« Une plateforme de s-maintenance est un système collaboratif et distribué basé sur l'ingénierie des connaissances fournissant des services dynamiques et des services à la demande selon les exigences de ses utilisateurs grâce à des fonctionnalités d'autogestion des processus de maintenance et d'auto-apprentissage ».

Il est à noter aussi que nous entendons par services dynamiques les services évolutifs ayant la capacité d'adapter leurs comportements aux différents contextes d'utilisation.

Ainsi, ce système basé sur l'ingénierie des connaissances[16] prend appui sur la sémantique de la connaissance experte du domaine de maintenance, et a la possibilité de faire évoluer son degré d'intelligence. A cette fin, nous utiliserons comme cœur de la plateforme un système à base de connaissances[17] permettant d'inférer une nouvelle connaissance et de l'exploiter à partir d'une ontologie du domaine de maintenance. Ceci fera l'objet du chapitre 4.

[16] À distinguer de la gestion des connaissances, l'ingénierie des connaissances, fait référence à l'ingénierie de systèmes intelligents incorporant beaucoup de connaissances tels les Systèmes Experts.

[17] Les systèmes à base de connaissances (Knowledge Based System en anglais) sont des outils d'intelligence artificielle qui fonctionnent sur un domaine étroit pour fournir des décisions intelligentes avec justification. Les avantages offerts par un tel système sont la documentation des connaissances, aide à la décision intelligente, auto-apprentissage, le raisonnement et l'explication. Les composants de base de ces systèmes sont une base de connaissances, des mécanismes d'acquisition de connaissances et des mécanismes d'inférence (Akerkar & Srinivas, 2009).

La Figure 1-9, montre la relation d'inclusion entre les plateformes de maintenance, de e-maintenance et de s-maintenance et définit à partir de chaque concept leurs caractéristiques.

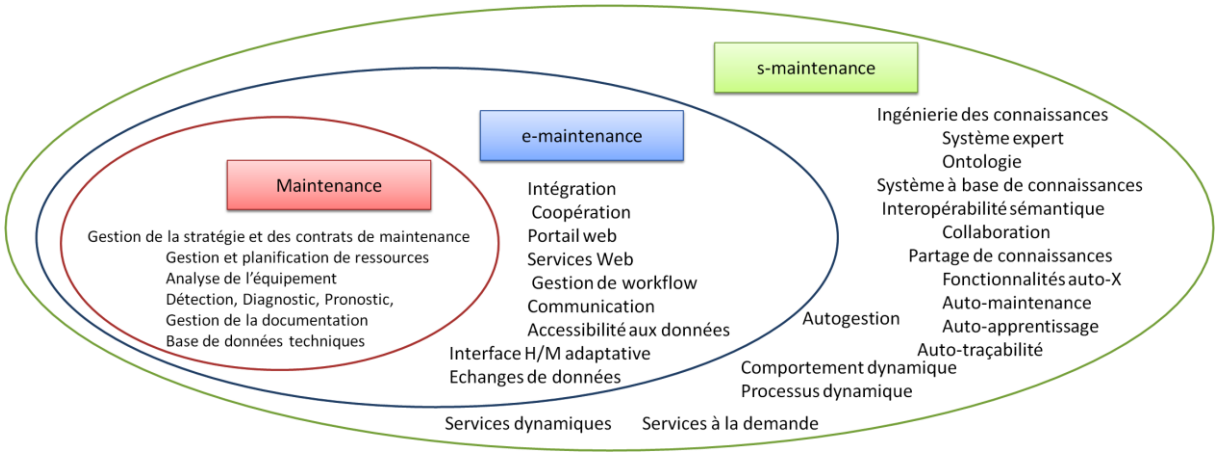


Figure 1-9 Inclusion des fonctionnalités dans les plateformes de maintenance, e-maintenance et s-maintenance

Nous constatons que la plateforme de s-maintenance englobe les fonctionnalités de la plateforme de e-maintenance, et transforme un outil d'intégration en un acteur intelligent au cœur du processus de maintenance. En outre, les services fournis par un système de maintenance ou de e-maintenance sont inclus dans un système de s-maintenance comme indiqué dans la figure 1-9.

Ces systèmes informatiques de maintenance fournissent des fonctionnalités à valeurs ajoutées pour les opérateurs de maintenance (end users). Nous nous intéressons au problème de réutilisation des connaissances et de consultation de nouveaux indicateurs dans le cadre d'un système de maintenance traditionnel, de e-maintenance et de s-maintenance.

La réutilisation des connaissances, par un nouvel employé remplaçant une personne partie n'est pas évidente. Elle n'est possible ni dans une GMAO ni dans le cadre d'une plateforme de e-maintenance, car aucun module spécifique, sur l'expertise demandée, n'est censé être intégré dans ces outils logiciels. Ceci étant dû à la non foramlisent des connaissances. Par contre les fonctionnalités de la plateforme de s-maintenance de management de connaissances (exploitation de l'ontologie, auto-apprentissage, etc.) permettent d'obtenir de nouvelles connaissances pouvant être réutilisées par les nouveaux arrivants.

Nous nous intéressons maintenant au besoin de nouveaux indicateurs par le *end-user*. Dans une plateforme de e-maintenance ou d'une GMAO, les services ou applications intégrés fournissent des indicateurs prédéfinis. Si l'utilisateur d'une plateforme de e-maintenance a besoin d'un nouvel indicateur, il doit demander à un fournisseur, ou à un informaticien de lui développer un service destiné à cet effet. Par contre, dans le cadre de la plateforme de s-maintenance, grâce au service à la demande que la plateforme fournisse, l'utilisateur peut exprimer ses besoins de manière semi-formelle en utilisant la base de connaissance de la plateforme. La plateforme grâce à son moteur d'inférence lui fournit l'indicateur demandé.

7. Conclusion

Les services, les fonctionnalités et les indicateurs fournis par les systèmes informatiques de maintenance n'évoluent pas automatiquement avec les besoins des utilisateurs. Ces besoins peuvent se résumer par la maitrise et le contrôle du processus complet de maintenance, la réutilisation et l'exploitation des historiques relatifs aux différents types de retour d'expériences et par des outils et techniques intelligents supportant l'aide à la décision.

Afin de remédier à l'insatisfaction des utilisateurs quant à leurs besoins et aux limites des systèmes existants, nous avons étudié les enjeux auxquels doit faire face les futurs systèmes informatiques de maintenance.

Les grands enjeux identifiés à l'issue de cette étude sont la synchronisation entre les différentes applications de support (coopération, collaboration), la standardisation (langage commun), le traitement intelligent, la réutilisation et l'exploitation des connaissances métier, la mise en place de différents modules autonomes[18] intelligents évolutifs orientés vers un système d'utilisateur intégrant les nouvelles technologies performantes.

Afin de situer ces enjeux par rapport aux différentes générations de systèmes de maintenance et de monter l'insatisfaction des utilisateurs, nous avons été amenés à étudier l'évolution de ces systèmes, leurs caractéristiques et leurs fonctionnalités.

Cette étude nous a permis de recenser cinq générations de maintenance, la première basée sur les compétences des opérateurs humains, la deuxième concernant les systèmes de gestion de maintenance assistée par ordinateur (GMAO), la troisième connectant la GMAO aux différentes fonctions de l'entreprise, la quatrième intégrant des briques logicielles intelligentes dans une plateforme d'échange d'informations, plateforme de e-maintenance, la cinquième orientée connaissance, garantissant l'interopérabilité sémantique entre les applications de la plateforme, nommée s-maintenance.

De manière générale, malgré la différence de niveau par rapport aux services fournis par ces systèmes, les aspects principaux que nous avons remarqués et qui limitent leurs réponses aux enjeux sont le manquede réutilisation et d'exploitation intelligente des connaissances expertes du domaine, l'utilisation deconnaissances non standardisées et le faible niveau de synchronisation qui se limite à la communication et rarement à la coopération.

Un état de l'art sur le concept de e-maintenance nous a amené à constater, que la plupart des travaux dans le domaine ne différencient pas le concept, la plateforme et l'utilisation des technologies.

Par conséquent, nous avons proposé deux définitions, une concernant le concept de e-maintenance caractérisé par l'interaction coopérative et électronique entre les différentes actions du processus global de maintenance, et l'autre développant un modèle de plateforme caractérisée par l'intégration des modules informatiques intelligents réalisant les activités du processus de maintenance.

Ainsi, partant des limites de ce concept , nous l'avons fait évoluer vers le concept de s-maintenance.

La s-maintenance propose une réalisation de la maintenance basée sur le partage des connaissances expertes du domaine en fournissant des services adaptatifs et autonomes. Ce concept est concrétisé par une plateforme qui

[18] Fait référence à « autonmic computing » .

formalise ces connaissances et les partage entre les différentes applications intégrées dans le système, ce qui garantit une interopérabilité technique et sémantique.

Basé sur un système de gestion des connaissances comportant un moteur d'inférence supportant différents types de raisonnement, la plateforme doit fournir dynamiquement des services à la demande (pour les indicateurs définis par l'utilisateur) ainsi qu'assurer des fonctionnalités auto-X (connus par *self-X functionalities* en anglais). Parmi ces fonctionnalités nous trouvons l'autogestion des processus de la plateforme et l'auto-apprentissage relatif aux interactions des utilisateurs avec la plateforme pour assurer la dynamicité de la base de connaissance de celle-ci.

Après avoir répondu à la question sur la caractérisation des nouveaux systèmes intelligents de maintenance, et une partie de la question sur la démarche à entreprendre pour la mise en œuvre de cette nouvelle génération de systèmes, les chapitres suivants répondront aux autres questions portant sur la réalisation de ce nouveau concept. Le deuxième chapitre sera donc consacré à faire un état de l'art des plateformes de maintenance existantes afin de les analyser par rapport aux critères définissant la plateforme de s-maintenance et exploiter ou proposer l'architecture la mieux adaptée pour élaborer une plateforme de s-maintenance.

Chapitre 2 Étude et analyse des plateformes de maintenance existantes

1. Introduction

Une plateforme de e-maintenance est caractérisée principalement par la coopération et l'intégration des composants logiciels intelligents de support, tandis qu'une plateforme de s-maintenance prend appui sur la collaboration et l'intégration de fonctionnalités autonomes (auto-apprentissage et autogestion du processus de maintenance). Les points clés communs entre ces plateformes sont l'intégration, la synchronisation et la prise en compte du processus complet de maintenance.

Dans l'objectif de mettre en place une plateforme de s-maintenance, nous nous sommes intéressés aux plateformes existantes classées sous la dénomination e-maintenance.

Avant toute chose, nous faisons un tour d'horizon sur les différentes études et revues qui ont été faites sur l'élaboration de plateforme logicielle concernant des projets de maintenance aussi bien théoriques qu'académiques.

Dans le cadre de la maintenance conditionnelle (CBM) Jardine et al (Jardine, Daming, & Banjevic, 2006) ont présentés une revue sur les systèmes de diagnostic et de pronostic, en mettant l'accent sur les modèles, les algorithmes et les technologies utilisés pour le traitement des données et la prise de décision.

Campos (Campos, 2009) a fait une étude sur l'application et l'intégration des technologies d'informations et de communications et plus spécifiquement sur les outils d'intelligence artificielle, les technologies des agents et le web dans le cadre de la surveillance conditionnelle et la maintenance. Campos classifie ces systèmes selon les technologies utilisées et les catégorise par rapport aux couches OSA-CBM et conclu à la limitation d'exploitation des TIC dans ce cadre.

D'autre part, Levrat et al (Levrat, Iung, & Crespo-Marquez, 2008) ont étudié les différentes définitions de la e-maintenance et ont proposé un cadre de e-maintenance basé sur le framework de Zachman (Sowa & Zachman, 1992), dans l'objectif de donner une méthodologie de déploiement d'un projet de e-maintenance à l'aide de services, de processus, d'organisation et d'infrastructures. Cette étude était plutôt orientée sur la e-maintenance. Par contre, dans (Muller, Marquez, & Iung, 2008) Muller et al après avoir examiné les différents points de vue et définitions de la e-maintenance, ont passé en revue la plupart des systèmes de e-maintenance existantes en les classant selon les objectifs sous jacents à leurs constructions et leurs champs d'exploitation.

Dans ce chapitre, nous allons faire un état de l'art sur les plateformes de e-maintenance existantes qu'elles soient théoriques ou issues du monde industriel.

Par ailleurs, l'étude que nous ferons dans la deuxième section de ce chapitre analyse ces plateformes suivant quatre caractéristiques en lien direct avec la définition du concept de s-maintenance à savoir :

1. Le type de maintenance traité (CBM, diagnostic, processus complet, …)
2. L'interopérabilité entre les applications de la plateforme
3. Services dynamiques et ingénierie des connaissances au cœur de la plateforme
4. Le degré d'intégration (fonction de coopération, collaboration)

Cette analyse a comme objectif l'identification de la ou des plateformes en concordances avec la définition de la plateforme de s-maintenance. Nous en déduirons ainsi s'il y a lieu de recycler une plateforme de e-maintenance ou de redévelopper une nouvelle plateforme de s-maintenance avec sa propre architecture.

2. Principales plateformes de maintenance existantes

2.1- Introduction

Plusieurs plateformes de e-maintenance ont été développées ces dernières années, et elles sont opérationnelles aujourd'hui (CASIP, ICAS-AME, INID, WSDF, PROTEUS, TELMA, etc.). Muller et al (Muller, Marquez, & Iung, 2008) constatent qu'il y a quatre types de plateformes : les plateformes propriétaires (i.e. ICAS), les plateformes développées dans des projets européens (i.e. PROTEUS, DYNAMITE) ou bien des plateformes de recherche et/ou d'éducation (i.e. TELMA) et enfin des plateformes issues d'étude théorique (Ribeiro, Barata, & Silvério, 2008) (Han & Yang, 2006). Ils classent ces contributions selon les capacités et les besoins qui sont destinés à être satisfait. Ces différentes contributions sont développées pour la e-maintenance afin de répondre à une (ou plusieurs) des quatre questions suivantes: (1) l'établissement de normes, (2) la conception d'une plateforme de e-maintenance, (3) la formalisation des processus de e-maintenance, et (4) la mise en place de système d'e-maintenance (c.à.d. plateforme + processus).

Nous allons étudier dans cette section deux types de plateformes à savoir des plateformes de projets instanciés entre des industriels et des universitaires et des plateformes académiques lancées par des universitaires et des groupes de chercheurs.

2.2- Plateformes de Projets

Nous ferons l'étude de sept plateformes définies dans le cadre de projets d'envergures : le projet international MIMOSA, les projets européens ESPRIT-REMAFEX, PROTEUS, PROMISE, SAMMART, et DYNAMITE et finalement le projet suédois e-Maintenance 24/7–NFFP4.

2.2.1- Le projet MIMOSA

La première initiative était prise par le projet MIMOSA (The Machinery Information Management Open Systems Alliance) pour développer un système complexe d'information unique pour la gestion de la maintenance (Kahn, 2003). Le projet a eu pour objectif de développer un réseau de collaboration de maintenance en proposant la norme open de protocole EAI (Enterprise Application Integration). L'organisation préconise et développe des caractéristiques d'intégration de l'information pour permettre la gestion et le contrôle de la valeur ajoutée par les solutions ouvertes, intégrées et orientées vers l'industrie. Les solutions développées à partir des îlots d'informations afin de créer une plateforme d'e-maintenance ont été proposées dans ce projet (Mitchell, Bond, Bever, & Manning, 1998).

Une architecture fonctionnelle OSA/CBM (Open System Architecture for Condition-Based Maintenance) (Thurston, 2001) dédiée au développement de stratégies de maintenance conditionnelle ou prévisionnelle (Lebold & Thurston, 2001) a été développée à partir du schéma relationnel MIMOSA CRIS pour répondre au

besoin d'une norme concernant le flux transactionnel de l'information entre les composants logiciels dans un système CBM et en vue de pour garantir l'interopérabilité entre eux. Elle contient sept modules flexibles dont le contenu (méthodologie et algorithmes) est configurable par l'utilisateur (Figure 2-1). Elle peut être simplifiée et adaptée à chaque besoin industriel en réduisant des modules.

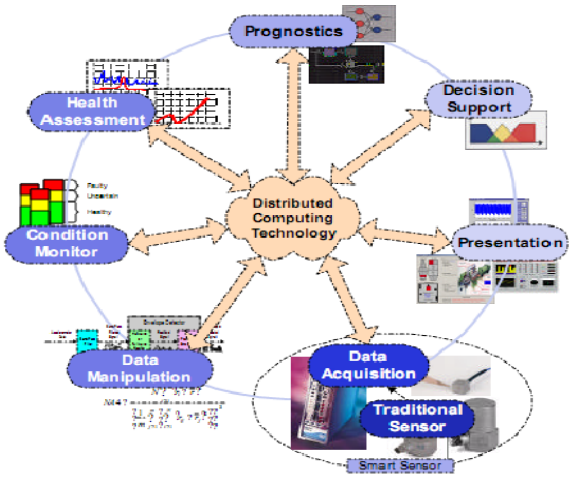


Figure 2-1. Architecture MIMOSA- OSA/CBM (Lebold & Thurston, 2001)

Type de maintenance :

En effet, OSA/CBM a mis le point sur les différentes couches présentant le cadre complet du processus de maintenance mais sans prendre en considération la partie gestion. Ainsi il ne traite qu'un seul type de processus à savoir le CBM (Condition Based Maintenance).

Interopérabilité :

Concernant l'aspect d'interopérabilité, MIMOSA a traité l'interopérabilité syntaxique en fournissant le schéma de données partagé CRIS. Mais ceci n'assure pas un niveau d'interopérabilité sémantique.

Gestion des connaissances :

Aucune démarche d'ingénierie ou de management de connaissances ni de services dynamiques évoluant selon les besoins de l'utilisateur n'est donné. En effet, MIMOSA s'intéresse à la normalisation et la standardisation des flux transactionnels dans un système complexe de maintenance.

Mécanisme de synchronisation :

Dans MIMOSA on parle de mécanisme de collaboration sans plus de détail. Toutefois, il n'y a pas vraiment du partage de ressource, ce qui nous oriente plutôt vers une coopération entre les composants de ce système.

2.2.2- POMAESS

Dans le cadre du projet ESPRIT-REMAFEX, Yu et al proposent un système d'e-maintenance basé sur un système multi-agents fournissant une aide à la décision grâce au raisonnement à partir des cas (Yu, Iung, & Panetto, 2003). Celui-ci se nomme «Problem-Oriented Multi-Agent based E-Service System (POMAESS) ». Chaque agent dans ce système est un agent expert de résolution de problèmes (homme ou machine autonome) avec des connaissances différentes et une localisation dans des sites différents.

Le but principal de POMAESS est de créer un système qui interconnecte les agents séparés (agents réactifs et cognitifs), leur donner la possibilité de coopérer entre eux dans un environnement ouvert tout en respectant des contraintes temporelles. Ce qui permettra ainsi à l'ensemble des agents à opérer au-delà des capacités de l'un de ses membres et de résoudre les problèmes au sein du processus industriel.

Nous rappelons quand même que l'architecture de la coordination dans POMAESS est une combinaison du modèle de coordination maître-esclave (voir la structure tableau noir et la formation du comité). A cela deux types d'agent sont adoptés par le système, agent de négociation (NA), qui lance et organise la négociation, calcule le résultat mis sur la base des propositions par l'agent expert (EA), mais il ne donne pas son opinion sur la solution du problème. La négociation et la coordination entre agents se fait par rapport a un ensemble de règles déjà définies. L'application de ces règles permet de trouver une solution, qui pourrait être apprise et réutilisée dans d'autres situations (illustration sur la Figure 2-2).

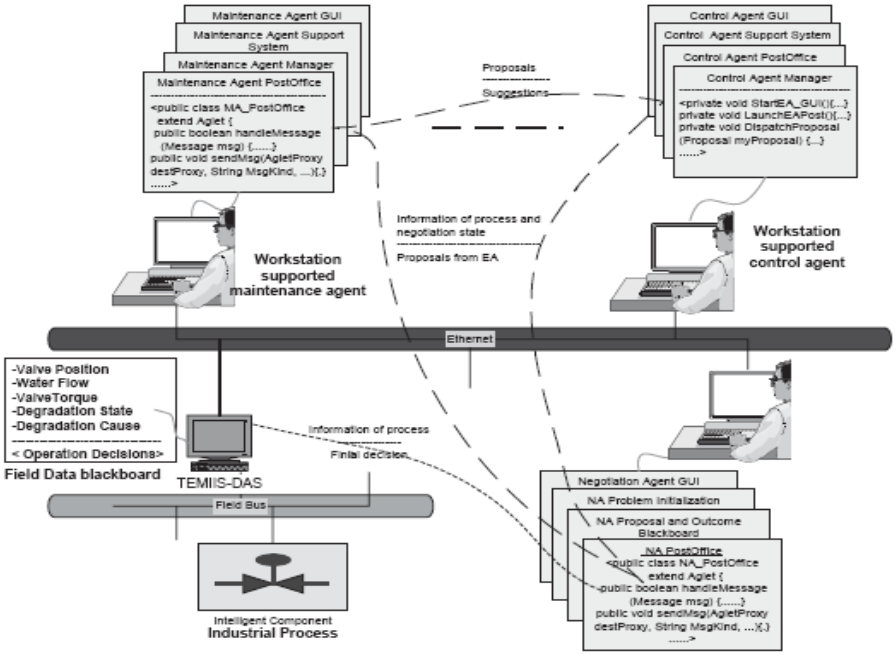


Figure 2-2 Architecture de négociation des agents dans POMAESS (Yu, Iung, & Panetto, 2003)

Type de maintenance :

La plateforme proposée dans ce travail vise la résolution des problèmes et est une plateforme d'aide à la décision dédiée principalement au diagnostic, et n'assure pas une gestion complète du processus de maintenance.

Interopérabilité :

La négociation entre les agents est faite à base de règles prédéfinies, mais la communication entre ces agents ne paraît pas être standardisée et n'a pas de sémantique partagée.

Gestion des connaissances :

Les connaissances dans cette plateforme sont sous forme de règles de négociations, mais la résolution de problèmes n'est pas intégrée comme étant des connaissances globales du domaine de maintenance.

Nous tenons à rappeler ici que chaque agent utilise des connaissances locales et non pas partagées.

En ce qui concerne la proposition de service dynamique à la demande, ceci n'était pas inclus ou pensé dans ce travail. Le service fourni d'aide au diagnostic est statique.

Mécanisme de synchronisation :

Nous savons que les règles de négociations sont partagées entre agents grâce à des mécanismes mis en place. Il n'y a donc pas de collaboration entre agents pour la prise de décision, mais il y a un mécanisme de coopération.

2.2.3- PROTEUS

Dans le cadre du projet Européen PROTEUS une première plateforme d'e-maintenance fournissant des services web a été développée (Bangemann, et al., 2006). Le principal objectif du projet est le développement d'une plateforme logicielle pour l'intégration et l'interopérabilité des modules logiciels de et de maintenance à distance. L'idée originale du projet dédiée à la maintenance industrielle, est liée à l'intégration de tout outil nécessaire à l'activité de maintenance, dont les fonctions vont de la détection des alarmes à la gestion des pièces de rechange, avec comme finalité d'optimiser les coûts et améliorer la productivité de l'équipement maintenu. L'objectif de la plateforme PROTEUS vise à l'intégration de ces sous-systèmes grâce à une description de l'équipement unique et cohérente (à travers une description d'ontologie de l'équipement), une architecture générique (basée sur la technologie "Web Services") et des modèles cohérents de composants hétérogènes et des solutions technologiques d'intégration (Figure 2-3). Ces techniques permettent de garantir l'interopérabilité de systèmes hétérogènes afin d'assurer l'échange et le partage des informations, des données ainsi que des connaissances.

Bien que PROTEUS ait touché plusieurs points et ait été considéré comme une révolution en 2006 dans les systèmes de maintenance, il ne répond plus aux recommandations et aux besoins actuels d'un système de s-maintenance.

Type de maintenance :

L'un des objectifs de PROTEUS est intégrer tous les outils et applications autour de la maintenance et permettant de couvrir tout le processus de maintenance. Sur ce point PROTEUS est l'un des rares projets à avoit une vision complète sur le processus de maintenance.

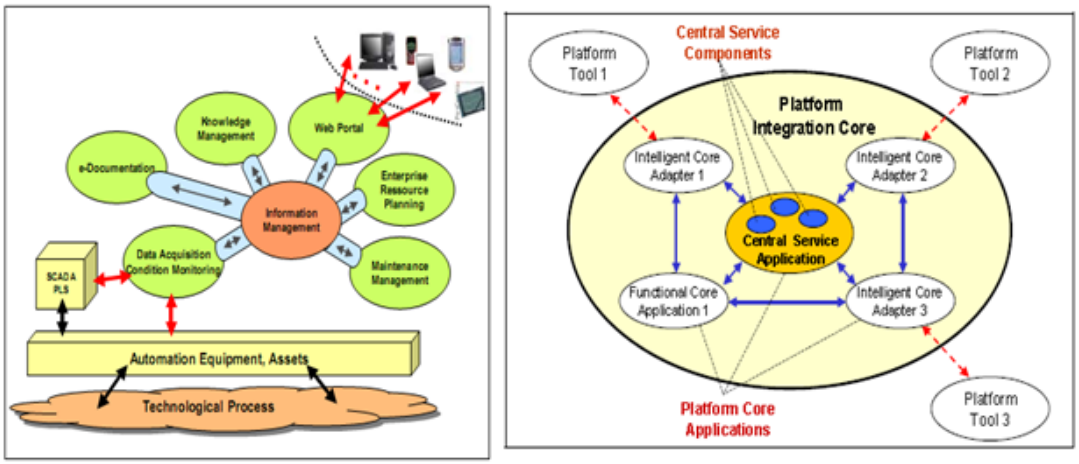


Figure 2-3 Architecture d'intégration de PROTEUS (Bangemann, et al., 2006).

Interopérabilité :

PROTEUS traite du problème d'interopérabilité au niveau syntaxique via les services web ce qui permet un échange facile des données sans le partage des connaissances entre ses services.

Gestion des connaissances :

Concernant l'ingénierie et le management des connaissances, PROTEUS se base sur un modèle de connaissances de l'équipement. Ce dernier est exploité par le service de diagnostic basé sur le raisonnement à partir de cas, tout en contenant une mémoire d'entreprise. Toutefois, il n'y a pas d'ontologie du domaine de maintenance dans la plateforme.

Ainsi, PROTEUS a prévu l'évolution de la plateforme via l'intégration de nouveaux services qui ne sont pas dynamiques.

Mécanisme de synchronisation :

La nature « boite noire » des services élimine la possibilité de partage des ressources internes entre ces services. De plus, comme tout système orienté service, la plateforme PROTEUS utilise un modèle d'orchestration entre les services d'où l'aspect de coopération.

2.2.4- PROMISE

Le projet européen PROMISE *(PRoduct lifecycle Management and Information tracking using Smart Embedded systems)* a pour objectif la gestion des flux d'informations au-delà du client, de clôturer la boucle d'information de gestion du cycle de vie du produit *(PLC, Product Life Cycle)* et de transformer les données brutes du PLC à des connaissances exploitables (Kiritsis, 2004).

Une architecture de plateforme a été proposée à l'issue de ce projet comme le montre la Figure 2-4. Cette architecture utilise un modèle peer-to-peer pour l'échange d'informations, où tout dispositif implémentant l'interface de messagerie de PROMISE (PMI-PROMISE Messaging Interface) basée sur les services Web peut communiquer avec tout autre dispositif prise en charge par le PMI, peu importe la taille de l'appareil (Främling & Nyman, 2008). L'interface PMI est essentielle et permet une approche basée sur les web-services.

La plateforme utilise la PMI pour la transmission des informations codées en XML principalement destinées à la transformation automatique des systèmes d'information.

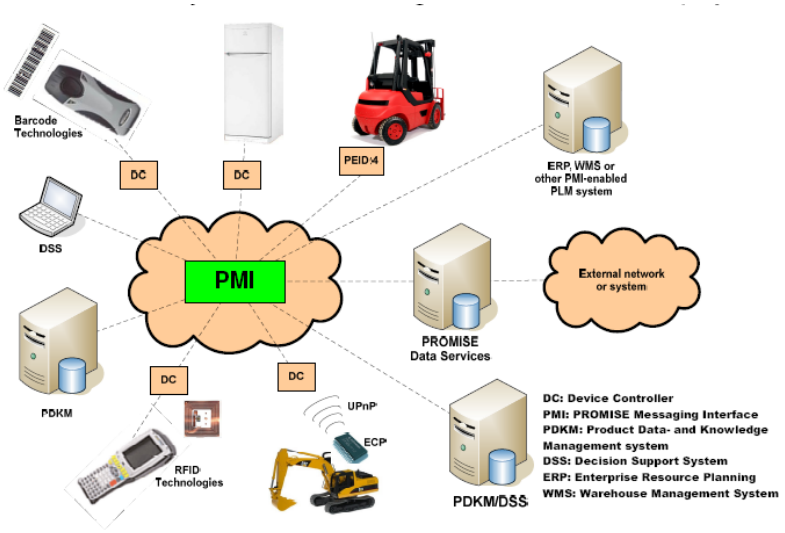


Figure 2-4 Architecture de la plateforme de PROMISE (Främling & Nyman, 2008)

Gestion des connaissances :

Dans le cadre de ce projet, un modèle ontologique appelé SOM *(Product Data and Knowledge Management Semantic Object Model)* représenté par un diagramme de classe UML fournit les concepts génériques du cycle de vie d'un produit ainsi que les relations entre ces concepts. L'objectif du SOM est d'avoir une sémantique commune entre les différents intervenants durant le cycle de vie d'un produit. Bien que l'architecture proposée intègre tout un systeme de gestion de connaissances (le PDKM, voir figure 2-4) et un système d'aide à la décision basé sur la connaissance enregistrée dans le PKDM. Le modèle ontologique n'est pas exploité par des méthodes d'ingénierie des connaissances dans l'architecture de plateforme proposée. . L'approche générale du PDKM est d'intégrer les données relatives aux produits de toutes les phases, qu'elles proviennent de bases de données ou de PEIDs (Product Embedded Information Device), et de permettre l'intégration des fonctions d'analyse en vue de soutenir les processus de décision dans toutes les phases du cycle de vie. En outre PDKM permet de gérer les connaissances sur les produits qui peuvent être facilement récupérées par les utilisateurs et présentées dans leurs contextes respectifs. Toutefois, le système de gestion de connaissances n'est exploité que par l'utilisateur mais non pas par les applications et services intégrés dans le système.

Type de maintenance :

. L'architecture de PROMISE a été conçue pour répondre aux besoins spécifiques du PLM (Product Life Cycle Management) et permette la récupération et la mise à jour d'informations sur les produits durant tout leurs cycle vie, et ne s'est pas interssé au process complet de maintenance.

Interopérabilité :

L'architecture basée sur les services Web et le modèle SOM nous ont permis de constater que PROMISE propose une solution d'interopérabilité syntaxique et sémantique. Rappelons que l'avancée technologique proposée par PROMISE a introduit l'intégration de nouvelles technologies comme le RFID, et les réseaux informatiques, mais ne s'est pas intéressés à la dynamicité des services.

Mécanisme de synchronisation :

L'approche Peer-to-peer adoptée par PROMISE facilite la communication, l'orchestration et la collaboration entre les serveurs de données et les services web du PMI.

2..2.5- SMMART

Le projet européen SMMART (System for Mobile Maintenance and Accessible in Real-Time) (système de maintenance mobile et accessible en temps réel) vise à fournir les étiquettes des nouvelles technologies intelligentes. Ce système surveille l'utilisation et les données de la maintenance à travers le cycle de vie des pièces critiques.

Par ailleurs, ce système fournit aussi de nouveaux services comme les outils avancés de dépannage qui contiennent un contrôleur de configuration, un outil de planification des ressources et un service de traçabilité afin d'améliorer celle-ci (Kusper, 2007).

En outre, ce projet vise également à établir un référentiel normatif en termes d'organisation des procédures et d'outils impliquant le MRO (Maintenance Repair & Overhaul) des intervenants, les fabricants, les opérateurs, les organismes de régulation ainsi que les compagnies d'assurance (Zephir, 2007).

L'architecture proposée dans ce projet est basée sur la technologie RFID servant à la traçabilité des unités logistiques et dans le gestionnaire du moteur de configuration qui sert à la récupération les données de configurations des composants critiques (voir Figure 2-5).

Type de maintenance :

Le projet s'intéresse à une partie du processus de maintenance et particulièrement à la gestion des unités logistiques.

Interopérabilité :

L'architecture de ce projet est bâtie sur une base de données commune créée à partir d'un référentiel normatif. Ceci assure une interopérabilité syntaxique entre les intervenants du système.

Gestion des connaissances :

Dans ce projet l'exploitation des connaissances est ignorée, sachant qu'elle vise l'intégration des services de traçabilité.

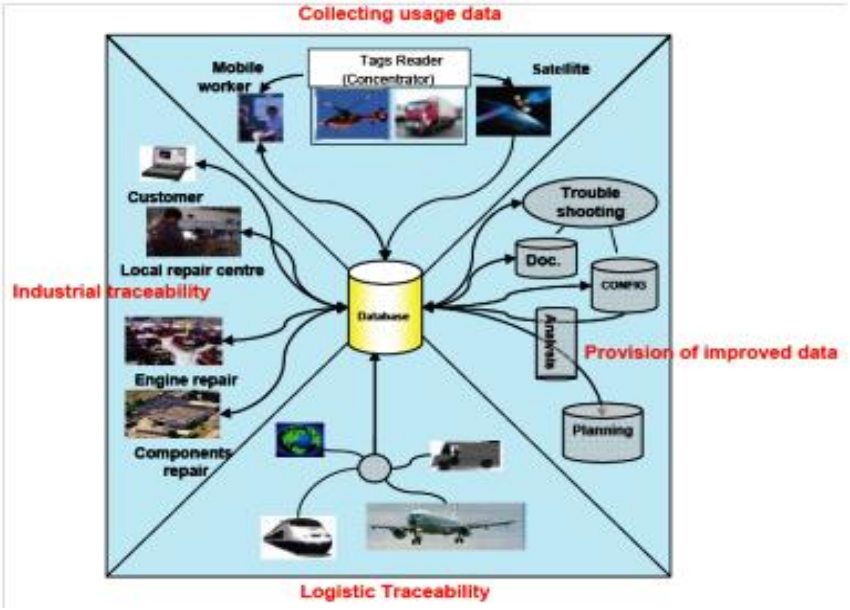


Figure 2-5 Architecture de SMMART (Zephir, 2007)

Mécanisme de synchronisation :

Les mécanismes de synchronisation entre les intervenants du système n'étant pas détaillés dans ce projet, nous ne pouvons identifier ce mécanisme à un mécanisme de coopération ou de collaboration.

2.2.6- DYNAMITE

Dans la continuité des projets Européen, le projet DYNAMITE (Dynamic Decisions in Maintenance) a eu comme objectif de créer une infrastructure pour les technologies mobiles de surveillance et de créer de nouveaux appareils/ instruments intelligents. Ces réalisations permettront des avancées majeures dans la capacité des systèmes de décision intégrant capteurs et algorithmes (Holmberg, Helle, & Halme, 2005).

Selon les objectifs déclarés par DYNAMITE, ce projet est centré sur la partie hardware et l'intégration des technologies mobiles et de nouveaux types de capteurs dans une plateforme de maintenance. Les dispositifs principaux incluent de la télémétrie sans fil, la présence d'un historique local dans des balises actives, et une instrumentation en ligne.

Dans ce contexte, une plateforme de maintenance DynaWeb a été développée (Arnaiz, Iung, Jantunen, Levrat, & Gilabert, 2007) et se réfère à une architecture basée sur les services web et sur des logiciels de communication qui prennent appui sur des fonctionnalités avancées de maintenance en lien avec le diagnostic, le pronostic et le CBM (voir Figure 2-6). C'est une plateforme d'information et de communication permettant une interaction opérationnelle entre les différents acteurs dans le cadre d'un scénario applicatif de gestion de maintenance distribuée respectant le processus OSA-CBM (Figure 2-7).

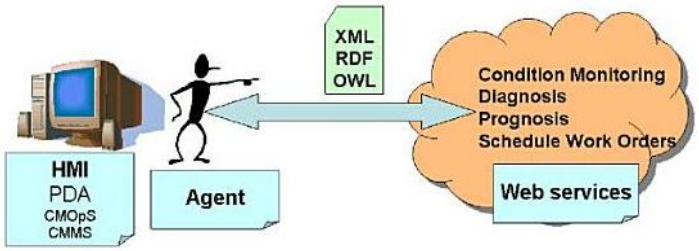


Figure 2-6 DynaWeb : une plateforme basée sur les services Web (Arnaiz, Iung, Jantunen, Levrat, & Gilabert, 2007)

L'architecture logicielle de DynaWeb est basée sur des composants logiciels offrant des services distribués sur le Web, ce qui est un point central.L'information est traité dans un système distribué et de collaboration, où il ya différents niveaux d'entités qui peuvent accomplir des tâches intelligentes. Compte tenu de cela, une architecture de système a été définie en prenant appui sur la norme OSA CBM pour identifier les interactions entre les acteurs et les fonctions requises, notamment la surveillance des états, l'évaluation de la santé, le pronostic, et l'aide à la décision (Lebold & Thurston, 2001).

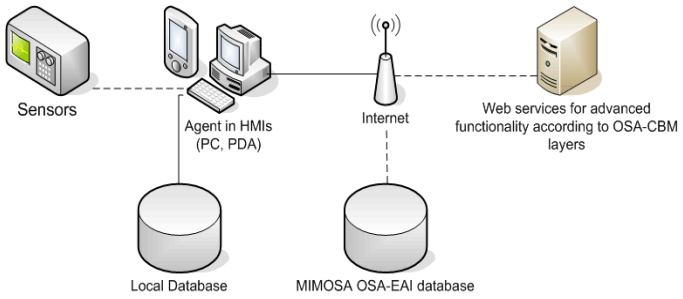


Figure 2-7 Architecture web de DynaWeb (Arnaiz, Iung, Jantunen, Levrat, & Gilabert, 2007)

Une plateforme dédiée à l'enseignement appelée TELMA a été définie comme plateforme de formation (et d'expérimentation) dans les domaines de la maintenance, télémaintenance et e-maintenance (Levrat & Iung, 2007).

Selon Iung, cette plateforme est en cohérence avec la philosophie de e-maintenance globale qui maintient un processus physique connectés à la fois à l'architecture d'automatisation et à l'architecture de maintenance en intégrant :

- L'ingénierie et le déploiement de CBM et les stratégies de maintenance proactives compatibles avec la proposition d'OSA/CBM (Lebold & Thurston, 2001);
- L'évaluation de ces stratégies sur la prdoductivité du système industriel global (la disponibilité, l'exploitation), la Qualité, le Prix …°

La plateforme TELMA (voir la Figure 8) est une plateforme dite « *infotronic* » assurant des services en ligne et hors ligne soutenant le traitement, le stockage et la communication entre les trois niveaux de données, d'information et de connaissances. Elle contient des agents intelligents assurant les services en ligne. Les services hors ligne sont généralement les services d'aide à la décision, qui contrairement aux nouveaux besoins ne sont pas des services à la demande et ne sont pas dynamiques.

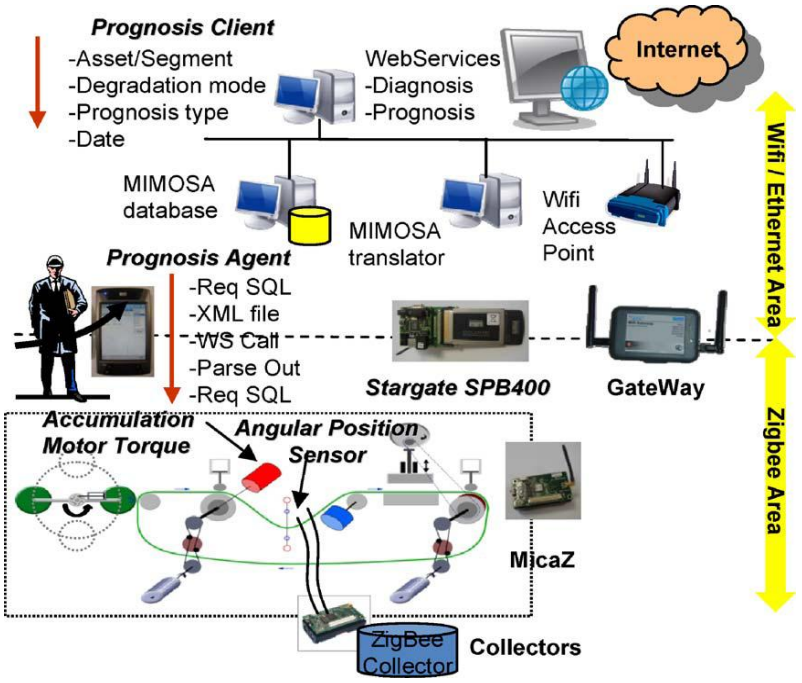


Figure 2-8 Architecture de TELMA (Levrat & Iung, 2007)

Interopérabilité :

L'architecture logicielle offre une interopérabilité entre les applications logicielles indépendantes sur Internet grâce au protocole SOAP[19] assurant la communication entre ces applications. L'échange de données information étant un des points majeurs de DYNAMITE, ce projet a traité le point d'interopérabilité syntaxique via les services web, une interopérabilité technique sans se préoccuper du niveau sémantique. DYNAMITE a adopté le modèle CRIS MIMOSA comme base de données commune et partagée et non pas une ontologie commune.

Type de maintenance :

D'autre part, le projet s'est focalisé sur le processus OSA-CBM, notamment la maintenance proactive mais n'ont pas développés les autres stratégies de maintenance, ce qui limite la gestion de la maintenance dans la plateforme.

Gestion des connaissances :

L'architecture de la plateforme contient une base de connaissances, et des agents intelligents de TELMA assurant le passage de niveau entre données et connaissances, ainsi que des services hors ligne d'aide à la décision. La base de connaissance est développée pour alimenter le niveau « health assesment » et « prognostics » du processus OSA-CBM dans la plateforme, ce qui permet de définir des services intelligents. Toutefois, la structure de cette plateforme ne permet pas d'être auto-apprenante et de proposer des services dynamiques à la demande.

Mécanismes de synchronisation :

Les différents modules logiciels de DynaWeb communiquent entre eux afin d'accomplir une tâche spécifique tout en partageant la base de données MIMOSA. La plateforme utilise donc un mécanisme de collaboration.

[19] ***Simple Object Access Protocol : http://www.w3.org/TR/soap/***

2.2.7- eMM

Le projet intié dans le cadre d'un programme de recherche (NFFP4), et supporté par *« Saab Aerotech »*, l'Agence Nationale Suédoise de l'aéronautique et l'Agence suédoise de la Force aérienne *« Wing F21 »*, un environnement logiciel de travailpour la gestion de la e-maintenance (*eMaintenance Management Framework [eMM]*)) basée sur une architecture orientée service a été proposée (Candell, Karim, & So derholm, 2009). Cet environnement logiciel comportera:

1. un modèle de gestion de eMaintenance [*eMaintenance Management Model (eMMM)*]
2. une plateforme de e-maintenance [*eMaintenance Platform (eMP)*.] (voir Figure 2-9).

L'eMMM est l'ensemble de rôles, de processus et d'entrepot de données nécessaires pour la gestion de la plateforme. Cette dernière a une Architecture Orientée Service(AOS) (Bell, 2008) visant à fournir à ses utilisateurs des informations personnalisées pour la prise de décisions Le but de cette plateforme est d'extraire toutes les informations nécessaires àpour certaine tâche de maintenance, dans l'objectif d'analyser, de synthétiser ainsi que de les compacter dans un processus de maintenance dédié à l'intervention.

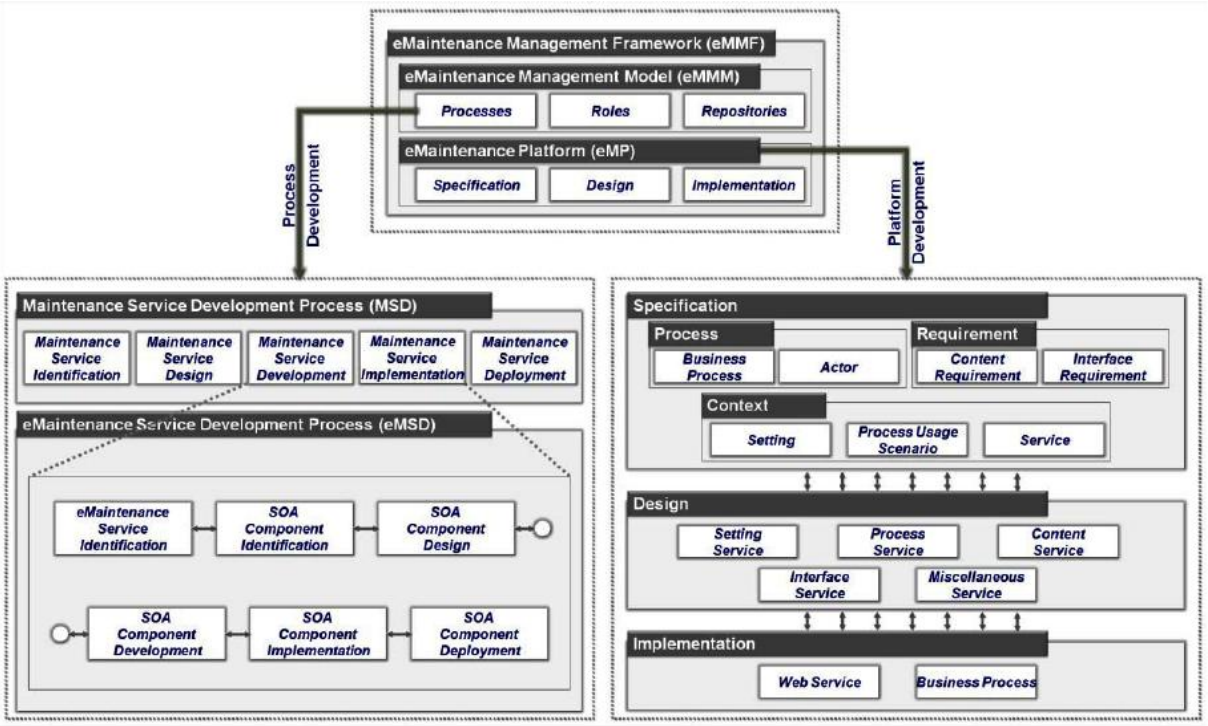


Figure 2-9 Architecture de eMMF (Karim, Candell, & Söderholm, 2009)

L'architecture eMP est divisée en trois niveaux: la spécification, la conception et l'implémentation (voir Figure 2-9). Le niveau de spécification contient toutes les informations nécessaires à la réalisation de la maintenance. Il est réalisé de différentes façons en utilisant tout type de données (notations, modèles, etc.). Le niveau de conception identifie les composants de conception. Il est réalisé en tant que composants AOS, en transformant la plupart des composants de spécification en composants AOS. Enfin, le niveau de mise en œuvre matérialise les composants AOS dans des services Web et dans les processus métiers orchestrés pour les besoins. Dans le cadre

de l'ePM, tout technicien de maintenance doit fournir des entrées utiles à l'application. L'application délivre alors un processus métier pertinents adapté à ses besoins.

Ceci montre qu'il n'y a pas de connaissances communes partagées entre ces techniciens et chacun d'eux adapte la plateforme à ses besoins spécifiques ce qui élimine toute possibilité de réutilisation des connaissances ou même d'exploitation, sachant que la sémantique des concepts et des visions peut changer d'un technicien à un autre.

Interopérabilité :

Il est à noter que l'AOS est une excellente approche pour faciliter la maintenance industrielle. Elle met l'accent sur la conception d'un point de vue processus métier, et est l'une des principales conditions préalables pour parvenir à une approche flexible, extensible et rentable pour la mise en place de la plateforme eMP.

Toutefois cette approche d'intégration, est similaire aux plateformes des projets PROTEUS et DYNAMITE basées sur les services Web qui est une dérivée de l'AOS. L'approche AOS utilisée dans l'eMP assure par conséquent une interopérabilité syntaxique entre les différents services déployés par la plateforme.

Gestion des connaissances :

Nous constatons que l'ePM est applicable par toute organisation assurant la maintenance industrielle, car elle est basée sur un processus de maintenance industrielle générique et complet. Par contre, la gestion et l'exploitation des connaissances ne sont pas mentionnées dans ce projet. ,

Type de maintenance :

L'objectif de cette plateforme est de faciliter les tâches d'interventions par les techniciens, et ne s'intéresse pas à l'exploitation des services par les autres acteurs impliqués dans le processus de maintenance.

Mécanismes de synchronisation :

Finalement, l'objectif sous jacent à l'architecture orientée service dans eMM estl'intégration des services dans un seul système permettant l'accès à l'information de façon unifiée. Toutefois, l'aspect synchronisation entre ces services n'a pas été détaillé.

2.3- Plateformes académiques

Dans cette section nous ferons l'étude de six plateformes académiques instanciées dans le cadre universitaire ou de centre de recherche. La plupart de ces plateformes restent dans une sphère de proposition théorique sans réalisation concrète à l'exception de IMS-WATCHDOG dans D2B^{TM}. Les plateformes que nous étudierons sont : WSDF, D2BTM, Plateforme Web, Plateforme Orientée Agents, EMAST et IIMED.

2.3.1- WSDF

Hung et al en (Hung, Chen, Ho, & Cheng, 2003) ont proposés *un cadre de e-diagnostics basé sur des services Web* (*WSDF*) visant à améliorer les systèmes de diagnostic à distance en fournissant automatiquement le mécanisme d'intégration des informations de diagnostic par l'Internet. La plateforme WSDF a comme objectif

soutenir les tâches de e-diagnostics dans les usines de semi-condicteurs. Les technologies utilisées dans cette plateforme (voir Figure 2-10) sont les Services Web, XML Signature, chiffrement XML, https, SOAP, ISDN… .

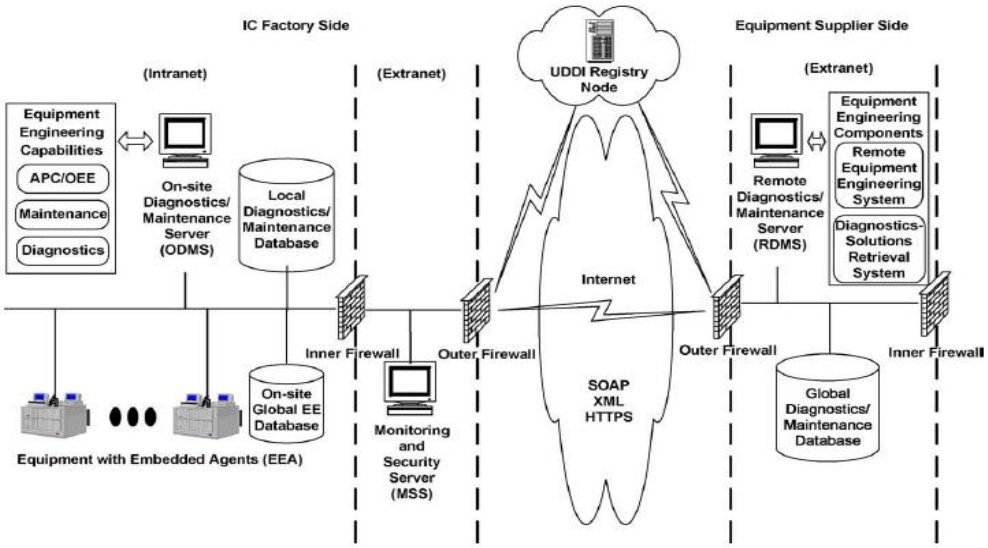


Figure 2-10 Architecture WSDF (Hung, Chen, Ho, & Cheng, 2003)

Interopérabilité :

L'architecture technologique basée sur les services web de cette plateforme, grâce aux échanges XML et aux services Web prend en compte l'interopérabilité syntaxique mais pas son niveau sémantique.

Gestion des connaissances :

D'autres parts, l'architecture proposée ne se base et n'exploite dans aucune partie ou composants l'ingénierie des connaissances.

Type de maintenance :

Cette plateforme traite essentiellement la partie diagnostic du processus de maintenance, la communication et la sécurité.

Mécanismes de synchronisation :

La synchronisation entre les acteurs de cette plateforme, n'étant pas détaillée nous ne pouvons pas identifier si ces mécanismes sont orientés collaboration,…

2.3.2- D2B™

L'un des premiers projets académiques en e-maintenance a été élaboré au centre de systèmes intelligents de maintenance (IMS) aux Etats-Unis. Ce projet supporte le déploiement et l'expérimentation de la plateforme « dispositif-to-business » (D2B™) prenant appui sur un élément de base qui est le Watchdog Agent™ (Lee & Ni, 2004). L'objectif de cette plate-forme est de transformer les données équipement à des données au format compatible Web (par exemple XML) de sorte que de nombreuses applications Web peuvent être exécutées (Ali, Chen, Lee, & Koc, 2002). Via ce dispositif d'information les utilisateurs de différentes parties du réseau de

collaboration distribuée géographiquement peuvent partager la même information pour différentes applications synchronisées (Koc, Ni, Lee, & P, 2003).

Le « Watchdog agent » développe des outils et des algorithmes de pronostics innovants, des technologies de maintenance prédictives locales et à distance pour prédire et prévenir les défaillances des machines (voir **Erreur ! Source du renvoi introuvable.**).

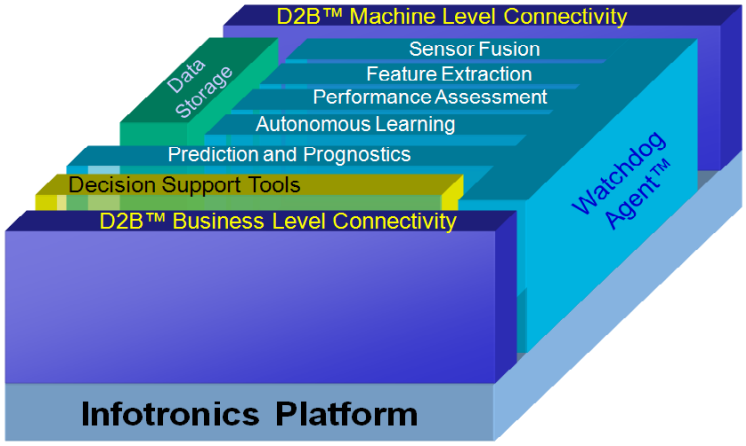


Figure 2-11Integrated *infotronics* platform (Lee & Ni, 2004)

Type de maintenance :

Dans la continuité de ce projet, une plateforme d'*infotronics* intégrant différents composants intelligents a été proposée par IMS center. Cette plateforme ne s'intéresse qu'à la maintenance prédictive, et plus précisément au pronostic qui est la stratégie de maintenance la plus évoluée, mais néglige les autres stratégies. Par contre cette plateforme assure quelques services avant-gardistes comme le *self-learning*.

Interopérabilité :

L'interopérabilité est assuré au niveau syntaxique par un échange standardisé sous un format XML, et ne traite pas le problème d'interopérabilité sémantique.

Gestion des connaissances :

En ce qui concerne la gestion des connaissances, la plateforme contient des agents de connaissance permettant d'extraire des connaissances à partir de la base de données ainsi que le acquérir à partir de connaissances pour aider à la décision.

Mécanismes de synchronisation :

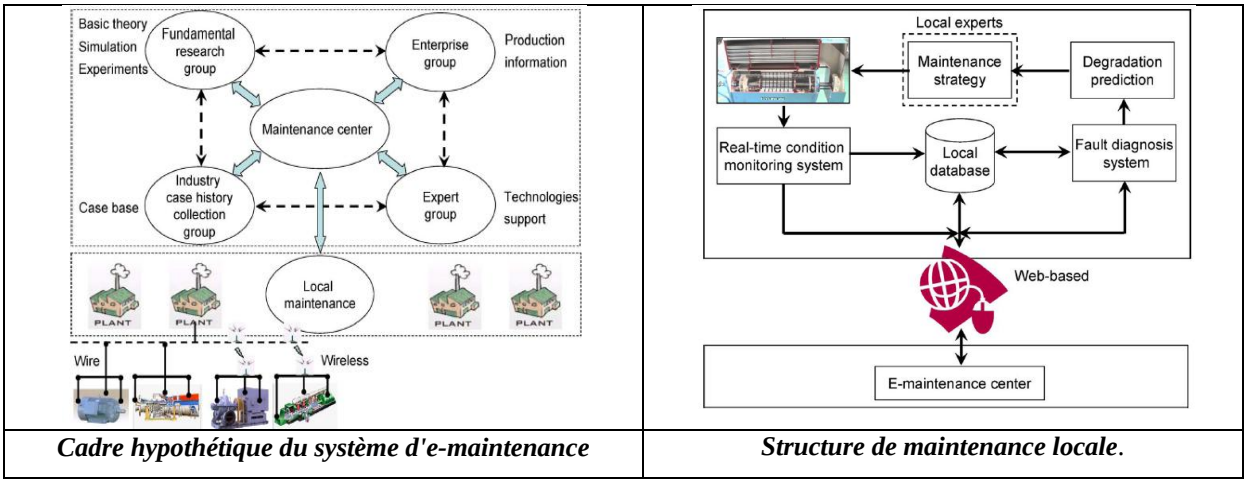
Par contre, les notions de synchronisation et d'intégration ne sont pas décrites dans ce travail, mais d'après la description de l'architecture du système, on peut déduire l'existence d'un processus de coopération entre ces différents modules.

2.3.3- Plateforme web

Han et al (Han & Yang, 2006) ont proposé un nouveau système d'e-maintenance dépendant de la coordination, la coopération et de la négociation en utilisant internet, les technologies de la communication *tether-free* (web,

télécommunication sans fil). Ce système permet de réaliser des opérations de production pour atteindre la performance zéro temps d'arrêt sur une plateforme distribuée grâce à l'intégration des technologies avancées de réseaux. La structure du système de e-maintenance, contrairement aux systèmes classiques, comprend un centre de maintenance (fournisseurs) et une maintenance locale (clients) ce qui réduit efficacement les coûts de maintenance, le temps de conception de cette maintenance ce qui résout le problème du manque d'experts.

Le centre de e-maintenance contient en fonction de l'application un système de maintenance de base incluant la surveillance temps réel, le diagnostic de panne, la prédiction de dégradation et la gestion de la stratégie de maintenance. D'autre part, la maintenance locale se compose du service maintenance interne à l'entreprise ayant le rôle d'évaluation continue de la santé des équipements, de la gestion du processus d'intervention, de réparation et d'entretien ainsi que la compréhension et présentation des données (voir Figure 2-12).



Cadre hypothétique du système d'e-maintenance | ***Structure de maintenance locale.***

Figure 2-12 Architecture d'une plateforme web (Han & Yang, 2006)

Type de maintenance :

L'avantage du processus de maintenance est que les experts peuvent définir la stratégie ainsi que le processus de maintenance qu'ils jugent le mieux adapté pour chaque situation, le système ne s'autogère pas.

Interopérabilité :

Les échanges d'information effectués sur le web entre le centre d'e-maintenance et le centre de maintenance locale sont basés sur des messages XML standardisés. Ceci n'assure qu'une interopérabilité syntaxique pour ces échanges.

Gestion des connaissances :

L'ingénierie des connaissances n'est pas utilisée dans cette plateforme. Ceci est dû au faite que toute constation et toute opération (présentation, compréhension et analyse) sur les données sont faites par les experts (humain) de maintenance locale.

Mécanismes de synchronisation :

Le partage des connaissances entre les acteurs sur site local et à distance confirme la collaboration entre ces acteurs à des fins de maintenance.

2.3.4- Plateforme orientée Agents

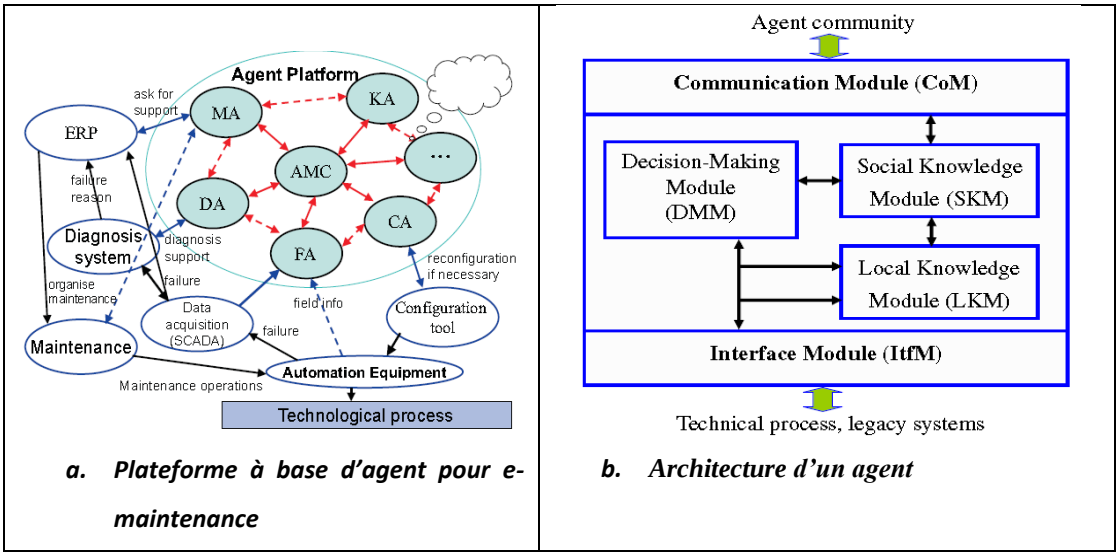
Zhang a proposé un système multi agents (Wooldridge, 2002) de e-maintenance basé sur la gestion des connaissances (Zhang, 2007). Ce système multi-agents permet d'échanger des connaissances/informations entre des systèmes d'automatisation industrielle. La figure 2-13.a montre l'infrastructure à base d'agent dédiée à la gestion de connaissances dans ce système. Toutes les sources d'information dans les systèmes d'automation industriels sont analysées et modélisées par des agents. Les relations entre ces sources sont mises en évidence dans le contexte de la maintenance, notamment la maintenance proactive[20].

L'architecture de la plateforme est composée principalement d'un AMC (Agent Management Core) et de cinq types d'agents à savoir l'agent de connaissance [Knowledge Agent (KA)], l'agent de configuration [CA (Configuration Agent)], l'agent de diagnostic [DA (Diagnosis Agent)], l'agent de récupérations des données [FA (Field Agent)] et l'agent de management [MA (Management Agent)].

Ainsi, comme le montre la figure 2-13.b chaque agent est composé de cinq modules à savoir le module de connaissances local (LKM) qui définit les méta-connaissances relatives à la connaissance du système d'automatisation qui est représentée par l'agent; le module de connaissance sociale (SKM) qui définit les associations communautaires de l'agent; le module de prise de décision DMM qui définit les algorithmes et les modes de prise de décision proactive, par exemple, la logique floue, réseaux neuronaux artificiels, ou de raisonnement à base de cas ; le module de communication (CoM) définit les mécanismes de communication et des protocoles; et enfin le module d'interface (ItfM) qui définit les spécifications pour les interfaces et les adaptateurs.

Type de maintenance :

Cette orientation connaissance adoptée par Zhang, argumentée et orientée vers la maintenance proactive. Selon lui, la maintenance proactive peut être définit comme l'association entre la maintenance prévisionnelle avec la gestion de la connaissance. Ceci permettra d'intégrer de la dualité entre le fonctionnement et les dysfonctionnements des systèmes à maintenir qui est en soit même l'un des services requis par la nouvelle génération des systèmes de maintenance.



a. Plateforme à base d'agent pour e-maintenance

b. Architecture d'un agent

Figure 2-13 Architecture Plateforme orientée agents (Zhang, 2007)

[20] Maintenance proactive : optimisation des conditions d'utilisation dans le but d'augmenter la durée de vie des équipements.

Interopérabilité :

Le module de connaissance sociale de chaque agent permet d'assurer une interopérabilité sémantique entre ces agents en adaptant les connaissances locales avec celles de la communauté, sans pour autant partager toute la connaissance experte de maintenance et communique avec les autres agents via le module de communication qui échange avec le module de connaissance.

Gestion des connaissances :

Cette plateforme prend appui sur l'ingénierie des connaissances grâce à l'agent de connaissance qu'elle intègre ainsi que les modules internes de chaque agent qui sont orientés connaissances, mais ne propose pas de fonctionnalités auto-X, ni des services dynamiques ni à la demande et autre.

Mécanismes de synchronisation :

Selon cette architecture, malgré le partage des connaissances de l'environnement, chaque agent a ses propres ressources qu'il ne partage pas avec les autres agents et limite la communication à la coopération et non pas à la collaboration.

2.3.5- EMAST (E-Maintenance Architecture to Support on-site Teams)

Ribiero et al proposent une architecture de e-maintenance pour supporter les équipes de maintenance sur le site de production (Ribeiro, Barata, & Silvério, 2008).

L'architecture EMAST propose une solution verticale allant du diagnostic et/ou pronostic de bas niveau à la création de séances de maintenance collaborative. Chaque module dans cette architecture effectue du *self-monitoring/diagnosis* (auto-surveillance et autodiagnostic) d'où sa capacité à utiliser ces informations pour émettre des alarmes de maintenance prédictive, en plus des alarmes basées sur le temps de la maintenance préventive. Ainsi l'ensemble de documentation (manuels techniques, plans, procédures de réparation / maintenance, etc) est stocké localement sur le site.

Le module de configuration de l'équipe de maintenance « *The Maintenance Team Configurator (MTC)* » est l'un des composants de base de l'architecture. Il gère le groupe de techniciens de maintenance disponibles, leurs compétences et l'instanciation des taches de maintenance et/ou de réparation (Figure 2-14).

Toutes les opérations de maintenance sont stockées dans une base historique et caractérisées par des variables qui sont : la nature de l'opération (critiques, non programmée ...), le taux de réussite de l'opération (mesure de la performance des techniciens et les orientations fournies par la plateforme) et les techniciens impliqués (chaque technicien dispose d'un score qui lui accorde du crédit pour effectuer des opérations plus complexes).

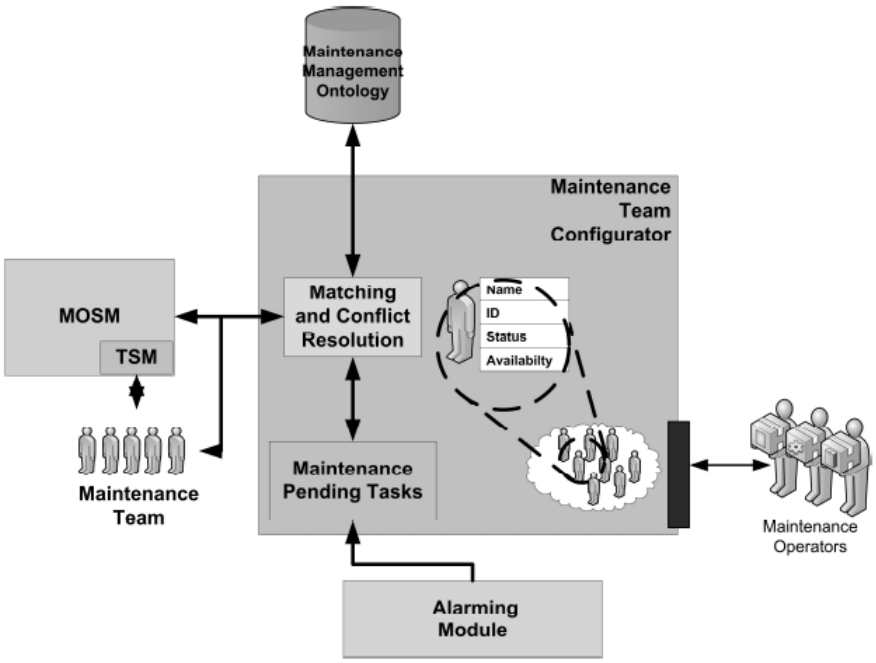


Figure 2-14 Architecture EMAST (Ribeiro, Barata, & Silvério, 2008)

Quand une alarme de réparation est reçue, le MTC propose une équipe de maintenance en se basant sur l'historique de maintenance des alarmes antécédentes. Le MTC vérifie ces informations par rapport à la connaissance dans le gestionnaire d'ontologie de Maintenance « Maintenance Management Ontology (MMO) » et la disponibilité des techniciens. Les conflits de correspondance lors de cette vérification sont résolus par les règles du MMO où les règles de composition de l'équipe sont mappées.

Le processus de formation d'une équipe de maintenance comprend (1) la définition du critère d'urgence par taches de maintenance, (2) l'attribution des rôles aux participants, (3) l'information sur la nature et l'emplacement de la tâche, (4) l'affectation du contrôle au module de support de maintenance opérationnelle *« Maintenance Operational Support Module (MOSM) »*. Grâce à une interface de soutien technique (TSM), le MOSM guide les équipes de maintenance en cours des opérations.

Le MTC gère, planifie et ordonnance la totalité des actions de maintenance et des équipes mais il peut aussi éventuellement différer l'exécution de la maintenance en reconnaissant que la meilleure équipe sera disponible dans un intervalle de temps raisonnable.

Interopérabilité :

Aucune information ne mentionne le caractère centralisé ou distribué de l'architecture de EMAST et sa possibilité à intégrer des modules externes. On ne peut pas savoir comment est traité le problème d'interopérabilité.

Type de maintenance :

D'autre part, le travail traite essentiellement le problème d'affectation des équipes de maintenance par rapport aux alarmes reçus, la planification des taches de maintenance et le stockage de l'historique des interventions.

Gestion des connaissances :

Bien que cette architecture mentionne la traçabilité et l'exploitation des opérations de maintenance précédemment faites, il n'y a aucune explication sur l'exploitation des connaissances et sur leur formalisation et l'objectif de celle-ci.

Mécanismes de synchronisation :

L'une des caractéristiques majeures de cette plateforme est la collaboration entre les différents acteurs par le partage des ressources qui est la base de données historique ainsi que l'ensemble des documentations.

2.3.6- IIEMD

Cao et al (Cao & Jiang, 2008) développent une plate-forme de maintenance intelligente coopérative et distribuée IIEMD (*Integrated Intelligent Equipment Maintenance Decision*) qui propose un ensemble d'outils d'aide à la décision pour différentes activités de maintenance. IIEMD est un système hybride, soutenu par une ontologie où deux types de raisonnement cohabitent le RBR (*Rules Based Reasoning*) et le CBR (*Case Base Reasoning*).

L'architecture de la plateforme IIEMD comme le montre la Figure 15 est une AOS (Architecture Orientée de Service [*SOA* en anglais]). Cette architecture basée sur la technologie Web, est modulaire, offrant la possibilité d'une intégration pervasive pour plus d'agilité en temps réel. L'architecture d'implémentation de cette AOS est une architecture à base de composants.

Cette plateforme peut réaliser la présentation et la synthèse des données d'une manière compréhensive, l'évaluation continue de la santé des équipements, la prise de décisions sur les opérations de maintenance et de réparation, ainsi que l'intégration d'autres systèmes.

Dans cette architecture, les outils de décision de maintenance (tous sont définis comme des services Web) peuvent enregistrer, récupérer l'information à partir d'autres systèmes, distribuer les informations aux autres systèmes intégrés dans la plateforme. Ainsi, elle fournit un processus de décision automatisé, transparent, continu de façon unifiée, indépendamment de son origine, du fabricant, des intégrateurs et des utilisateurs finaux de l'équipement grâce des méthodes de raisonnements à base de cas et à base de règles associées à des règles d'adaptation.

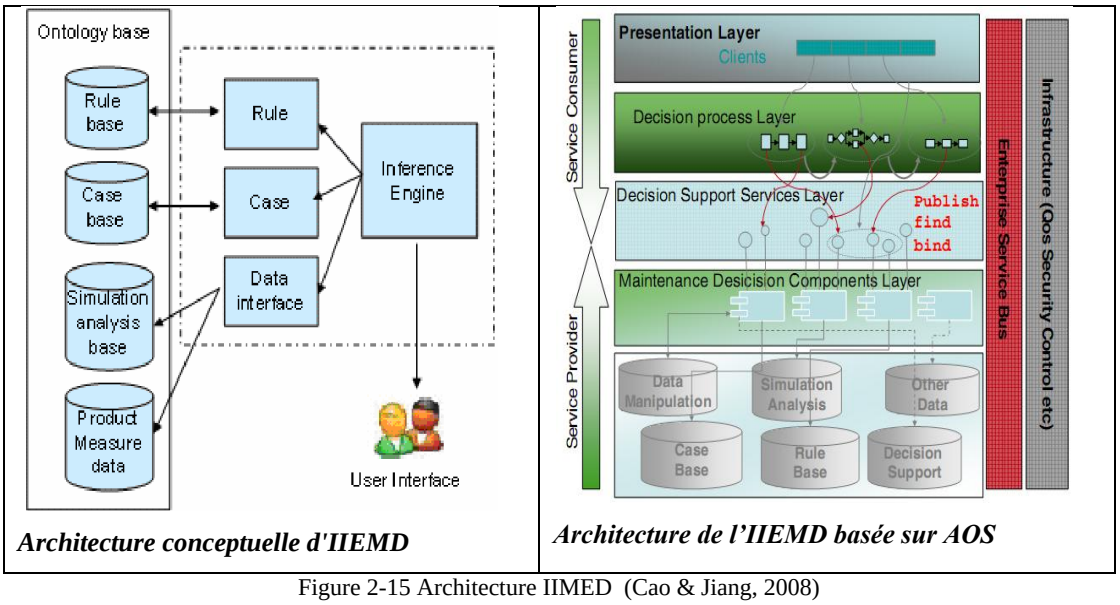


Figure 2-15 Architecture IIMED (Cao & Jiang, 2008)

Cette plateforme comprend une base de connaissances prenant appui sur une ontologie. Cette dernière est formalisée à partir du modèle standard CRIS défini par MIMOSA (Lebold & Thurston, 2001) présentée sous forme de base de règle OWL[21] ainsi qu'un moteur d'inférence.

Le moteur d'inférence traite les requêtes des utilisateurs grâce au module de règle, le module de cas, le module d'interface de données, la base de connaissance contenant l'ontologie, le modèle des données, la base de l'analyse par simulation et les données de mesure.

Interopérabilité :

Bien qu'il ne soit pas mentionné, l'ontologie a été utilisée au profit de l'interopérabilité sémantique, car les services web utilisés dans l'AOS manipulent les concepts de l'ontologie.

Types de maintenance :

La plateforme fournit les services d'aide à la décision pour l'intervention et la réparation dans le processus de maintenance. Toutefois, la plateforme ne propose pas les services concernant toutes activités inclus dans le processus de maintenance (comme le diagnostic, le pronostic, l'optimisation des ressources, la planification) bien qu'elle peut les intégrer grâce à son AOS.

Gestion des connaissances :

La plateforme est un système hybride prenant appui sur l'ingénierie des connaissances par l'exploitation d'un moteur d'inférence qui exécute différents algorithmes de raisonnement (raisonnement à base de règles et raisonnement à base de cas) appliqué sur une base de connaissances contenant une ontologie faisant référence MIMOSA-CRIS.

Mécanismes de synchronisation :

[21] Web Ontology Language: http://www.w3.org/TR/owl-ref/

IIMED est une plateforme de coopération, en effet, ceci est confirmé grâce à l'architecture AOS qui assure un échange de ressources entre ses différents outils intégrés. De plus, l'abscence de ressources partagées lui confère le statut de coopération.

2.4- Tableau de synthèse

Nous résumons dans le tableau 1 les caractéristiques des plateformes présentées dans ce chapitre selon les critères utilisés dans l'analyse de ces plateformes à savoir le processus de maintenance, l'interopérabilité, l'exploitation des connaissances, les services innovants fournis par ces plateformes, le degré de synchronisation et les objectifs des projets dans les quels les plateformes étaient construites.

Tableau 2-1 Tableau récapitulatif des plateformes de maintenance

Plateforme	**Processus de maintenance**	**Interopérabilité**	**Exploitation des connaissances**	**Services autonomes**	**Degré de synchronisation**	**Objectifs**
MIMOSA	CBM	Syntaxique (MIMOSA-CRIS)	--	--	Coopération	Intégration
WSDF	Diagnostic	Syntaxique (fichier XML)	--	--	--	Communication, sécurité
PROTEUS	Tout le processus de maintenance	Syntaxique (services web)	Mémoire d'entreprise pour le diagnostic.	--	Coopération	Intégration, interopérabilité
SMMART	Cycle de vie des pièces critiques	Base de données commune, Référentiel normatif.	--	--	--	Maintenance Mobile
PROMISE	BOL (début de vie),MOL (Milieu de vie), EOL(fin de vie)	SOM (Modèle sémantique d'objets)	--	Chien de garde pour le pronostic	Collaboration	Gestion du cycle de vie
DYNAMITE	OSA/CBM	Syntaxique (services web)	--	--	Collaboration	Création d'une infrastructure pour les technologies mobiles de surveillance
TELMA	OSA/CBM	Syntaxique (services web)	Base de connaissances pour le pronostic.	--	Collaboration	Enseignement, Télémaintenance.
D2B^{TM} / IMS WATCHDOG	Maintenance prédictive	Syntaxique (fichier XML)	Agent de connaissances pour l'aide à la décision.	Self-learning	Coopération	Prédire et prévenir les défaillances des machines, Assurer un temps d'arrêt près de

						zéro.
eMM	Aide à l'intervention	Syntaxique (services web)	--	--	--	Intégration, de faciliter les taches d'interventions
Plateforme Web	Tout le processus de maintenance	Syntaxique (messages XML)	--	--	Collaboration	Identification de la stratégie de maintenance adaptée pour chaque situation, assurer un temps d'arrêt près de zéro.
Plateforme Orientée Agents	Maintenance proactive	Sémantique (module de connaissances des agents)	Gestion et échange de connaissances	--	Coopération	Augmenter la durée de vie des équipements.
POMAESS	Diagnostic	--	Base de cas pour le diagnostique	--	Coopération	Aide à la décision pour le diagnostique.
EMAST	Maintenance prédictive et maintenance préventive	--	Maintenance Management Ontology (MMO) »	--	Collaboration	Supporter les équipes de maintenance sur le site de production.
IIEMD	Aide à la décision pour la réparation	Sémantique (Ontology OWL)	Ontologie, raisonnement et infèrence : RBR (Rules Based Reasoning) et le CBR (Case BaseReasoning)	--	Coopération	Service d'aide à la décision de maintenance des équipements.

D'après cette analyse, nous avons constaté que PROTEUS est l'un des rares projets qui présentait une vision complète sur le processus de maintenance. Par contre, en ce qui concerne l'exploitation de la connaissance, malgré l'intégration de l'ingénierie des connaissances dans l'application de diagnostic. Cette connaissance n'était pas exploitée au cœur de la plateforme, (c.à.d. la plateforme n'est pas un système orienté connaissances) comme défini pour la s-maintenance.

La plateforme PROMISE est basée sur un modèle de données dit sémantique. En effet, l'objectif de ce modèle est de fournir une compréhension commune aux utilisateurs finaux de cette plateforme, n'est exploité par les

applications et services intégrés dans la plateforme, et par conséquent ne fournit pas de raisonnement (point fondamentale requis dans une plateforme de s-maintenance).

La plateforme TELMA fournit un ensemble de services d'aide à la décision en hors ligne, service en inadéquation avec les services dynamique en ligne et/ou services à la demande.

La plateforme D2B^{TM} assure des services intelligents de *self-learning* dédié à l'application de pronostic, ce qui impacte la robustesse de la maintenance prévisionnelle et non pas sur l'auto-apprentissage de la plateforme et on retour d'expérience sur toutes les applications de maintenance. L'utilisateur ne pourra bénéficier de ce retour d'expérience dans ses services à la demande.

La Plateforme Orientée Agents possède quelques points de concordance avec la s-maintenance au niveau de l'interopérabilité sémantique et l'exploitation de l'ingénierie des connaissances. Néanmoins, cette exploitation de connaissances est faite localement par un agent et n'est partagées et réutilisées par toutes les applications d'aide à la maintenance intégrées dans la plateforme. De plus, les agents ne fournissent pas de services dynamiques.

La plateforme EMAST est basée sur une ontologie de domaine et exploite l'ingénierie des connaissances. Mais l'architecture de cette plateforme ne fournit aucuns détails ni sur l'utilisation de cette ontologie ni sur son exploitation par rapport à l'interopérabilité sémantique et par rapport aux services fournis aux utilisateurs.

La plateforme IIEMD est l'une des plus proches de la s-maintenance. Elle se basant sur l'ingénierie des connaissances (ontologie et moteur d'inférence) et en met l'accent sur l'aspect coopératif. Toutefois elle ne s'intéresse pas aux services dynamiques et autonomes. De plus, IIEMD ne traite que les services d'aides à la décision pour la réparation qui est une approche orientée utilisateur final (le technicien), mais elle ne développe pas sur la notion de génération de services à la demande pour les différents types d'utilisateurs de la plateforme (tous les acteurs de la maintenance).

De manière globale, le tableau de synthèse (tableau 2.1) nous permet de constater qu'il n y a que trois plateformes sur quatorze (la plateforme de PROMISE, la plateforme Orientée Agents et la plateforme IIEMD) qui traitent de la problématique de l'interopérabilité sémantique, alors que les autres se limitent au niveau syntaxique de l'interopérabilité.

En ce qui concerne le processus complet de maintenance il n'y a que la plateforme PROTEUS et la Plateforme Web qui se sont intéressées à cela. Les plateformes eMM et IIEMD ont plus ou moins traité un ensemble d'activités de ce processus sans se focaliser sur une seule activité ou sur une seule stratégie de maintenance comme dans la plupart des autres plateformes (par exemple : MIMOSA, DYNAMITE, WSDF).

A propos de l'exploitation des connaissances, nous constatons que différentes plateformes tirent profits de l'ingénierie des connaissances et ontologique comme dans le cas de EMAST, PROTEUS et IIEMD. Par contre, cette exploitation est restreinte dans la plus part des cas à une partie de la maintenance, concernant le pronostic dans le cas de DEBTM ou l'aide à la réparation dans IIEMD, ou à une exploitation réduite par un nombre limité d'applications comme dans le cas de PROTEUS où la base de connaissance est un module extérieur au noyau de la plateforme.

Nous constatons aussi que la majorité de ces plateformes sont partagées entre la coopération et la collaboration. Les plateformes MIMOSA, PROTEUS, D2BTM, Plateforme Orientée Agents, POMAESS sont classées comme des plateformes de coopération. Par ailleurs, les plateformes classées comme plateformes de collaboration sont PROMISE, DYNAMITE, TELMA, Plateforme WEB, EMAST, IIEMD.

Synthèse :

A l'issue de cette étude nous constatons qu'il y a une différence au niveau des caractéristiques et des objectifs de chacune des plateformes. Par ailleurs, la majorité de ces plateformes n'est pas en concordance avec les critères de la s-maintenance bien que quelques unes de ces caractéristiques soient prises en compte de manière partielle dans certaines plateformes.

Nous pouvons conclure aussi que les plateformes IIEMD et Plateforme Orientée Agents (POA) sont plus ou moins les plus proches de la plateforme de s-maintenance envisagée par rapport au partage et à l'exploitation des connaissances. Dans IIEMD la connaissance est gérée dans le cadre d'un système à base de connaissance contenant une base ontologique et un moteur d'inférence utilisé par des services d'aides à la décision dans la plateforme. En ce qui concerne POA, les connaissances sont partagées par des agents et chaque agent contient son propre module de décision qui manipule ces connaissances.

Dans une optique de réutilisation nous allons étudier les points clés de ces deux plateformes. Cette réutilisation pourra poser des problèmes sachant que les architectures des plateformes sont totalement différentes (SOA, multi agent).

En ce qui concerne la Plateforme Orientée Agents, son architecture se base sur la modélisation d'agent à partir des systèmes d'automation industriels existants (ERP, SCADA, etc). Le seul agent autonome dans cette plateforme qui ne fait pas référence à un système existant est l'agent de connaissances (KA-Knowledge Agent). Il est à noter que le type des agents intégrés dans POA n'est pas mentionné, nous ne savons pas si les agents sont proactifs, réactifs, cognitifs ou conversationnels animés. Cette plateforme nécessite ainsi l'intégration de nouveaux agents autonomes fournissant les fonctionnalités et les caractéristiques requises par la plateforme de s-maintenance comme des agents d'auto-apprentissage, d'autogestion, d'auto-traçabilité, etc.

L'architecture de IIEMD intégre des services concernant des activités d'aide à la réparation et l'intervention. Dans son état, cette architecture permet l'intégration de nouveaux services. Nous pouvons donc l'adopter et la faire évoluer pour qu'elle couvre le processus global de la maintenance et fournir les services requis par la s-maintenance. Ainsi, d'autres services spécifiques doivent être intégrés afin de fournir les fonctionnalités requises par la s-maintenance comme le self-X et les services à la demande.

Par ailleurs, il est à noter aussi que les deux plateformes sont conceptuelles et restent dans un cadre théorique. Les grandes lignes sont données et les détails de leurs fonctionnements sont omis, ce qui met en cause la reprise de l'une d'elles et renforce l'idée de proposer une nouvelle architecture. Toutefois, ceci n'élimine pas les possibilités d'inspiration concernant ces deux plateformes pour proposer une nouvelle architecture adaptée à la s-maintenance prenant appui sur les structures de leurs architectures.

3. Conclusion

Afin de de définir une plateforme de s-maintenance, nous nous sommes attelés dans ce chapitre à répondre à la question suivante : *« Peut-on réutiliser une plateforme existante de maintenance pour une plateforme de s-maintenance ? »*

Pour répondre à cette question, nous avons analysé les plateformes de e-maintenance existantes suivant quatre principales caractéristiques liés à la s-maintenance.

L'analyse de ces plateformes nous a permis de constater que :

- sept plateformes ont exploré le volet « exploitation des connaissances, mais de façon locale dans chaque service pour la plupart mais n'exploitent pas les connaissances quant à la gestion de la plateforme et celle du processus globale de maintenance mais juste pour une activité particulière (comme le diagnostic dans PROTEUS).
- Sur quatorze plateformes, onze sont orientées vers un type ou une activité particulièr(e) de maintenance telle que OSA-CBM ou l'activité de réparation. Seulement trois plateformes traitent le processus global de maintenance.
- En ce qui concerne l'interopérabilité, trois plateformes se sont intéressées au volet sémantique sur neuf autres qui n'ont traitées que le volet syntaxique. Nous pouvons remarquer que le niveau sémantique de l'interopérabilité est négligé dans la plupart des plateformes de e-maintenance.
- En ce qui concerne le niveau de synchronisation, six plateformes sur quatorze ont un mode de coopération et cinq plateformes sont en collaboration. Ceci montre que neuf plateformes sur quatorze ne supportent pas le partage de ressources communes.

En effet, le nombre maximum de caractéristiques liées à la s-maintenance dans une plateforme de e-maintenance est deux. Ainsi, seulement huit plateformes sur les quatorze répondent à la fois à deux de ces caractéristiques. Ceci confirme à la fois les liens et les différences entre la e-maintenance et la s-maintenance.

Ces résultats nous ont permis de confirmer les différences qui existent entre une plateforme de e-maintenance et une de s-maintenance. Malgré les correspondances notées ci-dessus, nous pouvons conclure que dans la plupart des cas, les caractéristiques de la s-maintenance ne sont assurées que partiellement par les plateformes de e-maintenance existantes. En effet, la plateforme de e-maintenance et celle de s-maintenance sont différentes au niveau de l'objectif des services fournis par chaque type de plateforme ainsi que leur propriétés associées.

Toutefois, seulement deux plateformes (conceptuelles) sur les plateformes étudiées sont les plus proches de la plateforme de s-maintenance et sont IIEMED et la Plateforme Orientée Agents (POA). En effet, bien que ces deux plateformes soient orientées vers la coopération, ce qui s'explique par l'absence de connaissances communes partagées entre leur composants logiciels (les services pour la première et les agents pour la deuxième), elles sont orientées connaissances par l'exploitation et la réutilisation des connaissances par leurs composants tout en assurant une interopérabilité sémantique.

Par conséquent, vu le gap entre les services fournis par la e-maintenance et ceux que doit offrir la s-maintenance, nous devons donc développer une architecture spécifique à une plateforme de s-maintenance partant des

caractéristiques et exigences de ce concept ainsi que les points forts de la e-maintenance. Toutefois, nous pourrons nous inspirer des aspects architecturaux et technologiques des deux plateformes IIEMED et POA.

Chapitre 3 Proposition d'une architecture de plateforme de s-maintenance

1. Introduction

Une plateforme de s-maintenance définit une nouvelle génération de systèmes de maintenance permettant la mise en action de la connaissance du domaine, élaborant de nouveaux services répondant à l'évolution des besoins des utilisateurs. Ces services prendront appui sur des fonctionnalités d'auto-apprentissage et d'autogestion pour supporter l'aide à la décision.

Il existe un gap entre ce que fournit les plateformes de maintenance et ce que nous attendons des plateformes de s-maintenance. En effet, assurer des fonctionnalités auto-X (auto-apprentissage, autogestion, etc.), n'est pas faisable avec n'importe quelle plateforme. De plus, proposer des mécanismes de collaboration, partager les connaissances dynamiques, garantir une interopérabilité sémantique, nécessite une plateforme orientée connaissances (basée sur l'ingénierie des connaissances).

Notre objectif peut s'exprimer donc par la question suivante : *« Comment développer une architecture générique et agile capable de répondre aux caractéristiques d'une plateforme générique de s- maintenance ?».*

Dans la section 2, nous avons brossé une brève description des différentes méthodes de génie logiciel (IEEE, 2004) pour restituer la méthode que nous allons adopter pour proposer une architecture de plateforme de s-maintenance. Parmi ces méthodes, l'architecture à base de composants (**ABC**[22]**)** semble le meilleur candidat pour conceptualiser et mettre en place une plateforme de s-maintenance en respectant sa définition. En effet la philosophie des ABC, est centrée sur l'intégration avec un accent particulier sur la sélection des composants qui correspondent aux besoins des intervenants (les acteurs des systèmes) (Mahmood & Lai, 2006). Ainsi le niveau d'abstraction fourni par ce type d'architecture, s'intéresse à la structure du système et non pas aux protocoles de communication et de déploiement. L'argument de choix concernant la méthode ABC est sa généricité qui lui permet d'être déclinée en d'autres types d'architecture comme l'architecture orientée service (AOS).

Par conséquent, nous développons dans ce chapitre une architecture à base de composants pour une plateforme de s-maintenance. Pour la conception de ce système, nous avons choisi une méthodologie comportant deux phases principales, une de pré-analyse et une de conceptualisation. La phase de pré-analyse sera décrite dans la troisième section, et développera une étape d'identification et de définition des composants et de leurs fonctionnalités selon les exigences de la s-maintenance. La deuxième phase de conceptualisation concernerar la modélisation de l'architecture du système par la définition des relations ainsi que des interactions entre ces composants. La quatrième section se focalisera sur les fonctionnalités et les opérations liées aux composants sans rentrer dans les détails techniques de l'implémentation.

Finalement, la discussion sur la possibilité d'implémentation et de mise en œuvre de cette architecture ainsi que la gestion des risques fera l'objet de la cinquième section.

[22] CBA : Component Based Architecture en anglais.

2. Méthodologie adoptée

2.1- Introduction

L'ingénierie des systèmes logiciels est une discipline dédiée à la conception, la mise en œuvre, et la modification des logiciels (Laplante, 2007). Ainsi cette discipline englobe différentes sous disciplines comme l'architecture, la conception, le développement des systèmes logiciels et d'autres (Abran, Moore, Bourque, & Dupuis, 2004). La sous discipline qui nous intéresse dans ce travail est la mise en place d'une architecture de système de s-maintenance.

L'architecture d'un système est un modèle conceptuel qui définit la structure, les comportements et d'autres vues d'un système (IEEE-SA, 2000). Selon Garlan, l'architecture logicielle d'un système définit sa structure de haut niveau, exposant son organisation brute comme une collection de composants en interaction. Une architecture bien définie permet à un ingénieur de raisonner sur les propriétés du système à un haut niveau d'abstraction. Les propriétés typiques de préoccupation incluent la compatibilité entre les composants, les fonctionnalités, la conformité aux normes, la performance, l'orchestration, et la fiabilité (Garlan, Monroe, & Wile, 1997). Ainsi, les détails privés et la mise en œuvre interne des éléments du système ne font pas parties de l'architecture (Bass, Clements, & Kazman, 2003).

Une architecture est dite *« bonne »* si elle répond aux besoins et exigences des parties prenantes (surtout les utilisateurs), à leur satisfaction, ne viole pas les principes établis de l'architecture du système, et prend en compte les propriétés pertinentes permettant la maintenance, l'évolution, le développement, l'intégration, etc. (Pbworks, 2007).

2.2- Choix de la méthodologie

La conception de l'architecture a toujours joué un rôle important dans la détermination de la réussite des systèmes basés sur des logiciels complexes : le choix d'une architecture appropriée peut conduire à un système qui satisfait ses exigences et peut être facilement modifié quand de nouvelles exigences se présentent, par contre une architecture inadéquate peut être désastreuse pour le système (Garlan, Monroe, & Wile, 1997).

En réponse à ce problème, un certain nombre des méthodes formelles pour représenter et analyser des conceptions architecturales ont était proposées. Ces architectures proposées sont classées selon la catégorie de leurs points d'intérêt que se soit la communication, le déploiement, le domaine ou la structure comme le montre le tableau 3-1.

Tableau 3-1 Principaux domaines des styles architecturaux (Microsoft Patterns & Practices Team, 2009)

Catégorie	Styles d'architecture
Communication	Architecture orientée service [Service-Oriented Architecture (SOA)], Bus de message [Message Bus]
Déploiement	Client-serveur [Client/Server], N-Tier, 3-Tier

Domaine	Conception dérivée par le domaine [Domain Driven Design]
Structure	Architecture basée sur les composants [Component-Based], Architecture orientée objets [Object-Oriented], Architecture en couches [Layered Architecture]

De ces différentes méthodes, la démarche de développement d'une architecture à base de composants ***(ABC)***, qui est une branche du génie logiciel insistant sur la séparation des préoccupations en ce qui concerne les fonctionnalités du système, nous semble la meilleure candidate pour modéliser et mettre en place une plateforme de s-maintenance sachant que le développement d'une *ABC* est centrée sur l'intégration, tout en mettant l'accent sur la sélection des composants qui correspondent aux besoins des intervenants (caractéristiques de la s-maintenance, utilisateurs de la plateforme et/ou applications intégrées dans notre cas) (Mahmood & Lai, 2006).

En effet, l'architecture à base de composants ***(ABC)*** se focalise sur la structure du système, et permet ainsi de définir une vue globale et générique du systeme sans rentrer dans les détails de déploiement comme dans le cas de l'architecture Client/Serveur et ses dérivées.

Ainsi, l'architecture à base de composant *(Component-based architecture)* se concentre sur la décomposition lors de la conception, desdifférents composants fonctionnels ou logiques qui exposent des interfaces de communication bien définies contenant des méthodes, des événements et des propriétés. Cela fournit d'une part, un niveau supérieur d'abstraction par rapport aux principes de conception orientée objet, et ne se concentre pas sur des questions telles que les protocoles de communication et de l'état partagé (Microsoft Patterns & Practices Team, 2009), et d'autre part, un niveau moins abstrait que propose l'architecture en couches.

En outre, l'architecture à base de composants et l'architecture orientée services semblent avoir le même but qui est de fournir une base pour l'architecture logicielle vaguement jointes *(loosely joined)* et hautement interopérables, permettant un développement logiciel efficace et sans erreurs (Petritsch, 2008). Nous rappelons qu'une architecture orientée services est un ensemble de composants qui peuvent être invoqués et dont les interfaces de descriptions peuvent être publiées et découvertes. Ainsi, un grand nombre de caractères de l'architecture AOS viennent de l'architecture à base de composants. L'architecture AOS peut être considérer comme une évolution de l'ABC ou une spécialisation où le composant est présenté sous forme de service suivant un protocole de communication particulier (Petritsch, 2008).

D'autre part, un composant est un objet logiciel, destiné à interagir avec d'autres composants, en encapsulant un ensemble de fonctionnalités. Un composant a une interface clairement définie et est conforme à un comportement usuel prescrit à tous les composants au sein d'une architecture (Petritsch, 2008).

Les principales clés de ce style d'architecture à base de composants est l'utilisation de composants qui sont (Microsoft Patterns & Practices Team, 2009) : réutilisables, remplaçables, extensibles, encapsulés et indépendants.

L'architecture à base de composants ABC, se compose de tous les composants nécessaires pour capturer, traiter, transférer, stocker, afficher et gérer les données et les connaissances. Les composants incluent des applications logiciels, des transformateurs de données, des processeurs, des réseaux, des bus de données, du *firmware*, des

applications spécifiques pour l'intégration, des dispositifs de stockage, et l'utilisateur humain (qui fait partie du système en interagissant avec les composants) (Encyclopedia, 2011). Un composant peut être un logiciel, un service Web, ou un module qui encapsule un ensemble de fonctions connexes (ou données). Par conséquent , le succès de l'*ABC* dépend de sa capacité à sélectionner les composants appropriés (Mahmood & Lai, 2006). En effet, une sélection inappropriée des composants peut entraîner des effets néfastes tels qu'une courte liste de composants qui à peine remplir les fonctionnalités nécessaires ou qui introduit des coûts supplémentaires dans les phases d'intégration et de maintenance (KRPH & Leung, 2002). D'autre part, les composants individuels fournissent habituellement des capacités fixes qui pourraient ne pas satisfaire à toutes les exigences du système et certains d'entre eux peuvent être inutiles dans le système. Cela réduit les chances de compatibilité entre un composant et les exigences des parties prenantes du système (les caractéristiques de la s-maintenance dans notre cas).

Ainsi, il faut rechercher à équilibrer la satisfaction des exigences par un composant par rapport aux risques qui peuvent être engendrés. L'évaluation des risques est un autre attribut important qui affecte le développement global d'une *ABC*, car cet attribut fournit une base pour comparer les composants candidats en se concentrant sur leurs profils de risque.

Le développement d'une ABC est un processus complexe et risqué qui nécessite une technique d'analyse souple des exigences permettant de comprendre les exigences des intervenants et les fonctionnalités des composants. Il existe différentes méthodes, approches et langages pour concevoir et décrire u*ABC*, comme cela est proposé dans (Stein & Louca, 1995), (Garlan, Monroe, & Wile, 1997), (Kotonya & Hutchinson, 2004) et (Gudas & Pakalnickas, 2006),. Comme notre système est spécifique au concept de s-maintenance, adoptons une approche simple contenant deux phases, une phase de pré-analyse concernant l'identification des composants intéressants correspondants aux exigences de la s-maintenance, et une deuxième phase concernant la conception de l'architecture du système en définissant les relations ainsi que les interactions entre ces composants sans rentrer dans les détails techniques ou les interfaces de communications entre ces composants pour laisser la liberté au développeur du système.

3. Phase de pré-analyse : Spécification des composants

3.1- Caractéristiques d'une plateforme de s-maintenance

Selon les définitions du concept de s-maintenance et celle de la plateforme fournies dans le premier chapitre nous pouvons dégager les sept exigences (caractéristiques) à respecter par une plateforme de s-maintenanceà savoir :

- Connaissances expertes du domaine partagées
- Capacités de raisonnements : système à base de connaissances
- Communication sémantiquement interopérable
- Collaboration entre utilisateurs et applications : gestion des groupes d'utilisateurs, gestion des accès et orchestration de l'exécution des processus
- Services à la demande : prenant appui sur la dynamique des connaissances et l'inférence

- Fonctionnalité auto-apprentissage : traçabilité des interactions et évolution dynamique des connaissances
- Fonctionnalité de d'autogestion : orchestration dynamique des processus

3.2- Identification des composants

Cette phase sera divisée en deux parties : 1) l'identification des composants génériques spécifique à la maintenance, 2) l'identification des composants spécifiques aux exigences de fonctionnement de la plateforme de s-maintenance.

3.2.1- Identification des composants génériques

Il faut mentionner que les composants d'un Système d'Information peuvent être très différents d'une organisation à une autre ou d'un domaine à un autre et peuvent recouvrir selon le cas tout ou une partie des éléments suivants de l'entreprise, à savoir les Bases de données de l'entreprise, les progiciels de gestion intégré (ERP), les outils de gestion de la relation client (Customer Relationship Management),et de la chaîne logistique (SCM-Supply Chain Management), les applications métiers, l'infrastructure réseau, les serveurs de données et systèmes de stockage, les serveurs d'application, les dispositifs de sécurité.

En prenant appui sur une première étude faite sur la e-maintenance dans le projet PROTEUS qui a proposé d'intégrer un ensemble d'applications dans un système d'information intelligent de maintenance industriel (Rebeuf, et al., 2004), nous proposons un ensemble minimale d'applications que peut intégrer aujourd'hui une plateforme de s-maintenance : un système d'acquisition de données (SCADA, Supervisory Control And Data Acquisition), un système de diagnostic, un système de GMAO (Gestion de Maintenance Assisté par Ordinateur), un système ERP (Enterprise Resource Planning), un système de documentation (eDoc), un système de pronostic, un système de diagnostic, un système de planification et ordonnancement, et enfin un système de géo-localisation.

Un système d'acquisition de données *(SCADA, Supervisory Control And Data Acquisition)* : c'est un système qui permet d'acquérir des données à partir de capteurs implantés sur unsystème physique, afin de suivre l'état du système et de détecter les alarmes et éventuellement les dérives qui nécessiteront une intervention immédiate ou programmée (Rebeuf, et al., 2004).

Un système de GMAO - CMMS en anglais (Gestion de Maintenance Assistée par Ordinateur) : C'est un système de gestion des activités de maintenance qui gére les diverses stratégies de maintenance, à savoir corrective, préventive, prédictive, etc, le planning des activités de maintenance et des interventions. Les objectifs d'une GMAO peuvent ainsi être décomposés selon les trois axes suivants : la gestion proprement dite de l'activité de maintenance, la génération du retour d'expérience (REX) et l'analyse des données tout au long des autres processus (Rebeuf, et al., 2004).

Un progiciel de gestion intégrée (**PGI**)[23] : C'est un progiciel qui a comme vocation l'optimisation de la productivité de l'entreprise qui est basée sur une gestion, une optimisation et une synchronisation des flux des

[23] ERP (Enterprise Resource Planning)

matières, des produits, d'information, de décision et évidemment des flux financiers. De tels systèmes intègrent toutes les informations de l'entreprise, ressources humaines, gestion des achats et de la logistique, service commercial et gestion des ventes, production et gestion des matières, qualité des produits et des services, gestion comptable et financière, et évidemment la maintenance qui est en relation avec chacune des fonctions précédentes (Rebeuf, et al., 2004).

Un système de documentation – e-Doc : Ce système contient au minimum la base documentaire associée à l'équipement à maintenir. et contiennent aujourd'hui tout ce qui est relatif à la qualité, à la sécurité, aux normes, aux règlements intérieurs, et plus généralement toutes les informations internes à l'entreprise (Rebeuf, et al., 2004).

Un système de Diagnostic : C'est un système qui peut être off-line et est utilisé comme d'assistance à la recherche de la cause du dysfonctionnement et se déclenche par l'utilisateur lors d'une panne. Il peut être on line et détecte les dysfonctionnements automatiquement et les relie aux équipements causant la défaillance. . Il s'appuie sur des données issues d'un SCADA et sur l'expérience acquise pour assister la décision des responsables de maintenance.

Un système de Pronostic : Ce système doit anticiper les défaillances par la surveillance de l'état de l'équipement. et donner l'ensemble des durées de vie résiduelles (de l'équipement, de ses composants et de ses sous composants) pour les modes de défaillances détectés et induits, à partir de connaissances sur l'équipement (structure, composition, technologie…) et d'informations passées (historique des modes de fonctionnement passés), présentes (état courant) et futures (scénario de production et de maintenance) (Cocheteux, Voisin, Levrat, & Iung, 2007).

Un système de planification : Ce système traite des calendriers de maintenance à réaliser ainsi que de leur validationet l'optimisation du temps planifié pour chaque tâche. En effet, il vise à organiser au mieux l'activité de l'atelier, en liaison avec en amont les prévisions de travaux à effectuer (actions de maintenance périodiques, maintenance préventive suite au pronostic ou réparation suite à un diagnostic de défaillance) et en aval la logistique (approvisionnements, stocks, échanges de pièces, activité de l'atelier, etc.).

Un Système de géo-localisation : Ce système quand il est intégré répond à des questions de base tel que : Où ? Cette interrogation permet de mettre en évidence la répartition spatiale d'un équipement. Quoi? Il s'agit de mettre en évidence tous les équipements ou phénomènes présents sur un territoire donné. Comment ? Quelles relations existent ou non entre les équipements et les phénomènes ? C'est la problématique de l'analyse spatiale. Quand? A quel moment des changements sont intervenus? C'est la problématique de l'analyse temporelle.

Chaque système s'appuie sur le modèle de l'entreprise, le système physique ou l'équipement à maintenir. Tous ces composants peuvent être intégrés comme des applications externes dans la plateforme de s-maintenance.

3.2.2- Identification des composants spécifiques

Nous procédons maintenant à l'identification des composants spécifiques à la plateforme de s-maintenance. On rappelle que les caractéristiques de la s-maintenance ont considéré les exigences des *partis prenants*[24] du CBS.

Concernant la caractéristique d'autogestion du processus de maintenance, un composant de gestion de processus est nécessaire. Ainsi, ceci nécessite la présence de deux composants majeurs qui sont à la fois une base de connaissances contenant les connaissances des experts du domaine afin de partager une sémantique commune entre les différents acteurs de la maintenance, ainsi qu'un moteur de raisonnement inférant de nouvelles connaissances et de l'intelligence à partir de cette base de connaissances. Ainsi, afin d'assurer la cohérence la consistance de cette base de connaissances, un composant assurant le contrôle et la gestion de cette base est nécessaire.

D'autre part, la collaboration a besoin d'un élément de coordination entre les différents acteurs de la collaboration du processus de maintenance. Ainsi, elle a besoin d'une base de connaissances pour partager les connaissances entre ces acteurs. Ainsi, Elle a besoin d'un composant assurant la sécurité dans la plateforme et gérant les rôles des acteurs de cette collaboration, ainsi qu'un composant pour une interface homme-machine dynamique adaptant son contenu selon le contexte et selon les rôles des utilisateurs.

Vis-à-vis de l'interopérabilité sémantique, un composant de médiation sémantique est nécessaire pour assurer cette fonctionnalité basée sur la sémantique partagée de la base de connaissances.

Pour la caractéristique d'auto-apprentissage, deux composants sont fortement recommandés, un composant assurant la traçabilité et un autre pour le raisonnement dans le but de sauvegarder et ensuite analyser les interactions dans la plateforme ce qui permettra d'apprendre et de comprendre ces interactions et enfin de rajouter de nouvelles connaissances dans la base de connaissances du système.

Finalement, à propos la génération dynamique de services à la demande qui est un point crucial dans la s-maintenance, la présence du moteur de raisonnement est inévitable, aussi un composant pour la génération des nouveaux services doivent être présents aussi bien qu'une librairie (répertoire) pour sauvegarder les nouveaux services générés.

En résumé, nous identifions différents composants qui doivent être inclus dans un système de s-maintenance répondant à ses besoins. Ces composants sont: 1) un coordinateur (*Coordinateur*), 2) un médiateur sémantique (*Médiateur*), 3) un gestionnaire d'interface homme machine (gestionnaire d'interface *H/M- GIM*), 4) un contrôleur d'accès (*Contrôleur d'accès- ConA*), 5) un gestionnaire de la base de connaissances *(gestionnaire de base de connaissances- GBS)*, 6) une base de connaissances *(base de connaissance- BaC)*, 7) un moteur de raisonnement (*Raisonneur*), 8) une bibliothèque de service (*bibliothèque de service- BiS*), 9) un générateur de service (*Générateur de service- GéS*), 10) un gestionnaire de processus (*Gestionnaire de proucessus- GeP*), 11) un gestionnaire de traçabilité modélisée (*Gestionnaire de traces modélisées- GTM*).

La définition de ces composants, leurs fonctionnalités ainsi que leurs interactions seront décrits dans la section suivante.

[24] stakeholders

4. Phase de conceptualisation : proposition d'une architecture

A partir de l'identification des composants faite dans la phase de pré-analyse tout en respectant les exigences des caractéristiques du concept et de la plateforme la s-maintenance, nous proposons une architecture de plateforme composée de 6 couches logicielles dont 5 font partie du noyau de la plateforme. La Figure 3-1 donne une vue globale de ces différentes.

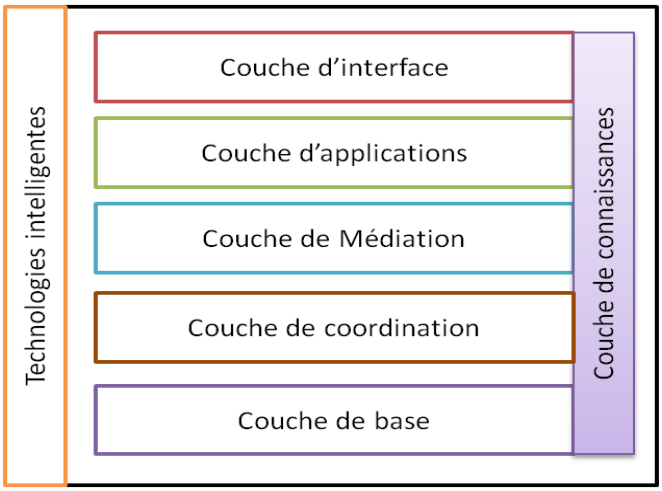


Figure 3-1 Différentes couches de la plateforme de s-maintenance

Le cœur de la plateforme se compose principalement des couches intégrante ces composants spécifiques.

La couche de base intègre les composants spécifiques qui servent à assurer les fonctionnalités innovantes et qui caractérisent la s-maintenance (comme les fonctionnalités d'auto-X) à savoir : le moteur de raisonnement, le générateur de services, la bibliothèque de services, le gestionnaire de processus et le gestionnaire d'apprentissage.

La couche de coordination intègre le composant coordinateur qui assure la coordination et la synchronisation entre les différents composants de la plateforme que ce soit génériques ou spécifiques.

La couche de médiation intègre le composant médiateur qui assure un échange interopérable sémantiquement entre tous les composants que ce soit génériques ou spécifiques.

La couche d'applications intégrée tous les composants génériques de la plateforme et qui représentent toutes les applications d'aide à la maintenance qu'intègre la plateforme.

La couche interface intègre les composants gestionnaire d'interface H/M et gestionnaire d'accès.

La couche de connaissances qui est partagée et exploitée par toutes les autres couches, intègre la base de connaissances et son gestionnaire associé.

Tout en respectant les différentes couches identifiées, nous présentons dans la Figure 3-2 l'architecture globale de cette plateforme et les différentes relations entre les composants génériques et les composants spécifiques

ainsi que la relation avec les équipements à gérer et les technologies intelligentes, ces technologies intégrées englobent les capteurs intelligents, le RFID[25], etc.

La partie encadrée en rouge dans cette Figure présente le cœur de la plateforme et se compose principalement de composants spécifiques, les blocs en vert présentent les composants génériques de la plateforme et la partie encadrée en jaune présente les technologies intelligentes qui relient la plateforme aux équipements qu'elle gère (la partie encadrée en orangé). La partie réseau dans cette architecture concerne les échanges entre les différents composants de la plateforme. En ce qui concerne les interactions entre composants et les détails de leur fonctionnement, ceci fera l'objet de la section suivante.

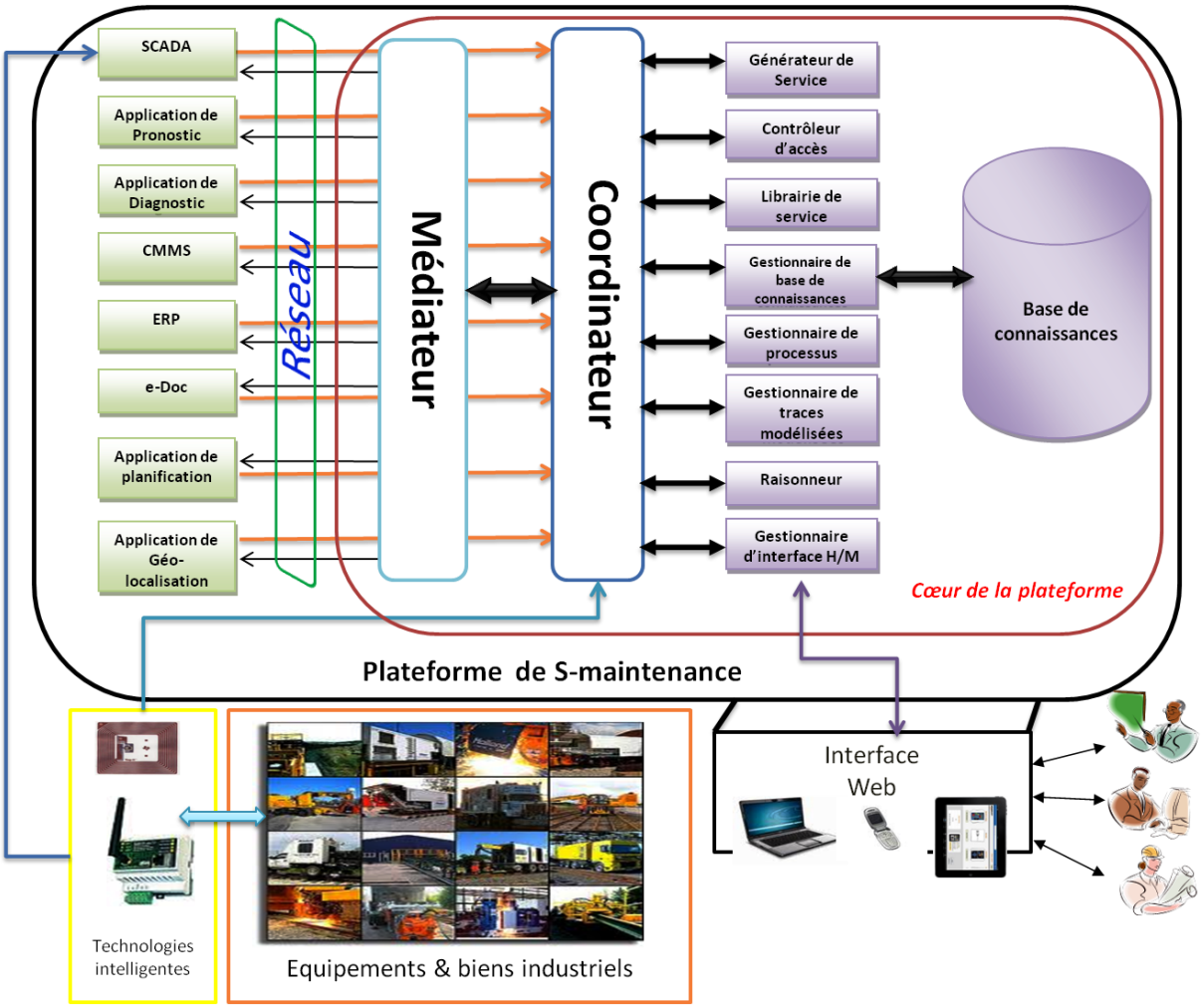


Figure 3-2 Architecture de la plateforme s-maintenance proposée

4.1- Modélisation et fonctionnement des composants du cœur de la plateforme

Dans cette section nous verrons plus en détail le cœur de la plateforme en définissant chaque composant, ses fonctionnalités et ses comportements avec les autres composants (comme indiqué dans la Figure 3-2).

4.1.1GIM - Gestionnaire d'interface H/M

[25] « Radio Frequency IDentification », en français, « Identification par Radio Fréquence »

Ce composant gère tous les affichages de l'interface Web au bon format et leur contenu adapté aux différents types de terminaux clients (PC traditionnels, un PDA, un téléphone mobile). Les utilisateurs interagissent avec la plate-forme de maintenance via cette interface Web. Cette dernière doit être simple, bien conçue et orientée utilisateurs. A travers l'interface Web et grâce au gestionnaire d'interface H/M (*GIM),* l'utilisateur peut demander l'accès aux données dont il a besoin pour compléter ses tâches en cours. Elle permet aussi à l'utilisateur d'ajouter de nouvelles informations et de s'assurer qu'elles sont stockées dans la base de connaissance. Elle doit également permettre l'échange de tout type de données pertinentes.

Un acteur de maintenance (manager, expert, operateur) doit être capable de se connecter à la plate-forme via un navigateur Web sur n'importe quel type de terminal client. Le *GIM*, permet de gérer des formats différents, comme les documents texte, les informations, les données établies par un SCADA, un ERP ou d'un système de GMAO.

4.1.2- ConA - Contrôleur d'accès

Différents acteurs peuvent intervenir sur la plateforme. Il faut donc pouvoir gérer les droits d'accès des utilisateurs comme ceux des applications. Le contrôleur d'accès *(ConA)* est responsable de l'authentification des utilisateurs dans le cas de requête d'accès à la plateforme et du contrôle de l'accès aux données ou à la base de connaissances. Il gère aussi l'accès aux contenus des autres composants et applications intégrées dans la plateforme dans le cas d'une requête de demande d'informations ou d'échange entre applications.

ConA utilise le mécanisme RBAC[26] qui présente un modèle de contrôle d'accès à un système informatique dans lequel chaque décision d'accès se base sur le rôle de l'utilisateur (Osborn, Sandhu, & Munawer, 2000). Donc, selon les identités et les fonctions des utilisateurs ainsi que des applications intégrées (composants génériques), les rôles sont classés selon différents groupes d'utilisateurs (manager, expert, opérateur). On accorde à chaque rôle un ensemble d'autorisations afin d'utiliser la plateforme de s-maintenance.

4.1.3- BaC – Base de connaissance

En général, une **base de connaissance** regroupe des connaissances spécifiques à un domaine spécialisé donné, formalisées de manière déclarative sous une forme exploitable par un ordinateur (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2007).

Les connaissances manipulées doivent être des connaissances susceptibles d'influencer le déroulement du processus de maintenance, de produire de nouvelles connaissances et permettre de prendre des décisions.

Dans notre cas, le composant base de connaissance ***(BaC)*** contient les connaissances expertes du domaine de maintenance. Cette ontologie que nous avons développée fera l'objet du prochain chapitre. L'ontologie est présentée dans la base de connaissances via un langage de description. Une base de connaissances sous ce format est composée généralement de deux types de connaissances contenues dans le *TBOX* et le *ABOX* (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2007). Le *TBOX (Terminology Box)* contient la description

[26] Role Based Access Control- http://en.wikipedia.org/wiki/Role-based_access_control

des concepts, les relations entre ces concepts ainsi que les règles logiques. Le *ABOX (Assertion Box)* contient les instances des concepts et des relations définis dans le *TBOX*.

En effet, le composant *BaC* est composé d'un *TBOX* contenant l'ontologie de maintenance et le *ABOX* contenant les instances de cette ontologie.

4.1.4- GBC – Gestionnaire de bases de connaissance

Ce composant fait référence au KBMS *(A Knowledge Base Management System)*. Un système de gestion de base de connaissances est une application informatique pour la gestion et la manipulation (création, amélioration et maintien) d'une base de connaissances, tout comme un SGBD (system de gestion de base de données) (Firestone, 1999).

Le gestionnaire de base de connaissance *(GBC)* vérifie la cohérence des connaissances d'entrées/sorties avec l'Ontologie de la base (toutes les connaissances, que les autres composants interrogent ou enregistrent dans le *BaC*, doivent être cohérentes avec l'Ontologie.)

Le *GBC* assure le contrôle d'intégrité des connaissances instanciées par rapport aux définitions des concepts, des règles, des relations et des contraintes (TBOX). De plus le *GBC* interdit d'ajouter des règles contradictoires dans le *BaC*.

Ce composant est associé à un contexte de persistance[27]. Un contexte de persistance est un ensemble d'instances d'entité dans laquelle il y a une instance d'entité unique pour toute identité d'entité persistante.

4.1.5- Raisonneur - Moteur de raisonnement

L'objectif global d'un moteur de raisonnement est de raisonner sur les connaissances d'une base de connaissances en fournissant et inférant des réponses, des prédictions et des suggestions similairement à la manière d'un expert humain. De nouvelles connaissances peuvent être générées à partir des connaissances existantes grâce au moteur de raisonnement. Celui-ci contient des algorithmes généraux qui permettent la manipulation des connaissances stockées dans la base de connaissances pour résoudre des problèmes. Le mécanisme de raisonnement utilisé dépend de la représentation des connaissances choisie et peut s'appliquer à une variété de domaine (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2007).

Dans le cadre de la plateforme de s-maintenance, le raisonneur peut contenir différents types de raisonnement (déductif, inductif, analogique, par renforcement, flou, possibiliste, connexionniste, etc). Ils sont utilisés pour raisonner sur le *BaC*. Le raisonneur peut être exploité par les applications intégrées (ex. diagnostic, pronostic) ou par les composants spécifiques (*ex. GBC, GéS*), qui envoient leurs demandes au Coordinateur qui les dirige au Raisonneur.

4.1.6- Coordinateur

[27] http://www.avaje.org/persistencecontext.html

Le composant Coordinateur est l'un des composants les plus importants de la plateforme après la base de connaissances. Il joue le rôle de coordination entre les applications intégrées et les autres composants de la plateforme. Tous les composants lui sont connectés. Ainsi, ce composant est aussi le pont de communication entre les utilisateurs et la plateforme. Le fonctionnement du coordinateur est inspiré du fonctionnement du bus de message appelé en anglais *« message bus »*[28]. Un bus de message est un composant logique sur lequel se connectent toutes les applications dans un système distribué. Il est spécialisé dans la transmission de messages entre les applications.

Les principales fonctions du *Coordinateur* sont:

- L'identification des applications fournissant les services demandés par une requête d'un utilisateur ou d'une autre application.
- L'identification de la direction de requête vers son destinataire
- Le renvoi de la réponse d'une requête à son application émetteur

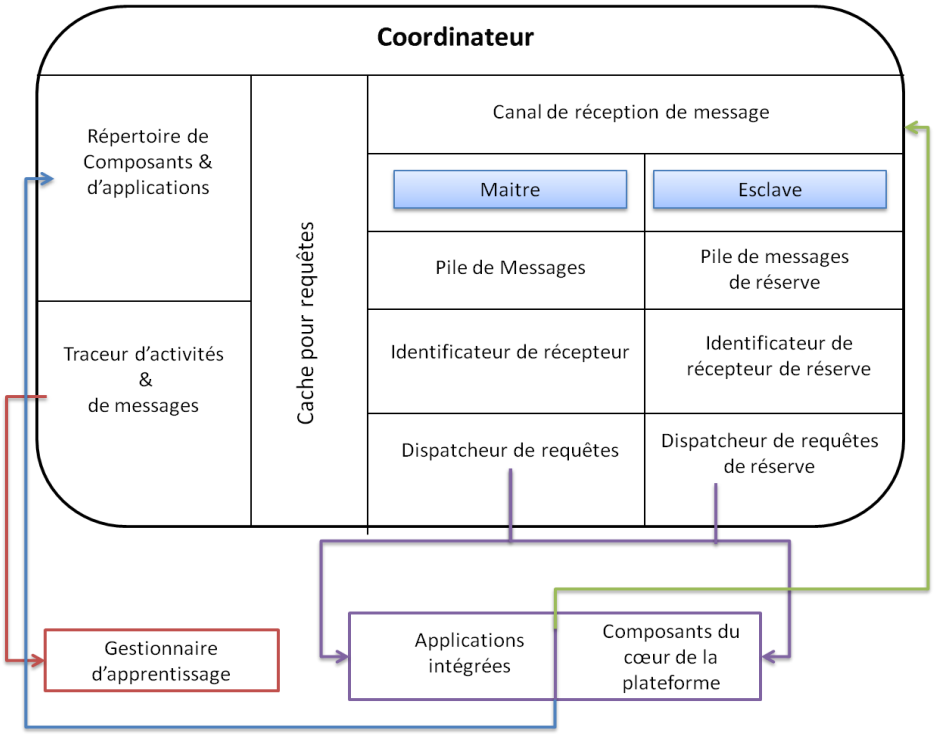


Figure 3-3 Architecture du coordinateur

4.1.6.1- Architecture du Coordinateur

La Figure 3-3 montre l'architecture du coordinateur qui se compose de 10 sous composants, à savoir :

- *« Répertoire de composants et applications » :* qui est un annuaire contenant la liste des composants et des applications intégrées dans la plateforme ainsi que la description de leurs fonctionnalités. Comme indiqué avec la flèche bleue sur la Figure, ce composant se met à jour à chaque fois qu'une nouvelle application est intégrée ou un composant est rajouté dans la plateforme. cet annuaire sert à localiser les

28 http://msdn.microsoft.com/en-us/library/ff647328.aspx

sources d'informations, la redirection des requêtes, la connaissance des ports de connexion et l'échanges avec ces applications.

- *« Canal de réception de message » :* qui est un canal de communication, ou média de transmission d'information. Il relie la source au destinataire et permet l'acheminement du message. Ce composant relie les applications intégrées à la plateforme et les composants au coordinateur et sur lequel les messages sont transmis grâce à un protocole de communication.
- *« Traceur d'activités et de messages -TAM » :* sert à tracer toutes les activités, les messages, les interactions qui passent par le coordinateur ou les activités exécutées par les applications liées aux coordinateur. Nous rappelons que la traçabilité peut être définie comme l'aptitude à retrouver l'historique, l'utilisation ou la localisation d'une entité aux moyens d'identification enregistrés (Puget, 2005). Ce type de traçabilité se consigne dans un journal d'interactions, qui sera transmis au composant *« Gestionnaire de traces modélisées »*. Cette traçabilité note qui a fait quoi, quand, où et avec quoi.
- *« Cache pour requêtes » :* présente une mémoire tampon qui est une zone de mémoire vive ou de disque utilisé pour stocker temporairement des requêtes (messages)[29]. Il consulte la pile de message et enregistre temporairement les requêtes passées par le coordinateur, leurs destinataires identifiés. Son rôle est d'optimiser le fonctionnement du coordinateur et accélérer le traitement des messages reçus et vider la *« Pile de messages »*.

Il y a trois composants dupliqués dans le coordinateur (composants maîtres vers les composants de réserves), à savoir la *pile de messages, l'identificateur de récepteur et le dispatcheur de requêtes*. Sachant que les composants esclaves ne fonctionnent que lorsque la charge des messages reçus est grande. De même, le bus de messages du *Canal de réception* est encombré et aussi le temps de réponse commence à devenir plus long. Le choix d'introduire des sous composants maîtres et esclaves rentre dans la stratégie de gestion de risques de dysfonctionnements dans le coordinateur. Cette gestion de risque sera détaillée un peu plus tard.

- *« Pile de messages et Pile de messages de réserve » :* elle permet de placer les messages reçus (via le canal de réception de message) dans la file jusqu'à leur prise en charge par le composant *Sources identifier* après un certain temps. La méthode de récupération est la méthode FIFO avec priorité.
- *« Identificateur de récepteurs et Identificateur de récepteurs de réserve » :* permet d'identifier l'application(s) ou composant(s) destinataire du message. Ce composant se base sur les descriptions enregistrées dans le répertoire des components et d'applications pour identifier le destinataire.
- *« Dispatcheur de requêtes et Dispatcheur de requêtes de réserve » :* prend en charge la transmission du message à son destinataire identifié. Il se base aussi sur le répertoire des composants et d'applications pour transmettre le message.

4.1.6.2- Les interactions du coordinateur avec les autres composants

Comme le montre la Figure 3-2 décrivant l'architecture de la plateforme s-maintenance, le composant *Coordinateur* est lié directement ou indirectement à tous les composants génériques (applications intégrées) et spécifiques à la plateforme.

[29] Dans les architectures clients serveur les messages envoyés par le client au serveur sont appelés requêtes

De plus comme le montre la Figure 9, les flèches bleues, rouges, vertes et mauves regroupent les différents types d'interconnexions entre les sous composants du coordinateur et les autres composants de la plateforme.

Dans le cadre du fonctionnement globale de la plateforme toutes les communications et les échanges passent par le coordinateur, mais il y a d'autres relations plus spécifiques avec d'autres cas :

- Le coordinateur ne peut accéder directement à la base de connaissances. Il demande au *gestionnaire de base de connaissances (GBS)* d'interroger le *BaC- (Base de Connaissances)* ou d'enregistrer de nouvelles connaissances.
- Le coordinateur doit respecter les règles métiers dans sa gestion de transmission de requêtes utilisateurs. Pour cela, il est toujours en lien avec le composant *gestionnaire de processus* qui gère les règles d'orchestrations métiers. Donc en respectant les orientations du *GeP* le coordinateur identifie les composants et les composants destinataires et transfère les requêtes.
- Lorsque la plateforme reçoit une requête demandant un service non fourni par les applications intégrées (composants génériques), le *Coordinateur* dirige l'utilisateur vers le composant *BiS-(Librairie de service).* Si le service demandé n'existe pas dans le *BiS, Coordinateur* redirige alors le demandeur du service au *GéS-(Générateur de service)* pour créer ce nouveau service.
- Le coordinateur contacte le *ConA-(Contrôleur d'accès)* dans le cas où la requête reçue est une requête d'authentification, ou dans le cas où un composant veut consulter un document (des informations, des données) à partir d'autres composants. Dans ce deuxième cas, le *Coordinateur* envoie une demande au *ConA* afin de connaître les parties de données ou du document pouvant être accessibles et envoyés aux demandeurs ou à l'affichage sur l'Interface Web.

Le coordinateur envoie périodiquement les journaux d'interactions au composant *gestionnaire de traces modélisées (GTM).*

4.1.7- Médiateur

Une des valeurs ajoutées de la plateforme de s-maintenance est l'interopérabilité sémantique. Cette dernière consiste à donner du « sens » (une sémantique) aux informations échangées et de s'assurer que ce sens soit partagé dans tous les systèmes entre lesquels les échanges doivent être mis en œuvre (Levy, Rajaraman, & Ordille, 1996).

L'architecture de la plateforme de s-maintenance intègre des applications (composants génériques) susceptibles de faire évoluer l'ontologie commune du domaine de maintenance (par rapport à leurs besoins internes). Pour assurer l'interopérabilité sémantique entre ces applications, la plateforme doit prendre en compte l'évolution des ontologies locales de ses applications intégrées.

4.1.7.1- Approches pour l'interopérabilité sémantique

Mettre en place une interopérabilité sémantique entre différents systèmes d'information est laborieux et source d'erreurs dans un environnement distribué et hétérogène (Mao, 2008). Cette interopérabilité a fait l'objet de

divers travaux de recherche qui ont été classés selon Park et Ram (Park & Ram, 2004) en trois grandes approches à savoir ; approche basée sur la Cartographie (mapping), Approche orientée requêtes, Approche basée sur la médiation ainsi que d'autres approches comme l'interopérabilité basée sur un modèle (*Model Driven Interopearbility*) (Mellor, Scott, Uhl, & Weise, 2002) ou l'interopérabilité orientée service (Chen, Doumeingts, & Vernadat, 2008).

Souvent, les médiateurs utilisent des ontologies afin de partager un vocabulaire ou des protocoles normalisés pour communiquer les uns avec les autres (Halevy & Madhavan, 2003) (Maedche & Staab, 2000). L'avantage d'utiliser les ontologies est leur capacité à acquérir les connaissances tacites détaillées dans un domaine afin de fournir une conceptualisation riche des objets et de leurs relations (Park & Ram, 2004).

Parmi ces différentes approches, nous avons choisi l'approche de médiation basée sur l'ingénierie ontologique. En effet, la majorité de chercheurs connus dans ce domaine comme Mizoguchi, Guarino, Bittner, et Obrstes affirment que l'ingénierie ontologique est reconnue comme la technologie clé pour traiter le problème d'interopérabilité sémantique (Heiler, 1995) (Obrst, Ontologies for Semantically Interoperable Systems, 2003) (Mizoguchi R. , 2004) (Yang & Zhang, 2006) (Bittner, Dinnelly, & Winter, 2006).

4.1.7.2- Architecture et fonctionnement du médiateur

Un médiateur est défini par Wiederhold (Wiederhold G. , 1991) en tant que module logiciel qui exploite les connaissances encodées sur un ensemble particulier de données permettant de mettre la source d'information sous une forme commune pour une couche supérieure d'applications. Par conséquent, un médiateur peut apporter des services à valeur ajoutée (appelés services de médiation sémantique), tels que (1) l'accès et la récupération des données pertinentes à partir de plusieurs applications hétérogènes et (2) l'abstraction et la transformation des données récupérées dans des représentations et des sémantiques communes (Wiederhold & Genesereth, 1997). Ainsi, le composant *Médiateur* se réfère à un agent logiciel qui est responsable de la réconciliation sémantique, qui est, le processus d'identification et de résolution des conflits sémantiques dans la plateforme de s-maintenance.

Pour cela, le composant *médiateur* de la plateforme doit assurer l'alignement entre les versions des ontologies des différentes applications intégrées dans la plateforme. En d'autres termes, le *médiateur* est un système de médiation sémantique permettant un alignement entre les différentes versions d'ontologies.

Comme le montre la Figure 3-4, le *médiateur* inclut quatre principaux modules tout en prenant appui sur la base de connaissances commune.

En effet, comme indiqué dans l'architecture de la plateforme, toute requête entre composants passe par le coordinateur qui transmet la requête au *médiateur* après avoir identifié le composant ciblé par la requête.

Ainsi, le composant gestionnaire de base de connaissance, alimente la base de connaissance commune en récupérant dynamiquement les TBOX des ontologies locales de ces systèmes intégrés. Chaque mise à jour faite sur le TBOX de l'ontologie locale, la base de connaissance commune est automatiquement mise à jour. Par conséquent, la base de connaissance réunit l'ontologie commune et les ontologies locales des différentes applications.

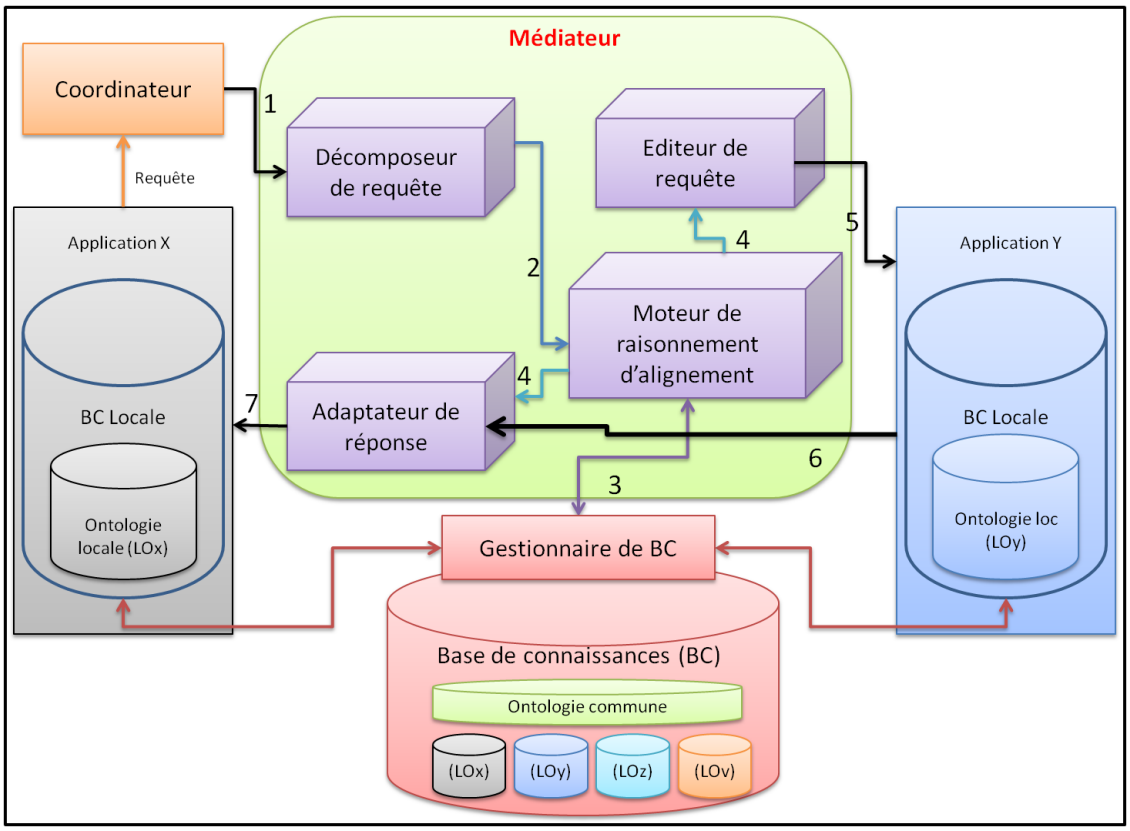


Figure 3-4 Architecture globale du système médiateur

Le composant « *Décomposeur de requête* » est responsable de la réception de requête envoyée par un système intégré, il identifie les concepts et les relations utilisées dans la requête.

Le composant *«Moteur de raisonnement d'alignement - MRA »* est un moteur de raisonnement pour l'alignement d'ontologies, il assure la cartographie *(mapping)* entre les concepts et les relations des ontologies locales des systèmes intégrés à partir de la base de connaissances partagée et en utilisant un algorithme d'alignement. Karray et al dans (Karray, Chebel-Morello, & Zerhouni, 2010) ont proposé un algorithme d'alignement orienté vers une ontologie de maintenance qui peut être utilisé pour ce composant. D'autres possibilités d'algorithmes d'alignement d'actualités et vérifiés peuvent être exploitées aussi comme ceux présentés dans (Euzenat, Meilicke, Stuckenschmidt, Shvaiko, & Trojahn, 2011) (Bellahsene, Bonifati, & Rahm, 20111) et (Granitzer, Sabol, Weng Onn, Lukose, & Tochtermann, 2010).

Ainsi, le *MRA* identifie les correspondances des concepts et les relations utilisées dans la requête reçue avec ceux de l'ontologie locale du composant de la requête. Après avoir trouvé les correspondances, il met à jour la base des connaissances en ajoutant les règles d'équivalence résultant de cet exercice pour que cela soit réutilisé directement lors des prochaines requêtes.

Le composant « *Editeur de requête* » réécrit les requêtes reçues par la plateforme en utilisant les concepts et les relations des systèmes cibles identifiées par le *«MRA»* et envoie les nouvelles requêtes aux systèmes ou composants cibles.

Le composant *« Adaptateur de réponse »* réécrit la réponse reçue de l'application cible en respectant l'ontologie locale de l'application qui a envoyé la requête concernée.

Concernant le fonctionnement du système médiateur, les flèches numérotées sur la Figure (3-4) présentent la séquence des échanges entre les composants du médiateur lors des étapes de la médiation suivantes :

1- Réception de la requête envoyée par le système concerné
2- Identification des concepts et des relations utilisés dans la requête
3- Transmission des concepts et des relations identifiés au *« MRA »*
4- Consultation de la base de connaissances par le *« MRA »* pour voir s'il y a correspondance entre les concepts de utilisés dans la requête et les concepts de l'ontologie locale du système cible. Trois cas se présentent :
    a. Les concepts de la requête sont juste des concepts de l'ontologie partagée de la plateforme, donc il n y a pas un besoin de correspondance, et la requête sera envoyée au système cible comme elle est.
    b. Si une correspondance existe déjà dans la base des connaissances, *MRA* transfère les règles de correspondances aux composants *« Editeur de requête »* et *«Adaptateur de réponse »*.
    c. Sinon, le *« MRA »* applique son algorithme d'alignement et infère de nouvelles règles de correspondances. Ainsi il les enregistre dans la base des connaissances, et les envoies aux composants *« Editeur de requête »* et *«Adaptateur de réponse »*.
5- Réécriture de la requête originale par le *« Editeur de requête »* en utilisant les concepts correspondants et génération d'une nouvelle requête qui sera envoyée au système cible.
6- Envoi de la réponse à la requête au *«Adaptateur de réponse »* qui adapte le résultat selon les concepts de l'ontologie locale du système émetteur de la requête originale, et enfin transmet la réponse à ce système.

4.1.8- GéS – Générateur de service

Les besoins des acteurs de maintenance évoluent et font appel à des services ne correspondant à aucune application de la plateforme. Pour répondre à ces nouveaux besoins, le composant *GéS (générateur de services)* permet de générer des services à la demande.

Ces services créés sont en lien direct avec les concepts définis dans l'ontologie, et les relations associées. La définition de ces services laissée à la discrétion de l'utilisateur, est basée sur un calcul d'indicateurs et de seuils par rapport aux concepts de la plateforme. Cette définition sera traduite en requête spécifique qui fera le calcul et fournira les résultats associés au service.

Dans le cas où une requête demande un service que les applications existantes ne fournissent pas, et le service n'existe pas dans la bibliothèque de services SLIB, le *Coordinateur* dirige l'acteur vers une interface graphique interactive gérer par le *GéS*. Sur cette interface l'acteur sera guidé et orienté pour crée en ligne de service adapté à sa demande tout en respectant les droits d'accès accordés à son rôle (voir Figure 3-5). L'acteur de maintenance, utilisateur de la plateforme exprime sa demande en définissant un nouveau service en utilisant le TBOX du *BaC*

par le biais d'une interface graphique sous forme de listes déroulantes et de champs de textes définissant de manière semi-formelle le service demandé.

L'interface contient des blocs contenant les concepts, les instances et les relations de la base de connaissances de la plateforme. Ainsi, le bloc contient des champs permettant de mettre des opérateurs et des contraintes sur le concept à traiter. Il contient aussi un champ *libellé* qui permet à l'utilisateur de nommer un champ d'affichage lors de l'exécution. Un indice qui mentionne si le bloc est bien défini et que les opérations peuvent être effectuées, un indice mis en vert, en rouge dans le cas contraire. Chaque bloc est relié au bloc suivant ainsi qu'un bloc intermédiaire permettant d'indiquer le type de relation entre les blocs (ET, OU, UNION). L'interface permet à l'utilisateur de rajouter des blocs pour définir ses souhaits.

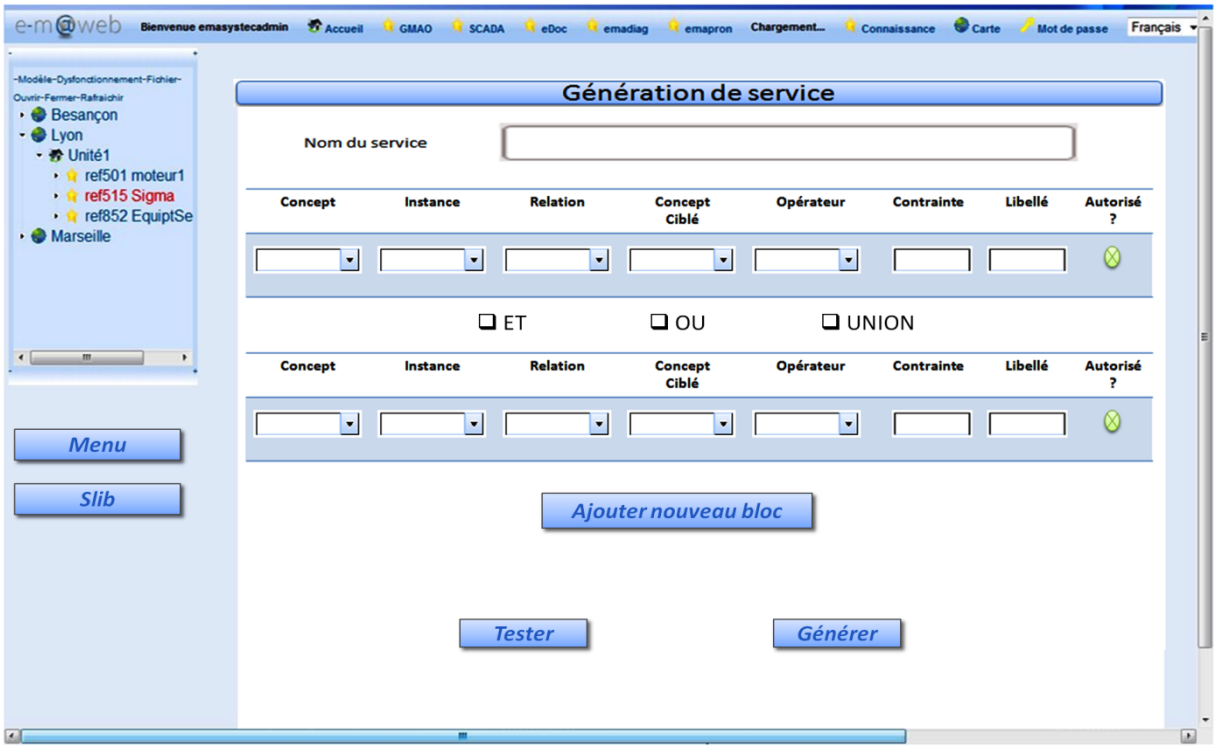


Figure 3-5 Interface de génération de service

Comme indiqué sur la Figure 3-6 illustrant le fonctionnement de ce composant ainsi que son architecture interne, au fur et à mesure avec la définition du service, le *GéS* reformule le service définit sous forme de requête formelle en langage de description PowerLoom pour qu'il soit exécuter via le moteur de raisonnement (Raisonneur).

A la fin de la formulation l'utilisateur teste le service défini (par un clic sur le bouton Tester). Une fenêtre pop-up contenant le résultat du service, si le résultat est satisfaisant aux attentes de l'utilisateur, celui-ci valide le service qui lui permettra de générer le service et le sauvegarder dans la librairie de service. Dans le cas contraire, l'utilisateur revient sur la formulation du service pour avoir ce qu'il attend comme retour. Un bouton de validation et un autre de modification sont dans l'interface sur la fenêtre pop-up du résultat.

Après avoir créé ce service sous forme de requête, le *GéS* enregistre ce nouveau service dans la librairie de service *(BiS)*.

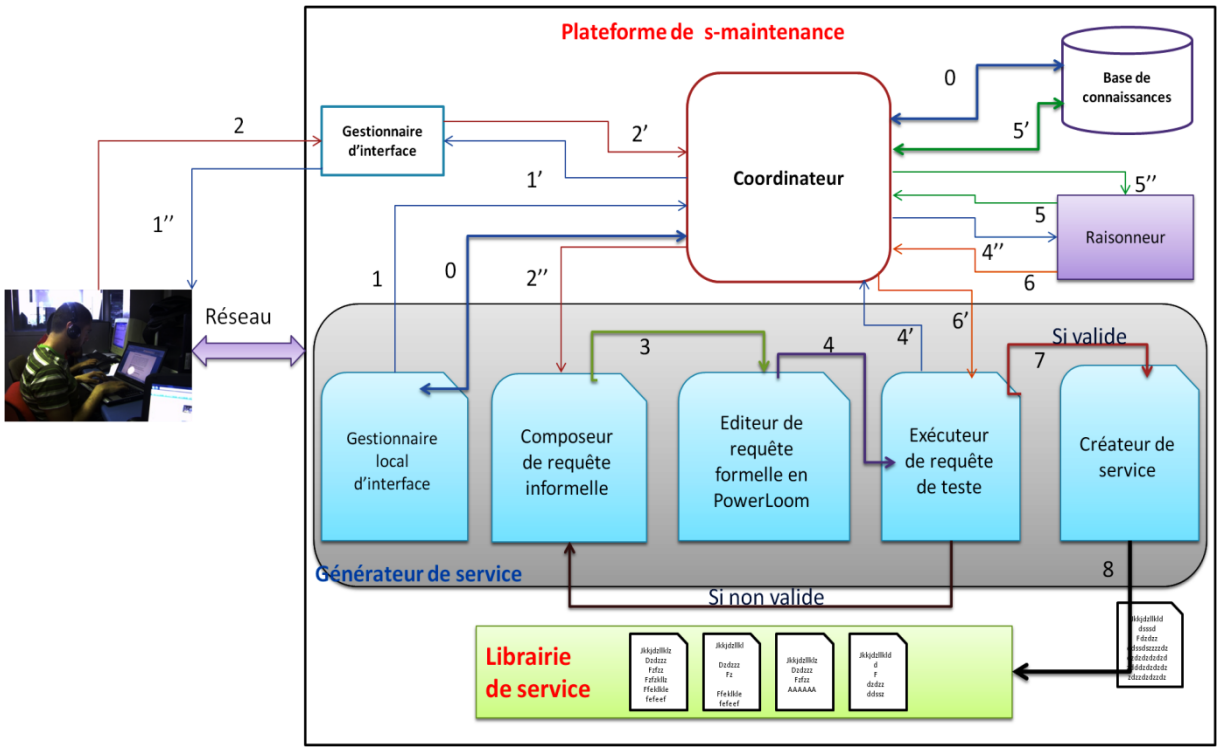


Figure 3-6 Architecture de génération de service

En effet, après avoir récupéré les concepts de l'ontologie dans la base de connaissance, à travers le module *«gestionnaire local d'interface»*, la plateforme fournie les paramètres de l'affichage au gestionnaire d'interfaces pour afficher l'interface à l'utilisateur et les concepts de la base de connaissances à utiliser.

L'interface graphique est gérée par le module *«Composeur de requête Informelle»* qui formule la requête selon les relations qui existent entre les concepts choisis par l'utilisateur. Une fois le service défini, et quand l'utilisateur demande le test, la description semi-formelle est donc envoyée au module *« Editeur de requête formelle en PowerLoom »* qui la traduit en une requête formelle définie avec PowerLoom. La requête est après transférée au module de test *«Exécuteur de requête de test»* qui envoie la requête au raisonneur pour qu'elle soit exécutée. Une fois validée, la requête est transférée au module *«Créateur de service »* qui crée le fichier contenant la description du service selon le XSD définie dans la librairie de service. Finalement le service est enregistré dans le *BiS* et il est prêt à l'exécution.

Finalement, nous notons que les applications intégrées n'ont pas la possibilité de créer de nouveaux services et d'interagir avec le *GéS* (tant qu'il est orienté utilisateur humain), mais elles peuvent exploiter les nouveaux services enregistrés dans le *BiS* en fournissant justes les paramètres d'entrés (s'il faut).

Cas d'illustration :

Dans cette section, nous présentons un exemple simple décrivant la définition d'un nouveau service. Le service souhaité par l'utilisateur est d'avoir les composants réutilisables lors du démontage de l'équipement SISTRE.

L'utilisateur défini un composant réutilisable comme un composant qui dispose d'une période de fonctionnement moins de 12000 heures et ayant moins de trois modes de fonctionnements de type panne.

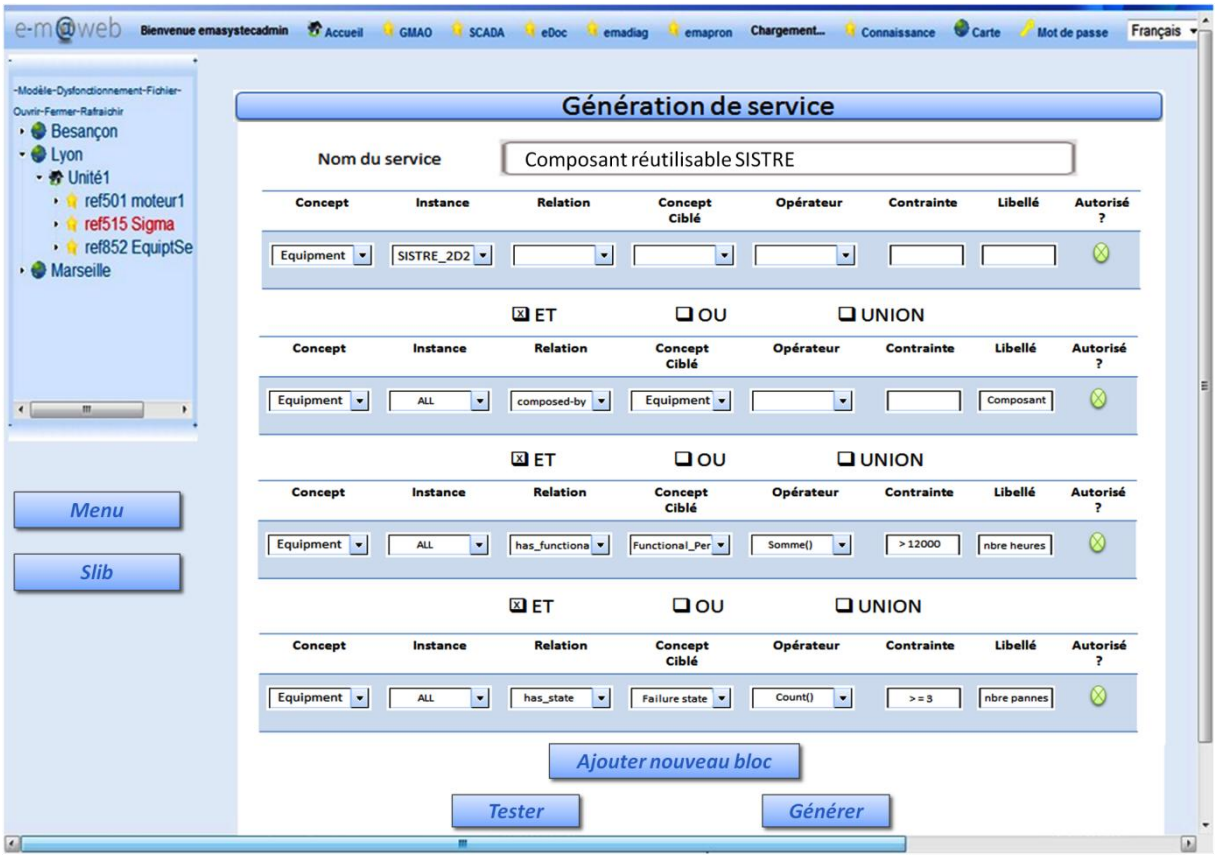


Figure 3-7 Définition d'un service par utilisateur

Les blocs définis de manière semi-formelle (Figure 3-7), sont traduits en arrière plan via le sous composant « Editeur de requête formelle en PowerLoom » en requête formelle exprimée en langage PowerLoom. La requête résultante est la suivante :

```
DEFRELATION REUSABLE-COMPONENT-OF ((?PE PHYSICAL-EQUIPMENT) (?COM PHYSICAL-EQUIPMENT))  :=>
(AND (
AND(
(HAS-EXPLOITATION-MODE ?COM ?EM)
AND ((HAS-FUNCTIONAL-PERIOD ?EM ?FP)
        (> (SUM (NUMBER-HOURS ?FP ?NH))60000)
)
                )
(AND (HAS-PERIOD ?EM ?P)
        AND ((DURING ?P ?OM)
                AND((TRUE(FAILURE-STATE ?OM))
                        (>(COUNT(STATE_ID ?OM ?X))3)
                )
        )
))
```

L'utilisateur clique sur le bouton Test qui lance l'exécution de cette requête et reçoit par conséquent la fenêtre pop-up suivante contenant le résultat de ce service (Figure 3-8) :

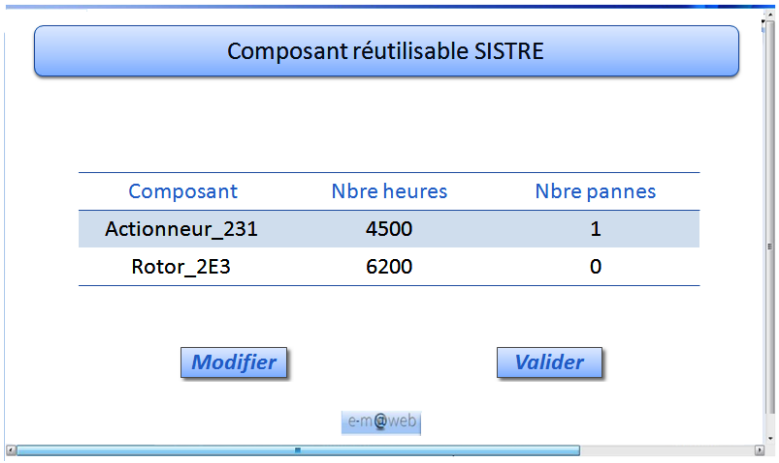


Figure 3-8 Fenêtre Pop-up du service composant réutilisable

Etant donné que le service répond aux attentes de l'utilisateur, ce dernier valide le service et clique sur le bouton qui se situe dans la fenêtre principale de l'interface de génération de service pour qu'il soit enregistré dans la librairie de service.

4.1.9- BiS – Librairie de service

Le BIS est le composant qui contient tous les services qui ne sont pas fournis par les applications intégrées dans la plateforme et qui sont générées de façon dynamique sous forme de requête (en langage de description).

Comme nous pouvons le constater sur la Figure 3-9, le BiS est composé d'un fichier XML contenant les descriptions de ces services, d'un parseur XML permettant de parcourir et extraire les données concernant les services enregistrés dans le fichier XML, et finalement d'un «lanceur de service» permettant d'envoyer la requête du service au moteur de raisonnement pour qu'elle soit exécutée.

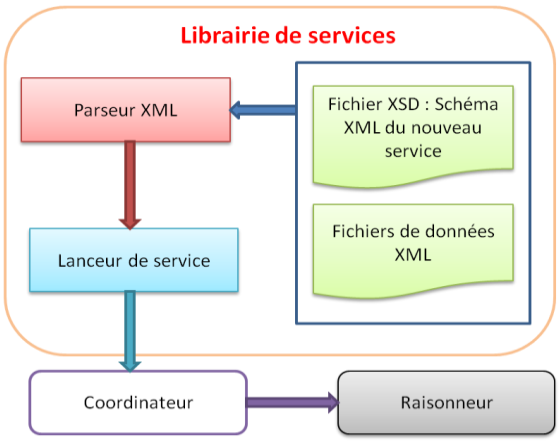


Figure 3-9 Architecture du *BIS*

La Figure 3-10 montre le fichier XSD contenant le *schéma XML* des nouveaux services. Un service enregistré dans la librairie de service est défini par, son nom, sa description, la requête exécutable concernée, le nom de l'acteur qu'il a défini ainsi que son rôle, la date et l'heure de sa création, la liste des concepts utilisés dans la

requêtes, la liste des relations définis dans la requêtes, la liste des contraintes ou conditions définies par l'acteur et finalement les inputs de la requêtes s'il y en a.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="GeneratedService">
  <xs:complexType>
   <xs:sequence>
            <xs:element name="ServiceName" type="xs:string"/>
            <xs:element name="ServiceDescription" type="xs:string"/>
            <xs:element name="ServiceQuery" type="xs:string"/>
            <xs:element name="ServiceDefiner" type="xs:string"/>
            <xs:element name="ServiceDefinerRole" type="xs:string"/>
            <xs:element name="ServiceCreationDate" type="xs:dateTime"/>
            <xs:element name="ServiceConcepts" type="xs:list"/>
            <xs:element name="ServiceRelations" type="xs:list"/>
            <xs:element name="ServiceConstraints" type="xs:list"/>
            <xs:element name="ServiceInputs" type="xs:list"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
```

Figure 3-10 XSD des services générés

4.1.10- GTM – Gestionnaire de traces modélisées

Pour respecter cette définition, nous avons proposé le composant *gestionnaire de traces modélisées –GTM* permet d'enregistrer dans la base de connaissances, de manière modélisée, les activités effectuées par la plateforme. Tant que toutes les communications et interactions dans la plateforme passent par le *Coordinateur*, le *GTM* récupère les fichiers logs enregistrés par le « traceur d'activités et de messages » et applique des opérations pour extraire des connaissances sous la forme de la structure de l'ontologie, pour cela nous l'appelons par traces modélisées (trace qui suit le modèle de l'ontologie).

Le but de cette traçabilité est de fournir les connaissances nécessaires pour que la plateforme assure la fonctionnalité d'auto-apprentissage qui lui donne la possibilité de faire évoluer ses connaissances et comprendre son fonctionnement. Cela est assuré par l'analyse de ces interactions tracées. Le développement de ce composant et les détails de son fonctionnement seront présentés dans le cinquième chapitre.

4.1.11- GeP - Gestionnaire de processus

Le composant *«GeP- gestionnaire de processus»* permet de gérer et d'orchestrer tous les processus qui s'exécutent dans la plateforme. Nous rappelons que l'orchestration exprime les conditions et les enchaînements des invocations aux processus (services). D'un point de vue de génie logiciel, le *GeP* joue aussi le rôle du contrôleur dans une architecture MVC[30] (Model-View-Controller) qui est chargé de faire appel aux règles métier adéquates à un événement.

[30] MVC est une architecture et une méthode de conception qui organise une application logicielle en séparant les objets applicatifs (Modèle), de leur représentation (Vue) et des interactions qu'ils subissent (Contrôleur). Pour plus de détails veuillez se référer à : http://msdn.microsoft.com/en-us/library/ff649643.aspx

Dans la plateforme, l'orchestration peut être une composition de processus. Une orchestration définit un workflow entre les processus. C'est-à-dire que le composant *GeP* est responsable de gérer l'ordre des processus et des activités exécutées dans la plateforme tout en respectant les contraintes métier et les conditions de transitions prédéfinies (après la fini du processus 1, on lance le processus 2, ensuite le processus 3 ...et enfin processus n.).

D'autre part, ce composant garantit la fonctionnalité d'autogestion des processus de maintenance en assurant l'exécution automatique de certaines activités de ces processus sans intervention humaine. Cette fonctionnalité sera aussi développée dans le cinquième chapitre de cette thèse.

4.2- Interaction des composants

4.2.1- Interactions utilisateurs/plateforme

Dans cette section, nous décrivons les interactions entre les utilisateurs et la plateforme s-maintenance. La Figure 3-11 illustre la communication entre ces utilisateurs la plateforme. Deux étapes se présentent:

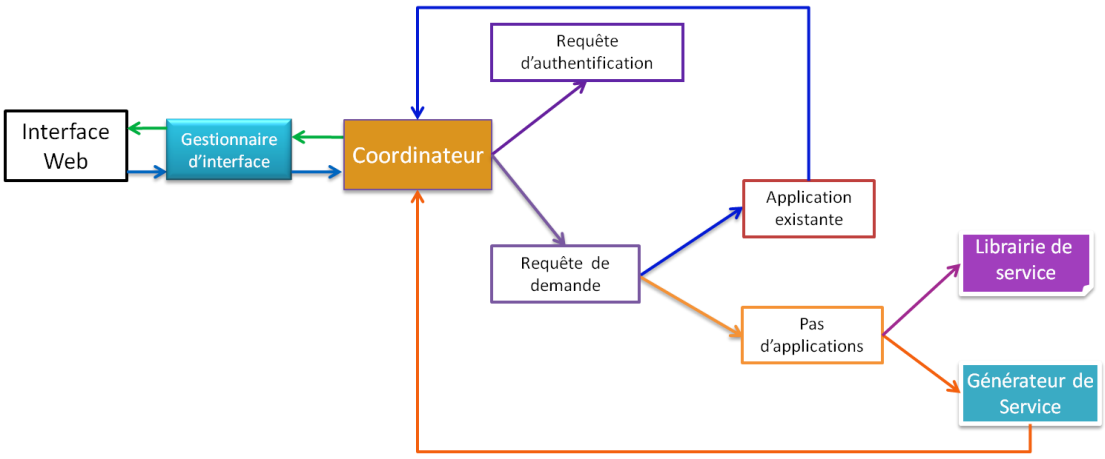


Figure 3-11 Interaction entre l'utilisateur avec la s-plateforme

La première étape est de *se connecter au système* (Requête d'authentification)

La deuxième étape est d'envoyer des requêtes de demande de services à la plateforme. Dans cette étape, il existe deux possibilités :

- Premièrement les services demandés sont fournits par les applications intégrées (composants génériques),
- Deuxièmement aucune application ne fournie ces services.

Plus de détails sur chacun des différents cas sera fourni dans les sections suivantes.

4.2.1.1- Requête d'authentification

Comme l'illustre la Figure 3-12, quand un utilisateur demande l'accès à la plateforme via l'interface web, cette requête passe par le *gestionnaire d'interface* et arrive au composant *Coordinateur*. Le *Coordinateur* envoie la

requête d'identification au composant *ConA* pour savoir si cet utilisateur peut accéder à la plateforme ou non ? Si oui, avec quel rôle ? Pour répondre à cette requête, le *ConA* a besoin d' informations concernant l'utilisateur, il demande donc au *Coordinateur* de les récupérer. Ensuite, le *Coordinateur* envoie la demande du *ConA* au *GBS* qui lui interroge la base de connaissances. Ce dernier répond au *GBS* qui renvoie les informations demandées au *Coordinateur* qui lui aussi les renvoie *ConA*. Ce dernier authentifie l'identité de l'utilisateur, c'est-à-dire une réponse avec le droit d'accès à la plateforme et le rôle de cet utilisateur (réponse d'identification à base de rôle) est envoyée au *Coordinateur* qui lui envoie le résultat d'authentification au *gestionnaire d'interface machine (GIM)*. En effet, le *GIM* affiche la réponse sur l'interface web (IW) : si l'utilisateur a le droit d'accéder à la plateforme, il se connecte. Par contre, dans le cas où l'utilisateur n'a pas le droit pour accéder à la plateforme, un message de «non autorisé » est affiché sur l'IW.

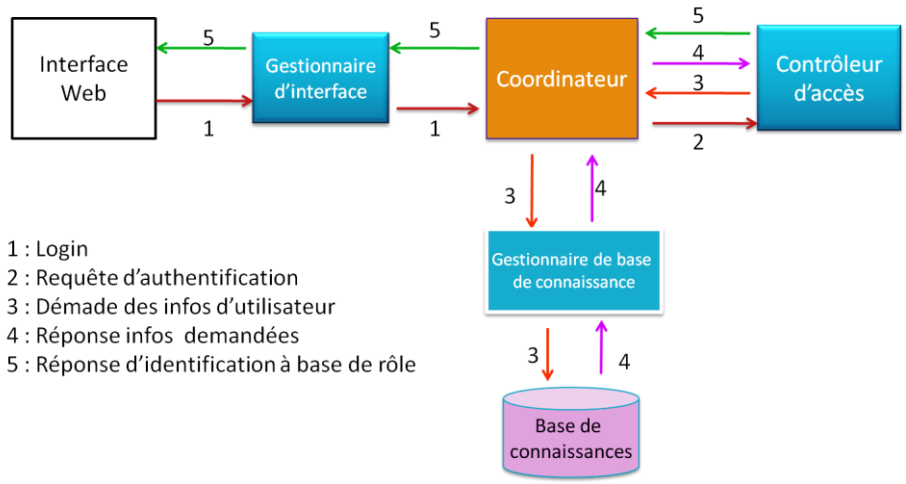


Figure 3-12 Interaction en cours d'une d'authentification

4.2.1.2- Requête de demande de services

Comme nous l'avons mentionné précédemment concernant l'authentification de connection à la plateforme, la plateforme peut être utilisée en fonction du rôle du demandeur, en manipulant et demandant l'exécution des services fournis par la plateforme (requête de demande des services).

Lorsque le *Coordinateur* reçoit une requête de demande de services fournis par la plateforme, deux cas possibles se présentent (Figure 3-13):

Ces services sont fournis par les applications intégrées. Dans ce cas le *Coordinateur* identifie les applications fournissant les services, et communique avec ces applications. Chaque application lance les services existants. En cas de besoin, l'application peut demander au *Coordinateur* de communiquer avec d'autres composants du cœur de la plateforme (exp : *Raisonneur*, *GeP*, *ConA*, *GBS*).

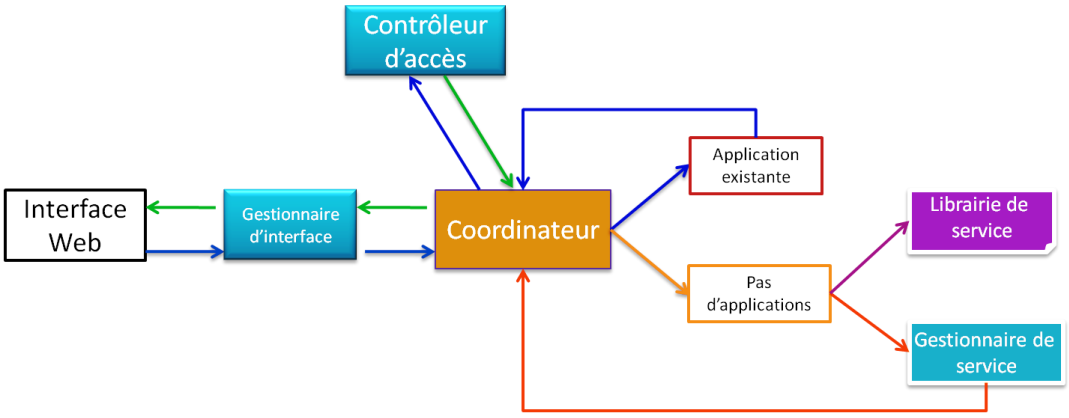


Figure 3-13 Interaction lorsque la requête de demande des services

Dans le deuxième cas où les services demandés ne sont pas fournis par les applications, le *Coordinateur* sollicite la bibliothèque de service, pour identifier si ces services existent. Si ce n'est pas le cas, le *Coordinateur* dirige l'utilisateur au *GéS (générateur de service)* pour définir ces services grâce à l'exploitation des connaissances existantes.

Si la bibliothèque *BIS* contient ces services, *Coordinateur* donne l'ordre d'exécution des services au *BIS* qui assure l'exécution via le sous composant *lanceur de service*.

4.2.2- Interactions entre composants au profit des fonctionnalités d'auto-x

Pour assurer les fonctionnalités caractérisant la plateforme de s-maintenance à savoir la traçabilité, l'auto-apprentissage et l'autogestion. Les composants impliqués dans ces fonctionnalités sont ; le coordinateur, la base de connaissances, le raisonneur, le gestionnaire de traces modélisées et le gestionnaire de processus (voir Figure 3.14).

En effet, le *GeP (gestionnaire de processus)* et le *Coordinateur* gèrent la conduite et la progression du processus de maintenance dans la plateforme tout en respectant les processus définis dans la base de connaissances. Le composant *GTM (gestionnaire de traces modélisées)* enregistre toutes les traces des interactions entre les utilisateurs et la plateforme et celles entre composants dans la base de connaissances.

Ainsi, après l'enregistrement de ces interactions, périodiquement, le raisonneur analyse ces connaissances pour en extraire de nouvelles. En d'autres termes, le raisonneur fait un auto-apprentissage afin de générer dynamiquement des nouvelles connaissances dans la plateforme. Les règles résultantes de cet apprentissage sont enregistrées dans la base de connaissances par le *raisonneur*. Finalement ces nouvelles connaissances peuvent être par conséquent exploitées soit par le *GeP* pour autogérer certains cas particuliers de processus soit par les autres composants génériques de la plateforme pour fournir des services dynamiques à l'utilisateur.

Nous notons que le développement de ces fonctionnalités sera intégré dans un système à base de traces que nous décrivons dans le chapitre 5 de cette thèse.

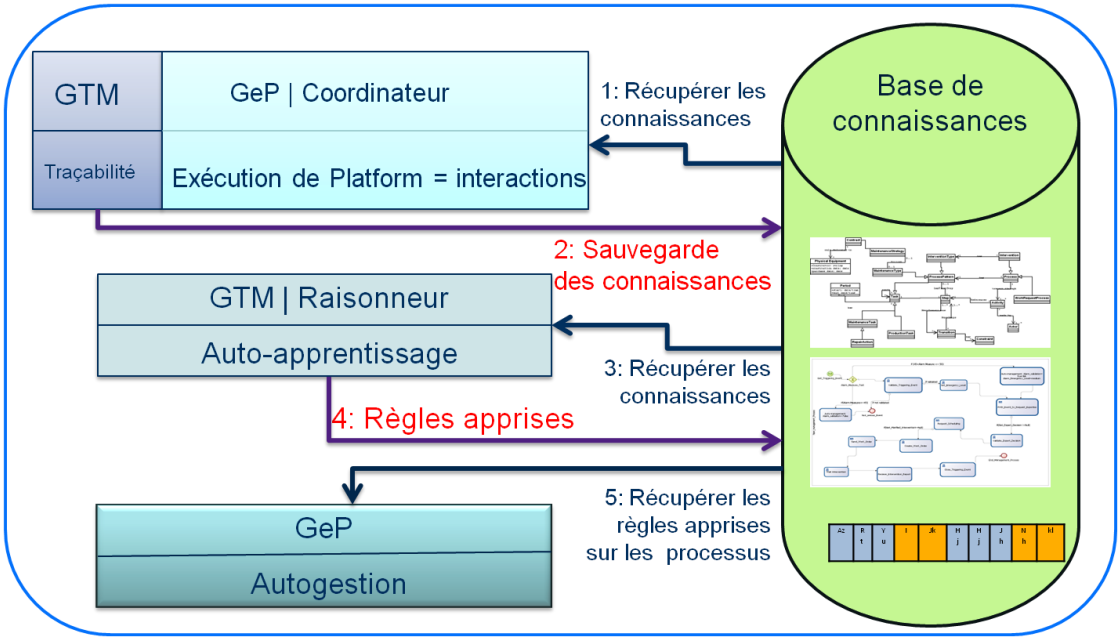


Figure 3-14 Interaction pour assurer les fonctionnalités de traçabilité, d'auto-apprentissage et d'autogestion

4.2.3- Illustration de l'exécution

Par ailleurs, pour illustrer l'exécution de la plateforme, nous présentons l'exemple d'un service d'autogestion d'un processus de maintenance fourni par la plateforme. Ce service s'interesse à un processus de maintenance conditionnelle suite à une alarme sans l'intervention d'opérateurs humains, à l'exception de la tâche d'intervention. Dans cet exemple, nous illustrons les interactions entre les composants de la plateforme tout au long duprocessus de monitoring assuré par le SCADA jusqu'à l'envoi de l'ordre de travail au opérateur de maintenance. L'exemple traite le cas d'une alarme générée par le SCADA concernant un composant x d'un équipement Y.

La Figure 3-14 montre le diagramme de séquence de ces interactions. Nous notons que le processus de maintenance conditionnelle est déjà instancié dans la base de connaissances et géré par le *GeP* sur l'illustration. Nous notons également que l'application de diagnostic utilisée dans l'illustration est basée sur le modèle de l'équipement qui identifie les causes des échecs et les actions à faire pour la réparation.

La liste suivante des messages présente l'ensemble des étiquettes des messages affichés dans la Figure 3-15 :

1 : alarme urgent.
2 : alarme urgent, quoi faire?
3 : faire une expertise.
4 : demande d'expertise (composant x, équipement Y)
5 : demande des données capteurs et les informations sur l'équipement Y.
6 : demander le modèle technique de l'équipement.
7 : demander les données capteurs.
8 : données capteurs
9 : demander le modèle technique de l'équipement.
10 : modèle technique de l'équipement.
11 : modèle technique de l'équipement.
12 : données capteurs
13 : modèle technique de l'équipement.
14 : besoin du modèle de comportement de l'équipement Y.

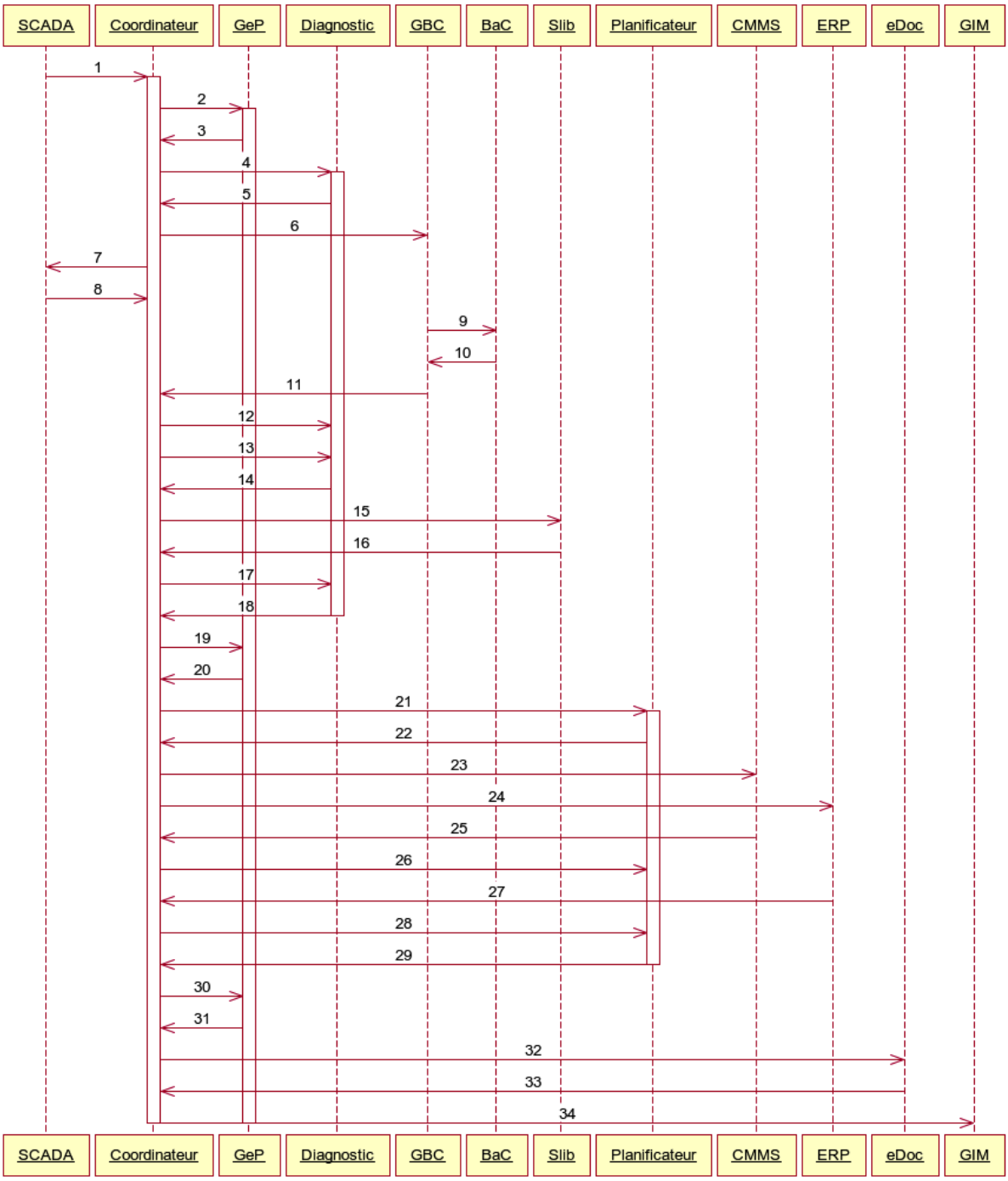


Figure 3-15 Interactions des composants lors de l'autogestion du processus de maintenance conditionnelle

15 : demander un service de la génération du modèle de comportement de l'équipement Y.

16 : modèle de comportement généré de l'équipement Y.

17 : modèle de comportement de l'équipement Y.

18 : rapport d'expertise (changer composant x avant 10 heures et le composant z qui pose le problème du composant x).

19 : quoi faire après l'expertise.

20 : planifier une intervention.

21 : intervention (changement des composants x et z avant 10 heures).

22 : demande (réservation pièces de rechange) et (disponibilité des ressources humaines).

23 : liste des pièces de rechange à réserver.

24 : demander la disponibilité des opérateurs de maintenance.
25 : confirmer la réservation des pièces de rechange.
26 : réservation des pièces de rechange confirmée.
27 : liste des opérateurs disponibles.
28 : liste des opérateurs.
29 : intervention planifiée.
30 : intervention planifiée, quoi faire?
31 : éditer l'ordre de travail.
32 : demander l'édition de l'ordre de travail (intervention planifiée).
33 : ordre de travail édité.
34 : envoyer l'ordre de travail aux opérateurs concernés (en nomination pour l'intervention).

5. Implémentation et discussions

5.1- Implémentation

Dans le cadre du projet SMAC et en partenariat avec l'entreprise em@systec, une première version de la plateforme de s-maintenance appelée em@web a été développée. em@web issue de recherche de notre équipe en partenariat avec em@systec, a été conçue comme plateforme de e-maintenance prenant appui sur un modèle de connaissance relatif à la maintenance. Nous faisons évoluer cette plateforme par l'ajout petit à petit de composants et de fonctionnalités associées. . Les principaux composants qui sont encours de développement sont le générateur de services, et le gestionnaire de traces modélisées.

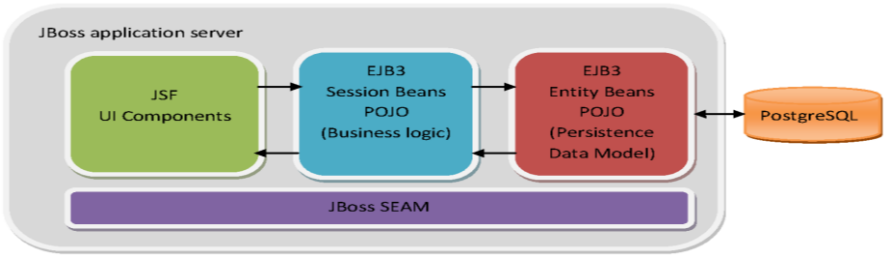


Figure 3-16 Architecture technologique de em@web

Les différents éléments développés dans la plateforme sont effectués sous le langage portable JEE (Java Entreprise Edition) qui facilite la création d'applications distribuées. Plus précisément, la mise en œuvre est réalisée grâce au serveur d'application JBoss qui est le plus couramment utilisé. Ce serveur d'application utilise le Framework Seam qui facilite le développement d'applications Web et l'intégration de JSF (Java Server Faces), EJB 3 (Enterprise Java Bean) et JPA (Java Persistence API). La Figure 3-16 montre les différents éléments technologiques d'em@web.

En effet, le *framework* d'interface utilisateur JSF est utilisé pour le développement du gestionnaire d'interfaces et l'interface web. L'EJB qui est un composant réutilisable conçu pour être déployé sur les serveurs d'applications est utilisé pour la mise en place du coordinateur et du gestionnaire de processus. JPA est une interface définissant un mapping entre les objets de la base de données. Elle est exploitée par le coordinateur et le gestionnaire de base de connaissances.

Actuellement em@web utilise une base de données PostgreSQL ayant un schéma de données respectant une partie du modèle ontologique de la base de connaissances

Par ailleurs, nous développons un module de liaison pour PowerLoom avec Java en nous inspirant de celui développé par (Watson, 2008) et illustré dans la Figure 3-17.

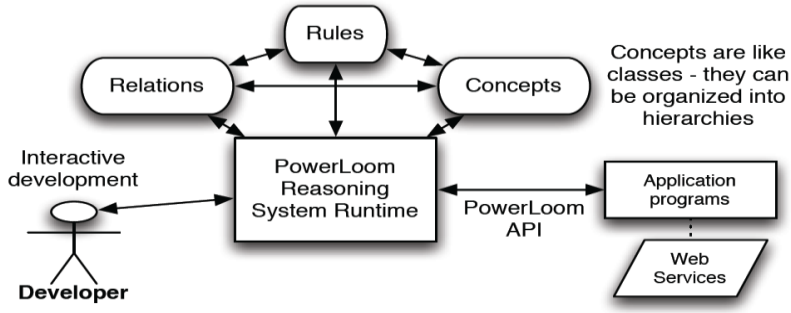


Figure 3-17 Overview of how we will use PowerLoom for development and deployment (Watson, 2008)

En ce qui concerne l'accessibilité à la plateforme, un client léger (Jae Yang, Nieh, Selsky, & Tiwari, 2002) a été utilisé, choix n'exigeant que l'installation d'un navigateur Web. En ce qui concerne la communication dans em@web, les communications avec les composants spécifiques du cœur de la plateforme sont assurées par EJB ; session fournissant un service clients via le protocole RMI (Remote Method Invocation). Par contre les communications avec les applications intégréessont assurées par la technologie des services Web via le protocole SOAP. De plus, les connections entre la plateforme et les smart technologies comme le RFID et les capteurs se font via le protocole de bas niveau Socket. D'un autre coté, un protocole de haut niveau a été fait lors de la mise en place d'un réseau de capteurs Zigbee[31] installé sur les équipements gérés par la plateforme. En ce qui concerne , l'interface homme machine, em@web intègre deux modules de réalité augmentée installés pour orinter l'affichage vers les besoins desopérateurs de maintenance lors l'intervention ou l'inspection. Lors d'une intervention, il suffit juste de passer le lecteur d'étiquette RFID sur l'équipement pour avoir accès aux informations concernant l'état de santé de l'équipement, sur un casque de réalité augmentée portée par l'opérateur tout en laissant les mains libre à ce dernier pour intervenir sur l'équipement comme le montre la Figure 3-18.

Ainsi, deux équipements sont gérés dans la plateforme, le premier est le Système Industriel Supervisé de TransfErt de palette SISTRE localisé dans le département AS2M et le deuxième est l'équipement « motobroche DX10 » de notre partenaire industriel Suisse TORNOS.
La Figure 3-19 présente quelques prises d'écran de em@web concernant la gestion des équipements, le calcul d'indicateurs, le monitoring, la gestion des ressources et la gestion des interventions.

[31] http://www.zigbee.org/

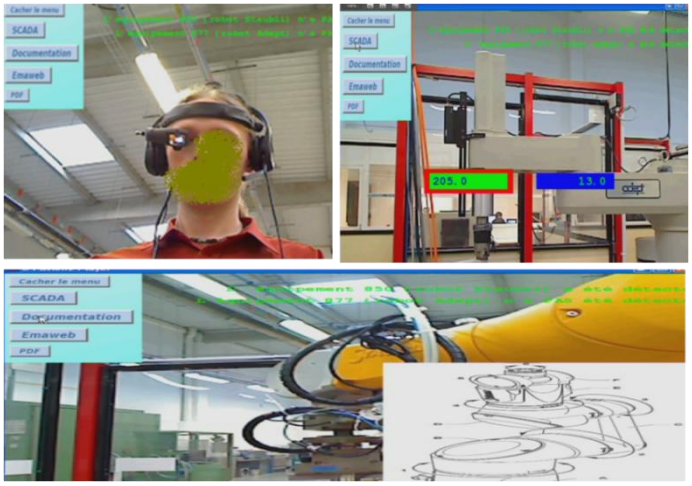


Figure 3-18 Prise d'écran d'une démo réalité augmenté sur em@web

5.2- Evaluation de la fiabilité de la plateforme

On doit tenir compte de l'équilibre entre la satisfaction des exigences par un composant et les risques engendrés par son mauvais fonctionnement ; donc lorsque nous mettons en place les composants nous devons prendre en considération ces risques et tout ce qui peut perturber le fonctionnement de la plateforme. Cet aspect de gestion de risque caractérise la persistance[32] et la consistance[33] de l'architecture de plateforme proposée.

Suivant la sollicitation du composant dans la plateforme et son importance (par rapport à ses fonctionnalités), nous devons garantir sa fiabilité afin de maintenir en mode opérationnelle la plateforme.

Les composants les plus critiques sont le coordinateur et la base de connaissance *(BaC)* car ils sont systématiquement sollicités lors du fonctionnement de la plateforme.

En ce qui concerne la base de connaissances, un crash dans la base de connaissance peut engendrer la perte de toutes les connaissances de la plateforme ou bloquer tous services fournis sachant qu'ils exploitent toutes les connaissances de cette base. La même chose peut être notée en ce qui concerne le gestionnaire de base de connaissances *(GBS)* qui prend en charge la lecture et l'écriture dans la base de connaissances. Pour cela nous proposons la mise en place de copies de secours de ces deux composants qui seront utilisées dans le cas de crash.

De plus, lors de l'implémentation, nous avons mis en place une base de données *PostgreSQL* pour que le fonctionnement de la plateforme ne soit pas bloqué si un crash est engendré au niveau du *GBS* ou du *BaC*, la base de données sera utilisée pour fournir les services qui ne nécessitent pas un raisonnement sur la base de connaissances.

De plus, dans le même esprit le sous composant *cache pour requêtes* dans le *coordinateur* servira pour récupérer les connaissances qui ne sont pas encore enregistrées ou en cas de crash du *GBS*, et de mettre à jour la base après recouvrement de ce composant.

[32] Se réfère au mécanisme responsable de la sauvegarde et la restauration de données, afin qu'un programme puisse se terminer sans que ses données ni son état d'exécution soient perdus.
[33] Se réfère à la cohérence, la stabilité et la solidité de construction.

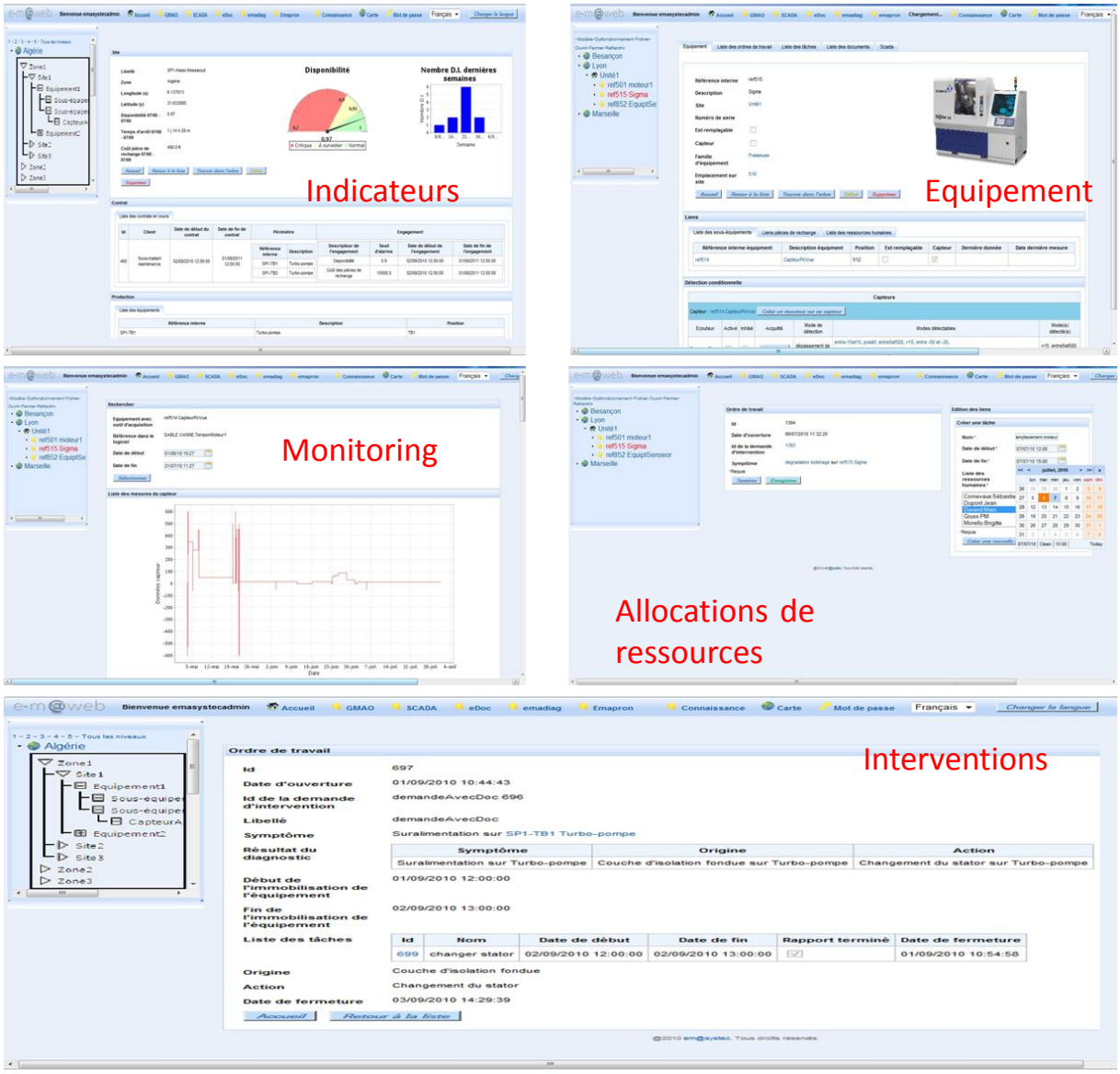


Figure 3-19 Prises d'écran de la plateforme em@web

D'autre part, concernant le Coordinateur, nous supposons qu'un volume important de requêtes envoyées peut surcharger la pile de message de ce composant, ce qui pourrait générer une augmentation en temps de réponse pour l'exécution des requêtes ou même un crash sur cette composante. Un échec sur le coordinateur signifie un échec total de la plateforme. Comme illustré sur la Figure 3-20, la charge de ce composant dépend du nombre d'équipements et de processus à gérer. Dans le cas d'un grand parc de machine ou d'une catastrophe, un grand nombre de défaillances simultanées induira une surcharge du coordinateur, susceptible d'engendrer un crash.

Pour limiter et contrôler la surcharge de ce composant nous avons mis en place la méthode de réplication maître/esclave dans le cœur du coordinateur (Charron-Bost, Pedone, & Schiper, 2010).

Quand le coordinateur note une augmentation de 30 % par rapport à la charge moyenne de messages transférés paramétrés par l'administrateur de la plateforme, nous proposons que les sous composants esclaves du coordinateur soient lancés. Ce qui amène un fonctionnement parallele entre les composants maîtres et esclaves qui se partagent l'empilement et le traitement (c.à.d. identifier les destinataires et dispatcher les messages) des

messages reçus. Ce fonctionnement parallèle ne s'arrête que lorsque la charge moyenne des messages reçue via le canal de transmission retombe à une valeur supérieure seulement à 10 % de la charge moyenne messages transmis déjà définie.

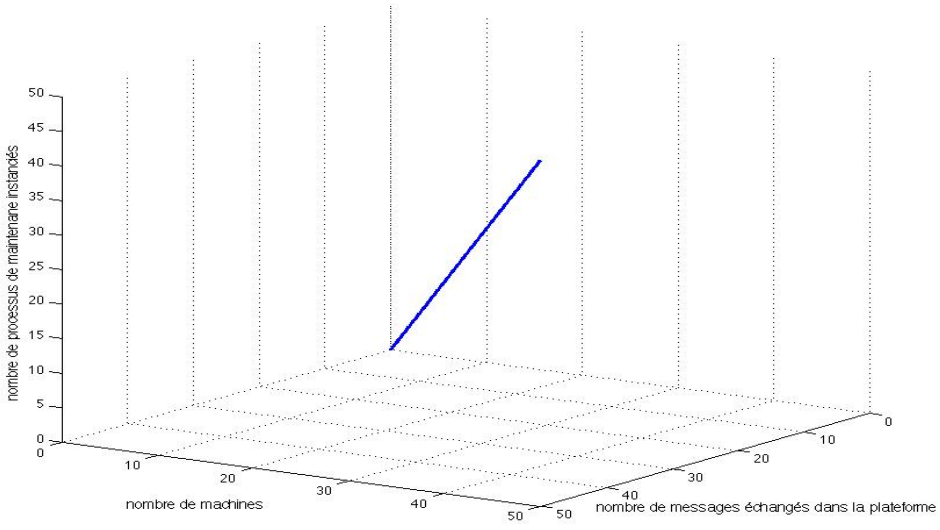


Figure 3-20 Évolution de nombres de messages échangés dans la plateforme

Ainsi, le sous composant *« Traceur d'activités et de messages »* peut servir aussi à la récupération des messages et des activités perdus en cas de crash d'une application ou d'un composant encours d'exécution et les relancer après le recouvrement du composant.

De façon générale, il est à noter que les risques de défaillances dans la plateforme ne résident pas que dans les deux composants critiques identifiés ci-dessus, mais aussi dans tout composant de la plateforme. D'où, l'apparition de différents problèmes pouvant survenir dans le fonctionnement de la plateforme en cas d'échec de l'un des composants. Tels problèmes peuvent causer une perturbation dans le fonctionnement des processus et le *« workflow »* de maintenance. Pour, cela une copie de secours de chaque composant peut être très utile en cas de dysfonctionnement d'un composant. Ces copies peuvent être répliquées dans un serveur distant.

Ainsi, des technologies et des mécanismes d'informatique autonome et d'auto-guérison (appelés respectivement *autonomic computing* et *self-healing* en anglais) (Horn, 2001) peuvent être exploitées dans la mise en place des composants de la plateforme, afin de remédier à ce risque et d'améliorer la robustesse de la plateforme.

6. Conclusion

Le but de ce chapitre était d'apporté des éléments de réponse à l'objectif suivant : *« Comment développer une architecture générique et agile capable de répondre aux caractéristiques d'une plateforme de s- maintenance générique ?».*

Nous avons choisi l'architecture à base de composant *(ABC)* qui est une branche du génie logiciel insistant sur la séparation des préoccupations en ce qui concerne les fonctionnalités du système. Ce style d'architecture se distingue par sa focalisation sur la structure du système faiblement couplé (loosly coopled) et par son niveau d'abstraction élevé, par la décomposition de la conception en différents composants fonctionnels et logiques indépendants, fortement interopérables, extensibles, réutilisables et remplaçables. Ainsi, les avantages de ce type d'architecture permettent de bien définir l'architecture par rapport aux caractéristiques de la s-maintenance.

La première étape de définition de cette architecture est l'identification des composants. Chaque composant dans cette architecture a été choisie pour répondre à une ou plusieurs des exigences, comme l'interopérabilité sémantique, l'inférence des connaissances, la génération de services à la demande, les fonctionnalités d'auto-apprentissage et d'autogestion, etc. Il convient également de noter que les composants de cette architecture sont définies par leurs fonctionnalités et non pas par leurs spécifications techniques.

Nous avons identifié deux types de composants, à savoir les composants génériques qui sont généralement communs dans les plateformes de maintenance (diagnostic, planification, etc.), et les composants spécifiques qui répondent aux exigences (caractéristiques) de la s-maintenance.

Avec un mapping composant/fonctionnalités, les composants clés de la plateforme que nous avons identifiés sont : Le coordinateur pour la gestion de la collaboration ; La base de connaissance et gestionnaire de la base de connaissances pour le partage et la réutilisation des connaissances ; Le raisonneur pour le raisonnement sur la base de connaissances et assurer la fonctionnalité d'auto-apprentissage ; Le gestionnaire de processus pour orchestrer les processus gérés par la plateforme et assurer la fonctionnalité d'autogestion ; Le gestionnaire de traces modélisées assurant la fonctionnalité d'auto-traçabilité ; Le générateur de services pour la gestion des services à la demande répondant aux nouveaux besoins des utilisateurs ; Finalement, le médiateur sémantique pour assurer l'interopérabilité sémantique.

En effet, une étude des différentes approches d'interopérabilité sémantique nous a permis d'adopter une approche de médiation basée sur 'une ontologie.

Après avoir défini les composants, leurs processus de fonctionnements, et l'architecture interne de quelque uns, nous nous sommes intéressés aux interactions dans la plateforme, que ce soit entre composants ou entre utilisateurs et plateforme.

Ainsi, après la présentation des implémentations faites de l'architecture proposée, nous avons discuté la fiabilité de cette architecture. Cette discussion nous a permis d'identifier les deux composants les plus critiques dans l'architecture qui sont le coordinateur et la base de connaissances vu les graves conséquences que peut engendrer un crash dans la base de connaissances ou le risque de plantage dans le coordinateur. Ces risques nous ont mené

à proposer la duplication des composants critiques et de faire appel à l'approche maître/esclave pour mettre une politique de fonctionnement dépendant du seuil d'interactions dans la plateforme pour assurer la durabilité et l'efficacité du coordinateur.

La validation de cette architecture et la mise en clair de ces valeurs ajoutées concrètes pour l'utilisateur seront fournies dans les chapitres suivants après avoir développé le composant clé de cette architecture qui est la base de connaissances qui fera l'objet du chapitre qui suit.

Chapitre 4 Proposition d'une ontologie de domaine pour la maintenance

1. Introduction

Un des points clés de la plateforme de s-maintenance est sa construction à partir d'un système à base de connaissance (SBC). L'architecture de la plateforme fonde son fonctionnement sur sa base de connaissances et son système de gestion.

Pour obtenir un SBC de qualité, la solution de choix est d'utiliser des ontologies ; solution permettant une réutilisation optimale des composants génériques que ce soit au niveau de raisonnement que des connaissances, sachant que l'ontologie sert de représentation des connaissances du domaine (Mizoguchi R. , 2004).

Une ontologie, qui est un système de concepts fondamentaux explique la conceptualisation du monde réel concerné et fournit une base solide sur laquelle des bases de connaissances partageables peuvent être construites pour une plus large utilisation que celle d'une base de connaissances traditionnelles (Charlet, Bachimont, & Troncy, 2004).

Le SBC dans la plateforme de s-maintenance a un rôle considérable et prend appui sur l'ontologie du domaine de maintenance qui est très importante. Cette ontologie sera exploitée aussi bien par le médiateur sémantique pour assurer l'interopérabilité sémantique partagée et réutilisée par les différents composants qui génèrent les services dynamiques fournis par la plateforme.

La section 2 sera consacrée à présenter le rôle d'uneontologie dans la plateforme de maintenance. Il est crucial que l'ontologie soit de qualité pour être exploitée au mieux par la plateforme et ne pas apporter un biais aux services proposés par celle-ci. Quelques travaux sur l'ontologie du domaine de maintenance ont été initiés par le passé. En effet on constate la présence de modèles standards dans le domaine comme MIMOSA-CRIS ou SOM, et un début d'ontologie ne couvrant pas tous les aspects du domaine de maintenance.

Par conséquence, pour remédier à ce manque, nous proposons dans ce chapitre de construire une ontologie du domaine de maintenance couvrant les différentes vues de ce domaine (par exemple : l'équipement, les ressources, les stratégies, etc) que nous appelons IMAMO (Industrial MAintenance Management Ontolgie).

La création d'une ontologie n'est pas une tâche triviale, elle doit suivre une méthodologie de construction et doit déterminer les outils et langages adéquats à utiliser. En effet, après un tour d'horizon sur les méthodologies de création d'ontologie décrit à la section 3, notre choix s'est porté sur la méthodologie METHONTOLOGY (Fernández-López, Gómez-Pérez, & Juristo, 1997) qui présente une stabilité plus grande par rapport aux autres méthodologies. METHONTOLOGY a été proposée comme une méthode structurée de construction d'ontologies. Elle comprend un ensemble d'activités qui couvrent le cycle de vie global des ontologies. Cette méthodologie décompose ces activités en trois niveaux : activités de gestion, activités de développement et activités de soutien.

Dans la quatrième section, nous présentons les différentes activités de METHONTOLOGY pour la construction, de IMAMO partant de l'acquisition des connaissances à la maintenance en passant par la conceptualisation, l'implémentation et l'évaluation.

L'activité d'évaluation classée comme activité de support dans cette méthodologie, a été privilégiée par l'application de différentes méthodes d'évaluation (utilisation des métriques, évaluation métier, etc.) sur les résultats des activités de développement, dans le but d'obtenir une ontologie de qualité.

D'autre part, concernant les langages de représentations des ontologies, une analyse de la littérature sur les langages et leur comparaison (avantages et inconvénients) nous a conduit à choisir le langage de description PowerLoom pour implémenter IMAMO via l'API java dédiée à ce langage.

2. Le rôle de l'ontologie dans la s-maintenance

Les ontologies ont une terminologie bien définie dont la sémantique n'est pas ambigüe (Guarino, 1998) grâce à une représentation formelle et explicite d'une compréhension commune des concepts du domaine et des relations entre ces concepts.

Dans l'approche basée sur l'ontologie, les significations des terminologies proposées et les propriétés logiques des relations sont spécifiées par des définitions et des axiomes ontologiques dans un langage formel.

En effet, une ontologie fournit (Mizogouchi & Bourdeau, 2004) :
1) une structure conceptuelle de base à partir de laquelle il est possible de développer des systèmes à base de connaissances qui soient partageables et réutilisables,

2) une interopérabilité entre les sources d'informations et de connaissances.

L'ingénierie ontologique succède à l'ingénierie des connaissances, et l'on s'attend à ce qu'elle devienne une technologie-clé pour la prochaine génération des technologies de traitement des connaissances.

Par rapport à notre contexte qui porte sur un domaine particulier, nous nous concentrons sur une ontologie de domaine pour la maintenance. La spécificité de cette ontologie est la réutilisation des connaissances et leur enrichissement au cours du cycle de vie des équipements à maintenir. Ainsi, le modèle d'ontologie devrait contenir des définitions de tous les objets de l'application dans ce domaine (par exemple : diagnostic ou documentation), ainsi que les contraintes et les relations entre les objets et les organiser.

Ce partage des connaissances par le biais de l'ontologie permet à chaque application intégrée dans la plateforme d'exploiter ces connaissances en parallèle avec ses connaissances internes et spécifiques. En outre, les méthodes de raisonnement qui peuvent être appliquées à l'ontologie lui fournissent une valeur ajoutée grâce à leur capacité de déduire de nouvelles connaissances. La plateforme offrira donc à ses acteurs ces valeurs ajoutées, toujours dans le but de fournir la bonne information au bon format pour les bonnes personnes pour faire les bonnes choses au bon moment grâce à l'interopérabilité sémantique entre les applications et les acteurs ainsi que l'exploitation et la réutilisation des connaissances partagées. Cette ontologie est une partie prenante du composant *« Base de connaissances »* de la plateforme de maintenance proposée.

L'exploitation de cette ontologie permet à la plateforme de comprendre et faire évoluer son comportement par l'inférence de nouvelles connaissances.

3. Construction des ontologies et approche adoptée

Pour construire une ontologie, (Corcho, Fernandez-Lopez, & Gomez-Perez, 2003) se sont questionnés sur la méthodologie la mieux adaptée, les outils et les langages à utiliser dans le processus de développement de celle-ci. Cette section est consacrée à répondre à ces questions et avoir un choix éclairé quant à la méthodologie à employer pour la construction de notre ontologie.

3.1-Langages, outils et méthodologies pour la construction d'ontologie

3.1.1- Méthodologies de développement

Les méthodologies de développement d'ontologies fournissent un support pout la création d'ontologies. Ainsi, Fernandez et al., dans (Fernández-López, Gómez-Pérez, & Juristo, 1997) (Gómez-Pérez, 1996), affirment que le processus de développement d'une ontologie se réfère à des activités nécessaires pour construire des ontologies. Plusieurs méthodologies comme TOVE, METHONTOLOGY, On-To-Knowledge, AFM, OntoClean, DILIGENT, NeOn, etc. ont été développées (Mizoguchi R. , 2004) (Corcho, Fernandez-Lopez, & Gomez-Perez, 2003).

- La méthodologie AFM est dédiée au développement des ontologies de tâches (Mizoguchi R. , 2004) . Ainsi, elle démarre le processus de construction après avoir déterminé le document source à partir duquel l'ontologie est extraite.
- La méthodologie de « Ushold et King » est utile pour développer une ontologie informelle à la phase précoce de développement.
- La méthodologie TOVE, est une des plus formelles. Elle énumère les questions auxquelles doit répondre l'ontologie à créer, qu'elle exprime dans un langage formel et qu'elle réutilise en phase de vérification.
- La méthodologie On-To-Knowledge fonctionne en analogie avec le fonctionnement des applications de gestion des connaissances (Mizoguchi R. , 2004) en comprenant les lignes directrices pour l'identification des concepts de l'ontologie à base de la gestion des connaissances par l'inspiration des questions de compétence pour déterminer la portée de l'ontologie. Le but de On-To-Knowledge est d'appliquer des ontologies à l'information électronique pour améliorer la qualité de la gestion des connaissances dans les organisations larges et distribuées. Cependant, la méthodologie n'est pas destinée aux développeurs de logiciels et aux praticiens d'ontologie.
- La méthodologie METHONTOLOGY est basée sur les activités menant à la construction d'une ontologie. Cet ensemble est basé sur les principales activités identifiées par le processus de développement logiciel et utilisées dans les méthodologies d'ingénierie des connaissances. Cette méthodologie comprend: l'identification du processus de développement d'une ontologie, un cycle de vie basé sur des prototypes d'évolution, et des techniques pour mener à bien chaque activité dans les activités de gestion, de développement et de support (soutien).
- La méthodologie OntoClean est orientée validation de taxonomies. Elle est basée sur de très générales notions ontologiques tirées de la philosophie pour caractériser les aspects pertinents de la signification voulue des propriétés des composants de l'ontologie. Ces aspects sont représentés par méta-propriétés

formelles qui imposent plusieurs contraintes sur la structure taxinomique d'une ontologie afin d'évaluer et de valider les choix effectués.

- La méthodologie DILIGENT est destinée à aider les experts de domaine dans un environnement distribué et de faire évoluer les ontologies, mais elle ne cible pas les développeurs de logiciels et de praticiens des ontologies. Cette méthodologie est centrée sur l'ingénierie ontologique de collaboration. Le centre de cette méthodologie est un système d'argumentation qui facilite les discussions sur la logique de conception concernant des modifications qui sont introduites dans les différentes phases du cycle de vie de l'ontologie.
- La méthodologie NeOn est créée dans le cadre du projet NeOn pour construire des réseaux d'ontologies[34]. Concrètement, en matière de dimensions NeOn, la méthodologie comprendra les prestations prévues par DILIGENT concernant la collaboration. Par ailleurs, NeOn prend en compte la proposition donnée par METHONTOLOGY et On-To-Knowledge concernant l'utilisation de questions de compétence pour l'activité de spécification de l'ontologie. En ce qui concerne la réutilisation des ontologies, NeOn considère comme point de départ la liste des activités proposées par METHONTOLOGY et propose des lignes directrices pour les améliorer et les étendre. Pour la construction de ces réseaux d'ontologies NeOn propose différents scénarios (neuf scénarios) par rapport à la méthode de la construction adoptée.
- NeOn est une combinaison de méthodes et peut être considérée comme l'évolution et l'extension de METHONTOLOGY prenant en considération plus en détails les activités ainsi que la prise en compte de la collaboration et du contexte.

Ainsi, Mizouguchi affirme que lors du développement d'une ontologie à grande échelle, METHONTOLOGY et On-To-Knowledge sont très utiles. Ainsi, Corcho et al., ont conclu que l'approche la plus mature est METHONTOLOGY (Corcho, Fernandez-Lopez, & Gomez-Perez, 2003).

Dans notre cas présent, nous ne développons pas de réseau d'ontologie, ni d'ontologie de tâches, pas plus de taxonomie, mais une ontologie unique[35] (*single ontology*) du domaine de maintenance. Nous adaptons ainsi la méthodologie METHONTOLOGY pour développer l'ontologie IMAMO.

3.1.2- Outils d'implémentation d'ontologie

Les outils d'implémentation ont augmenté de façon exponentielle ces dernières années. Ils sont destinés à soutenir le processus de développement d'une ontologie et son utilisation ultérieure. Comme exemples de ces outils, nous trouvons Ontolingua, WebOnto, Onto Edition, Protégé2000, le Protégé 3.4.5 et 4.1 Protégé soutenant respectivement OWL.1.0 et OWL.2.0, PowerLoom API and PowrLOOM GUI, TopBraid Composer[36], NeOn Toolkit[37], etc (Mizoguchi R. , 2004) (Corcho, Fernandez-Lopez, & Gomez-Perez, 2003). Par ailleurs, le choix de l'outil dépend du langage choisi pour l'implémentation de l'ontologie.

[34] Un réseau d'ontologies est une collection ontologies uniques interconnectées liés les uns aux autres via une variété de différentes métarelations.
[35] Une ontologie unique est une ontologie qui n'a aucun type de relation (domaine dépendant ou indépendant) avec d'autres ontologies.
[36] http://www.topquadrant.com
[37] http://neon-toolkit.org/wiki/Main_Page

3.1.3- Langages d'ontologie

Divers langages d'implémentation (ou de description) d'ontologies ont été proposés (Mizoguchi R. , 2004) (Corcho, Fernandez-Lopez, & Gomez-Perez, 2003). Ils peuvent être classés en deux catégories selon leur ordre chronologique d'apparition, avant et après le boom du web.

Au début des années 1990, un ensemble de langages d'implémentation d'ontologies basés sur l'intelligence artificielle a vu le jour. Fondamentalement, le paradigme de représentation de connaissances sous-jacentes est basé soit sur la logique du premier ordre (ex.KIF) soit sur des *« frames »* combinés avec la logique du premier ordre (ex. Ontolingua, OCML et FLogic), ou bien sur la logique de description (ex. Loom).

Avec la montée d'internet, plusieurs langages d'ontologie basés sur le web [web based] ont été créées afin d'exploiter les caractéristiques de la toile. Parmi ces langages, nous trouvons SHOE, RDF(S): OIL, DAML+OIL et OWL.

Un langage d'implémentation d'ontologie se caractérise par son expressivité, sa capacité de définition et de descriptions, l'existence d'un langage d'interrogation associé, et d'un outil d'implémentation ainsi que des outils de raisonnement.

Toutefois, quelques différences existent dans les primitives disponibles dans chaque langage pour la représentation des taxonomies des concepts. En ce sens, Ontolingua, Loom, OCML, OIL, DAML-OIL et OWL sont les plus expressifs, car ils permettent la création des sous-classes exhaustives et disjointes d'un concept (Corcho, Fernandez-Lopez, & Gomez-Perez, 2003). En outre, les fonctions peuvent être définies facilement dans cet ensemble de langages. Par contre, les règles ne peuvent être définies que dans Loom, OCML et OWL 2.0. Quant aux procédures, elles ne peuvent être définies que dans Ontolingua (bien qu'elles ne puissent pas être exécutées), Loom et OCML. Dans Loom et OWL.20, le moteur d'inférence effectue également des classifications de concept automatique (Corcho, Fernandez-Lopez, & Gomez-Perez, 2003).

PowerLoom, le successeur de Loom, fournit un langage et un environnement de construction intelligente, basée sur la connaissance des applications. Il utilise un langage de représentation tout expressif basé sur la logique (une variante de KIF) et un moteur d'inférence de déduction naturelle. Ce moteur associe le chaînage avant et arrière pour dériver ce qui suit logiquement les faits et les règles instanciés dans la base de connaissances. PowerLoom n'est pas une logique de description, par contre il contient un classificateur de description qui utilise une technologie dérivée du classificateur de Loom pour classer les descriptions exprimées en prédicats de premier ordre (PowerLoom, 2011).

Choix du langage :

Par conséquent, pour la mise en œuvre de IMAMO, nous avons choisi PowerLoom (Chalupsky, MacGregor, & Russ, 2010), parce qu'il offre comme facilités de raisonnement pour les concepts et les individus (c.à.d. les instances). Les deux caractéristiques de base qui le distinguent des autres systèmes de logique de description sont l'incorporation d'un langage d'interrogation expressive pour la récupération des individus, et son soutien à la programmation à base de règles. L'expressivité de PowerLoom fournit une évolutivité à grande échelle des ontologies et de bases de connaissances. Il fournit différents mécanismes de raisonnement, tels que la déduction

logique, le raisonnement hypothétique, le raisonnement égalitaire, le raisonnement arithmétique et le raisonnement des inégalités. Ceci permettra de vérifier la cohérence du modèle et des règles de notre ontologie, et aussi concevoir des améliorations de manière directe.

Par ailleurs, PowerLoom a un optimiseur de requêtes statiques et dynamiques qui est similaire à ceux utilisés dans les systèmes de bases de données. L'optimiseur dynamique fonctionne pour chaque sous-objectif conjonctif basé sur les liaisons (relations) réelles. Compte tenu de ce mécanisme, il est possible d'exécuter des requêtes PowerLoom retournant des centaines de milliers de solutions. PowerLoom possède également une interface puissante de base de données relationnelle qui lui permet de les utiliser pour le traitement de grandes bases d'instances (Chalupsky, MacGregor, & Russ, 2010). La plateforme de s-maintenance pourra donc profiter de ces performances de PowerLoom pour assurer l'interopérabilité sémantique et fournir de nouveaux services répondant aux besoins des acteurs de la maintenance. Les composants (génériques et spécifiques) de la plateforme sont censés bénéficier des capacités du moteur de raisonnement PowerLoom.

3.2- Méthodologie adoptée pour IMAMO

Nous avons ainsi adopté l'approche METHONTOLOGY pour construire l'ontologie du domaine pour la maintenance industrielle que nous appelons IMAMO (Industrial MAintenance Management Ontologie).

Dans ce contexte, METHONTOLOGY a été proposée comme une méthode structurée pour construire des ontologies. Son objectif est de couvrir le cycle de vie global des ontologies. Elle comprend un ensemble d'activités qui seront accomplies durant le processus de développement d'une ontologie. Comme le montre la Figure 1, il y a trois niveaux d'activités :

- activités de gestion incluant la planification, le contrôle et l'assurance qualité.
- activités de développement incluant la spécification, la conception, la formalisation, l'implémentation et la maintenance.
- activités de support incluant l'acquisition de connaissances, l'intégration, l'évaluation, la documentation, et la gestion de configuration.

Dans le cadre des activités de gestion, l'activité de la planification identifie les tâches devant être accomplies, leurs organisations et leurs temps d'acheminement et les ressources associées. Cette activité est essentielle pour les ontologies qui utilisent les ontologies stockées dans des bibliothèques ou ontologies qui requièrent un haut niveau d'abstraction et de généralité. Ainsi, l'activité de contrôle garantit que les tâches planifiées doivent être remplies et exécutées. Ensuite l'activité assurant la qualité assure que la qualité de chaque output de produits (ontologie, logiciel et document).

En ce qui concerne les activités de développement, l'activité de spécification définit l'objectif de la création de l'ontologie, ses utilisations prévues et ses utilisateurs. L'activité de conceptualisation structure la connaissance du domaine en tant que modèles significatifs au niveau des connaissances. L'activité de formalisation transforme le modèle conceptuel en un modèle formel ou semi-calculable. Ainsi, l'activité d'implémentation construit des modèles exécutables dans un langage informatique. Enfin, l'activité de maintenance met à jour et corrige l'ontologie.

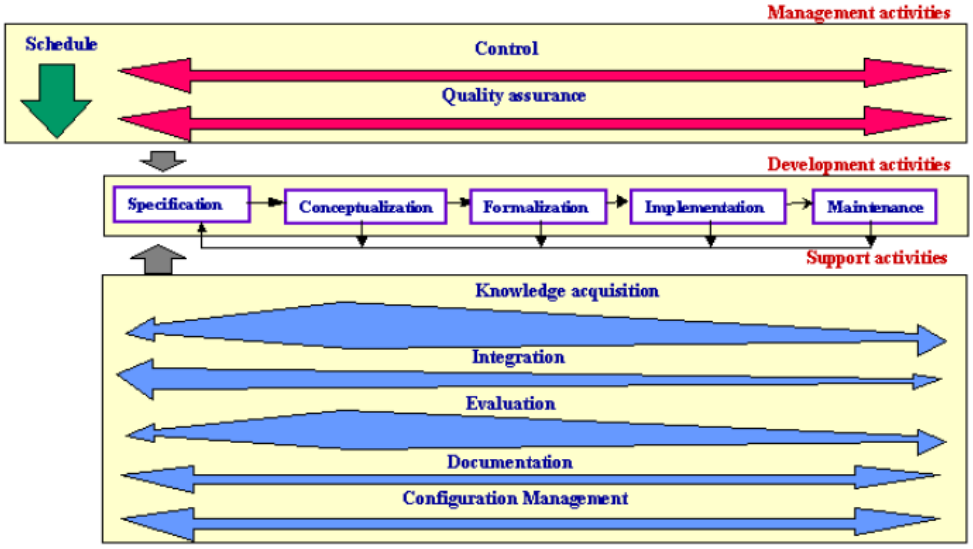


Figure 4-1 Décomposition of METHONTOLOGY (Fernández-López, Gómez-Pérez, & Juristo, 1997)

Finalement, les activités de support (soutien) comprennent une série d'activités, réalisées en parallèle aux activités de développement sans lesquelles l'ontologie ne peut être construite. L'activité d'acquisition des connaissances acquiert la connaissance d'un domaine donné. L'activité d'évaluation émet un jugement technique sur les ontologies, leurs environnements logiciels et la documentation concernant le cadre de référence lors de chaque phase et entre les phases de leur cycle de vie (Gómez Pérez et al., 1995). L'activité d'intégration des ontologies est nécessaire lors de la construction d'une nouvelle ontologie qui réutilise d'autres ontologies déjà disponibles. La documentation contenant les détails en toute clarté et exhaustivité concerne les phases terminées et les produits générés. La gestion de configuration enregistre toutes les versions de la documentation, des logiciels et le code de l'ontologie pour contrôler les changements.

4. Processus de développement de IMAMO

Dans le processus de développement de IMAMO, nous avons suivi la philosophie de METHONOTOLOGY pour utiliser conjointement les activités de support lors des activités de développement. Par conséquent, nous avons exploité l'activité de support *« Acquisition des connaissances »* lors de la mise en œuvre de l'activité de développement *« spécification »*. Ainsi, au cours de l'activité de développement *« conceptualisation »*, nous avons pris appui sur les activités de support *« Acquisition des connaissances »* et *« intégration »*. En ce qui concerne l'activité de support *« évaluation »*, cette dernière doit être répétée dans les différentes activités de développement *« conceptualisation », « formalisation »* et *« implémentation »*. Cependant, pour des raisons de structuration du chapitre, nous présenterons cette activité dans une sous-section séparée contenant les différentes évaluations faites dans chacune des activités des trois activités de développement concernées.

4.1- Spécification

La production d'un document de spécification d'ontologie peut être faite de façon informelle, semi-formelle ou formelle écrite en langage naturel, en utilisant respectivement un ensemble de représentations intermédiaires ou de questions de compétence (forme d'interview structuré fournissant des exemples réels du domaine) (Fernández-López, Gómez-Pérez, & Juristo, 1997). Sachant que la création d'une ontologie n'est pas une tâche

triviale, ceci exige non seulement des compétences dans les technologies de l'information mais aussi dans le domaine conceptualisé (Frankovic & Budinska, 2006) d'où l'importance de l'activité d'acquisition des connaissances pour l'édition du document de spécification.

4.1.1- Acquisition de connaissances

Pour acquérir les connaissances du domaine de maintenance nous avons pris appui sur les normes, les projets de recherche et les experts industriels dans la maintenance. Concernant les normes, nous utilisons les normes AFNOR[38] et la norme de MIMOSA[39]. Les modèles fournis par les projets PROTEUS [40] et PROMISE[41] ont également servis comme base. Enfin, nous avons adopté l'expertise métier de la maintenance de divers experts, gestionnaires et opérateurs de différentes entreprises telles que Cegelec SA France & Allemagne, Tornos (Suisse), Peugeot (Belfort, France) et em@systec (France). De plus, différents travaux de recherches telles que celles du Retour et al. (Retour, Bouche, & Plauchu, 1990), Kaffel (Kaffel, 2001) et Rasovska et al. (Rasovska, Chebel -Morello, & Zerhouni, 2005) sont également pris en compte.

Par conséquent les concepts qui doivent être identifiés couvrent en même temps l'ensemble du processus de maintenance et les couches des composants (voir section 2, chapitre 1). L'identification de concepts liés à chaque couche est basée sur les modèles de MIMOSA-CRIS (Kahn, 2003), du projet PROTEUS (Bangemann, et al., 2006), du SOM de PROMISE (PROMISE, 2008) et du projet SMAC (SMAC, 2009).

4.1.2- Document de spécification

A partir de l'activité d'acquisition, nous avons édité un premier document de spécification, tableau 1 en est l'illustration concernant IMAMO. Les principales cases clés de ce tableau sont le domaine à spécifier, l'objectif de la construction de l'ontologie, le degré de formalisation de l'ontologie et le cadre de cette ontologie.

Tableau 4-1 Document de spécification des exigences de l'ontologie

Domaine	Maintenance industrielle
Nom	IMAMO: Industrial MAintenance Management Ontology Ontologie de gestion de maintenance industrielle
Date	2010
Concepteurs	Mohamed-Hedi Karray, Brigitte Morello, Thibault Bobyck
Développeur	Mohamed-Hedi Karray, Thibault Bobyck
Objectif	Les préoccupations de l'ontologie concernent la plupart des concepts de maintenance industrielle où l'information sur toutes les activités et actions techniques, administratives et de gestion est requise dans le système d'information de maintenance. Cette ontologie peut être utilisée pour déterminer la prise de décision tout au long du cycle de vie des activités de maintenance et détecter la panne jusqu'à l'intervention et la réparation.

[38] http://www.afnor.org/
[39] http://www.mimosa.org/
[40] http://www.proteus-iteaproject.com/

[41] http://www.promise.no/

Degré de Formalité	Formel
Cadre	Structure des équipements à maintenir, les pièces de rechange, l'activité de surveillance, la détection de panne, les événements, les ressources matérielles, les acteurs de maintenance, documents techniques, documents administratifs, l'intervention, les rapports d'intervention, les états des équipements, cycle de vie de l'équipement.
Sources de connaissances	Standards (AFNOR, MIMOSA..), projets, experts

4.3- Conceptualisation

Dans cette activité, les connaissances du domaine sont structurées selon un modèle conceptuel qui décrit le vocabulaire du problème et sa solution du domaine étudié. Dans cette phase, comme le montre la Figure 2, Gomez-Perez et al recommandent une série de représentations intermédiaires pour conceptualiser sous forme d'arborescence de classification de concept, un dictionnaire de données, un tableau de la règle, etc. Un glossaire de termes complet doit d'abord être construit, incluant concepts, instances, verbes et propriétés. Cette activité est principalement basée sur les activités de soutien (de support) notamment d'acquisition de connaissances et d'intégration.

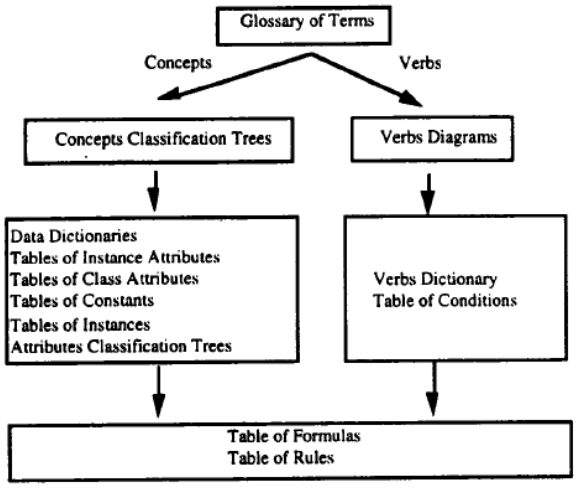


Figure 4-2 Set of Intermediate Representations in the conceptualization phase (Fernández-López, Gómez-Pérez, & Juristo, 1997).

4.3.1- Activités de support : Acquisition et intégration

Tant que les ontologies sont construites pour être réutilisées, leur réutilisation est l'une des questions importantes dans leur construction. Selon Pinto et al, il existe deux processus différents de réutilisation (Sofia Pinto, Gomez-Perez, & Martins, 1999) : la fusion et l'intégration. La fusion est définie comme le processus de construction d'une ontologie dans un sujet particulier en utilisant deux ou plusieurs ontologies différentes dans ce même sujet (Sofia Pinto, Gomez-Perez, & Martins, 1999). L'intégration est définie comme le processus de construction d'ontologie qui agrège, combine ou assemble des ontologies sources. Dans le cadre de ce travail pour la construction de IMAMO, nous avons adopté ces deux processus.

L'absence d'une ontologie opérationnelle couvrant tout le domaine de la maintenance industrielle doit également être notée. Par ailleurs, bien que la maintenance industrielle étant différente par définition de la maintenance de logiciels (IEEE-610.12, 1990), il y a quelques similitudes entre les deux domaines. Nous avons par conséquent pris en compte les ontologies de maintenance de logiciels existantes.

4.3.1.1- Acquisition des connaissances

Dans le domaine de la maintenance des logiciels, Kitchenham et al. (Kitchenham, et al., 1999) ont développé une ontologie préliminaire pour identifier un certain nombre de facteurs qui influencent la maintenance [SMO: software maintenance ontology]. Ruiz et al. (Ruiz, Vizcaino, Piattini, & García, 2004) ont développé une ontologie semi-formelle dans laquelle les principaux concepts sont décrits selon des sous ontologies concernant les produits, les agents, les activités, les processus, les workflow et les mesures. Cette ontologie représente aussi bien les aspects statiques que dynamiques liés à la gestion de projets de maintenance de logiciels.

Par contre dans le domaine de la maintenance industrielle, MIMOSA est la première initiative pour unifier les éléments de données à échanger entre équipements spéciaux tels que les outils de surveillance des états, par l'établissement du modèle MIMOSA-CRIS (Common Relational Information System). Ce dernier est un modèle de données relationnelles des informations de la maintenance (Kahn, 2003).

Par ailleurs, nous soulignons qu'au cœur de l'activité de maintenance il y a un équipement et que ce dernier est considéré comme un produit, nous nous sommes intéressés aux ontologies sur les produits. Dans ce cadre, Gzara (Gzara, 2000) a présenté dans sa thèse un référentiel produit comme une ontologie spécifiant les concepts décrivant le cycle de vie et états du produit, les objets techniques, la nomenclature d'un produit, les fonctions, les articles, les documents, les modèles de produit, les représentations et les points de vue. Ainsi cette ontologie spécifie les concepts décrivant les processus, leurs composants, leurs relations, les entrées sorties et les ressources.

Toujours dans le cadre des produits, Matsokis et Kiritsis (Matsokis & Kiritsis, 2009) ont proposé une approche basée sur l'ontologie de la gestion de cycle de vie des produits comme une extension de l'ontologie proposée dans le projet PROMISE (PROMISE, 2008). Ce dernier fournit un modèle d'objet sémantique (SOM) pour la gestion des données et des connaissances des produits.

Pour créer IMAMO, nous avons commencé à partir des modèles développés dans le projet PROTEUS (Rasovska, 2006) pour introduire une première version d'ontologie de maintenance que nous avons publié dans (Karray, Chebel-Morello, & Zerhouni, 2009). Cette ontologie est composée de 62 concepts et 70 relations intégrant les principaux concepts utilisés dans PROTEUS. Ensuite, dans le cadre du projet SMAC (SMAC, 2009), nous avons établi une correspondance (un *mapping*) entre cette ontologie et le modèle PROMISE. Comme résultat, nous avons proposé une version plus évoluée de cette ontologie orientée maintenance intégrant certains concepts liés et inclus dans la phase de milieu de vie *(MOF-Middel of Life)* de gestion du cycle de vie des produits *(PLM-Products Lifecycle Management)*. Cette ontologie, appelée Modèle SMAC, était conceptualisée par le diagramme de classe UML et implémentée avec OWL-DL via l'outil Protégé 2000 (Matsokis, Karray, Morello-Chebel, & Kiritsis, 2010).

4.3.1.2- Intégration

Ensuite, nous nous sommes intéressés plus particulièrement au cadre de la maintenance. Nous avons pensé à l'intégration de certains concepts du modèle SMAC en relation avec le cycle de vie des équipements de manière à prendre en compte (1) le début du cycle concernant la phase de conception, (2) la phase du milieu du cycle par le suivi de tous les événements et les états de santé de l'équipement, et (3) la fin du cycle par le calcul des indicateurs soutenant la décision pour la réutilisation et le démontage.

Un exemple de certains concepts réutilisés et intégrés dans IMAMO est présenté dans le tableau 2, sachant que dans quelques cas les dénominations ont été adaptées au contexte industriel. De plus, comme cela a été mentionné plus haut, MIMOSA-CRIS est considéré comme la référence du domaine. Par conséquent, lors de la création IMAMO nous avons également pris en considération les classes utilisées dans le modèle MIMOSA-CRIS en intégrant et réutilisant certaines classes communes comme concepts dans IMAMO. Ainsi, une correspondance (un *mapping*) entre MIMOSA CRIS et le modèle conceptuel de IMAMO a donc été faite.

Tableau 4-2 Exemple d'un tableau de réutilisation

Ontologies ou modèles sources	**Concepts**	**Concepts de IMAMO**
PROMISE // MIMOSA-CRIS	*Product* -- *Asset*	Physical equipment
MIMOSA-CRIS // PROMISE	*Model* -- *As-designed-product*	Equipment Model
MIMOSA-CRIS // PROMISE	*Asset type* -- *product group*	Equipment group
MIMOSA-CRIS // SMAC-Model	*site* -- *Location site*	Site
MIMOSA-CRIS // PROMISE	*Event type* -- *Event*	Triggering event
MIMOSA-CRIS // PROMISE	*Measurement Event* -- *Field Data*	Measure
MIMOSA-CRIS	*Geoposition*	Equipment location + Geo-location system
MIMOSA-CRIS // SMAC-Model	*Alarm type* -- *Alarm*	Alarm
MIMOSA-CRIS // SMAC-Model	*Work Management Type* -- *Process*	Process pattern
MIMOSA-CRIS // SMAC-Model	*Work Task Type* -- *Process*	Intervention type
MIMOSA-CRIS // PROMISE	*Work Step* -- *Activity*	Step
MIMOSA-CRIS // PROMISE	*Work Order* -- *Document resource*	Work Order
MIMOSA-CRIS // SMAC-Model	*Work Request* -- *Process*	Work request process
MIMOSA-CRIS // SMAC-Model // SMO	*Agent* -- *personal resource* -- *Human resource + software resource+ hardware resource*	Actor
MIMOSA-CRIS // PROMISE // SMO	work step -- Activity -- Maintenance activity	Maintenance task

MIMOSA-CRIS // PROMISE // SMO	*Logistic Resource* -- *resource* -- *resource*	Resource
MIMOSA-CRIS // SMAC-Model	Asset Function + Model Function -- *function + function group*	Function + subfuntcion

Cependant, une intégration plus poussée au niveau des instances est bien possible. Par exemple, comme le montre la Figure 4-3, l'ontologie fonctionnelle proposée par Kitamura et Mizoguchi peut être intégrée sous forme d'instances des concepts *«function» et « subfunction »*.

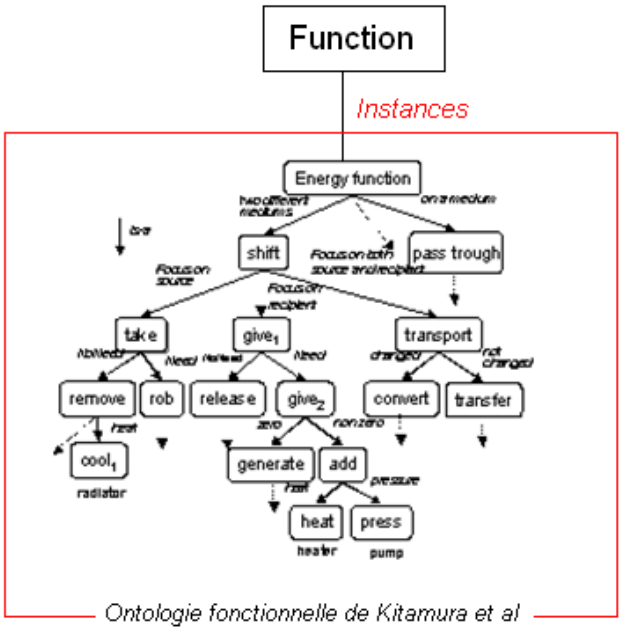


Figure 4-3 Relation entre le concept fonction et l'ontologie fonctionnelle

Ainsi, l'ontologie de pannes présentée par Kitamura et Mizoguchi (voir Figure 4-4) peut aussi être intégrée sous forme d'instances dans les concepts concernant la partie diagnostic et résolution de problèmes (Kitamura & Mizoguchi, 1999).

4.3.2- Glossaire des termes et dictionnaires de données

Nous avons commencé la conceptualisation de IMAMO par la construction du glossaire des termes. Les concepts sont d'abord classés dans le glossaire en respectant les quatre couches identifiées par Rasovska et al (voir chapitre 1, section 2). Ensuite nous avons affiné cette liste de concepts par l'approfondissement de la première classification. La deuxième classification va plus loin que la première en subdivisant chaque couche en des sous-couches plus fines. Nous notons que certains concepts sont partagés entre différentes couches ou sous-couches. Ceci est fait sciemment afin de prendre en compte tous les concepts dans chaque couche dans l'objectif d'obtenir une vision plus claire et une identification plus précise. Ensuite, nous avons édité le dictionnaire des données en nous basant sur la norme européenne NF EN 13306:2001 publiée par l'AFNOR.

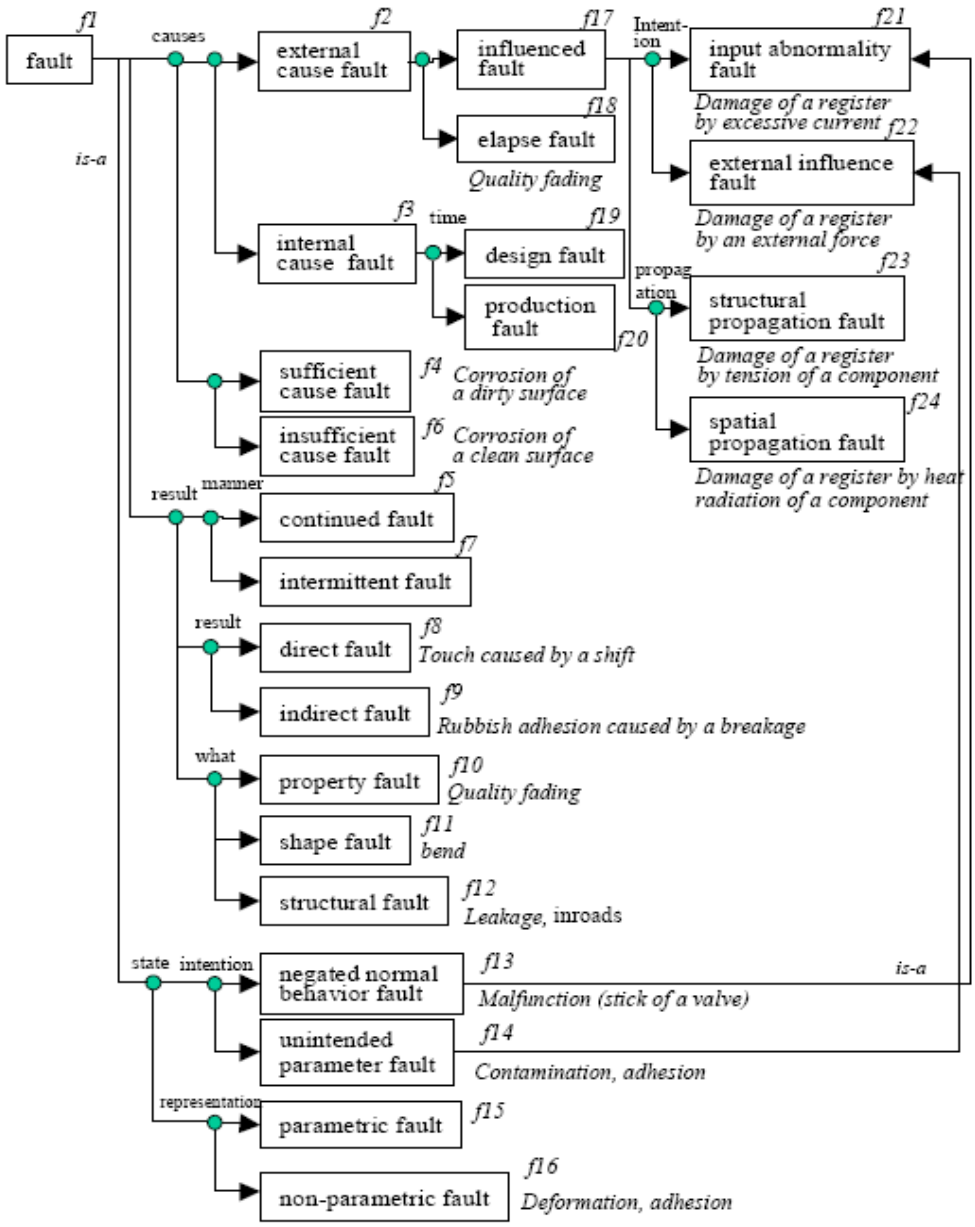


Figure 4-4 Différentes classes de panne de Kitamura (Kitamura & Mizoguchi, 1999)

Pour des raisons d'espaces, nous ne décrivons pas toutes ces étapes, mais nous ne présenterons que le dictionnaire de données associées au modèle conceptuel, afin d'avoir une meilleure compréhension des différents points de vue de l'ontologie (voir section 4.3.5).

Nous notons, toutefois, que IMAMO sera une ontologie globale[42] et négligera différents détails laissés à la discrétion des utilisateurs (nous entendons par utilisateurs les développeurs) pour les rajouter en fonction de leurs besoins. Dans ce cas, les utilisateurs peuvent adapter, faire évoluer et maintenir l'ontologie.

Dans notre cas, nous allons nous concentrer uniquement sur les concepts et les relations; les tâches et les activités de maintenance seront les instances des concepts se référant aux activités. Les relations entre concepts

[42] Une ontologie de référence, utilisée comme base pour la médiation et l'intégration des données.

ne seront pas présentées dans un tableau ou dans un dictionnaire de données, mais ils seront présentés sous forme d'associations avec les cardinalités dans le modèle conceptuel de l'ontologie.

4.3.3- Arbre de classification de concepts

Après cela, nous éditons les arbres de classification de concepts, représentant une taxonomie pour le domaine. Les taxonomies peuvent être également considérées comme des vues différentes pour présenter la même information. Par ailleurs, nous notons que le domaine de la maintenance est très vaste. Néanmoins, l'ontologie que nous développons ne contiendra pas beaucoup d'arbres profonds. Ceci est fait dans le but d'obtenir une ontologie riche avec différents types de relations et non pas une ontologie hiérarchique comme une taxonomie. Les relations soutenues par l'ontologie IMAMO sont *est-un (is-a), est-composant-de (is component of)* et d'autres verbes. La Figure 4-5 résume certains arbres de classification de concept dans IMAMO (ex. les relations *est-un (is-a)*).

4.3.4- Edition des règles

Comme préalablement mentionné, ce travail se focalise sur les concepts et les relations. Les règles sont laissées à la discrétion de l'utilisateur dans le cadre des activités d'évolution pour répondre à des besoins spécifiques. Dans la présente section, nous donnons quelques exemples de règles qui peuvent être éditées. Nous éditons ces dernières en utilisant la logique de description *ALCQHI*. Les règles peuvent ensuite être traduites dans un langage d'implémentation si celui-ci permet la définition de règles. L'avantage principal de ces règles est l'enrichissement de l'ontologie et l'accroissement des possibilités de raisonnements sémantiques ainsi que l'intelligibilité.

Par exemple, grâce à la règle définie suivante, l'identification des composants critiques est possible sans définir un nouveau concept appelé "composant critique". La règle décrit un composant critique comme un équipement physique ayant la valeur de la propriété de degré fonctionnelle supérieur ou égal à cinq.

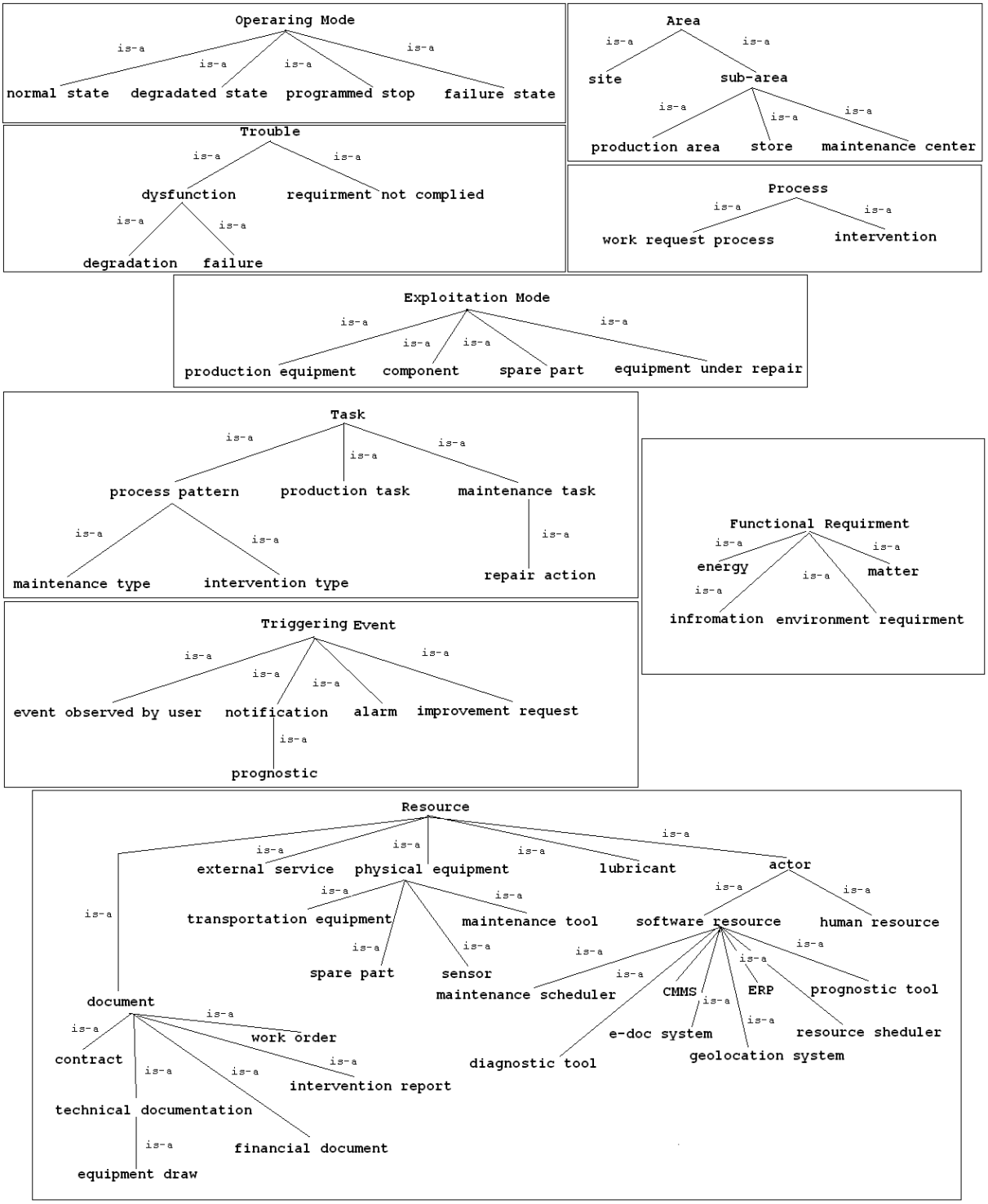


Figure 4-5 Exemples d'arbres de classifications de concepts dans IMAMO

4.3.5- Modèle conceptuel

Le langage de modélisation unifié UML serait un bon candidat pour représenter les ontologies et les connaissances (Cranefield S. , 2001). Cranefield et Purvis (Cranefield & Purvis, 2000) ont noté que les modèles UML ont certaines caractéristiques communes considérées comme celles du paradigme de la représentation des connaissances déclaratives (Bézivin, 2000). Les connaissances exprimées via UML sont facilement accessibles

pour la compréhension humaine. Dans un modèle UML, les connaissances peuvent être facilement modifiées en raison de la nature modulaire de la modélisation orientée objet.

Par ailleurs, de nouvelles connaissances peuvent être dérivées à partir de modèles UML par le raisonnement sur leur contenu (Cranefield S. , 2001). De ce point de vue, UML peut être considéré comme un candidat approprié pour la représentation des connaissances. Cranefield (Cranefield S. , 2001) s'est focalisé sur les avantages de l'utilisation de UML comme un langage d'ontologie. D'autre part, Bézivin (Bézivin, 2000) a souligné que le langage UML aborde le concept de représentation et plus particulièrement la définition d'ontologie présentée dans (Charlet, Bachimont, Bouaud, & Zweigenbaum, 1996).

Dans ce travail, nous avons choisi le diagramme de classes UML pour formaliser IMAMO. Ce choix est soutenu par l'expressivité graphique et la puissance sémantique d'UML recommandé dans les divers travaux de recherches mentionnés ci-dessus. Ceci facilitera l'échange entre les experts du domaine et de la compréhension humaine de l'ontologie. En outre, l'ontologie du domaine, bien formalisée de façon indépendante des méthodes de raisonnement, a une structure qui dépend « du comment les connaissances acquises seront utilisées » pour le raisonnement, car les experts fournissent des connaissances adaptées à leur raisonnement. Par conséquent, les méthodes de raisonnement seront considérées dans la phase de mise en œuvre (d'implémentation).

Pour une meilleure lisibilité, notamment sur le plan des relations entre concepts, nous décomposons le diagramme de classe en huit vues, selon la classification des couches utilisées dans la phase d'identification des termes du glossaire.

Nous notons que ces vues ne présentent pas de sous-ontologies ou de packages (voir Figure 4-6). Ces vues sont :

1- La vue structurelle présentant la composition d'équipement, liée à la couche d'analyse de l'équipement.
2- La vue fonctionnelle/dysfonctionnelle, caractérisant les différentes fonctionnalités de l'équipement et de ses composantes ainsi que le diagnostic de défaut et de la couche d'expertise.
3- La vue événementielle qui présente les événements déclencheurs lancés après les échecs et / ou la dégradation, liée à la couche de diagnostic de fautes et expertise.
4- La vue informationnelle qui présente les différentes ressources (documents, humains, logiciels, outils, indicateurs, etc.) qui sont liées à l'équipement et les tâches de maintenance ainsi que de la stratégie et les processus de maintenance, qui est elle aussi liée aux couches de gestion des ressources et de la gestion de stratégie de maintenance.
5- La vue interventionnelle qui propose des concepts liés au processus d'intervention.
6- La vue stratégique présentant les aspects managériaux de la stratégie de maintenance et des contrats.
7- La vue des processus qui englobe la présentation de tous les processus techniques, administratifs et de gestion.
8- La vue mémorielle qui présente les concepts permettant le suivi du cycle de vie de l'équipement.

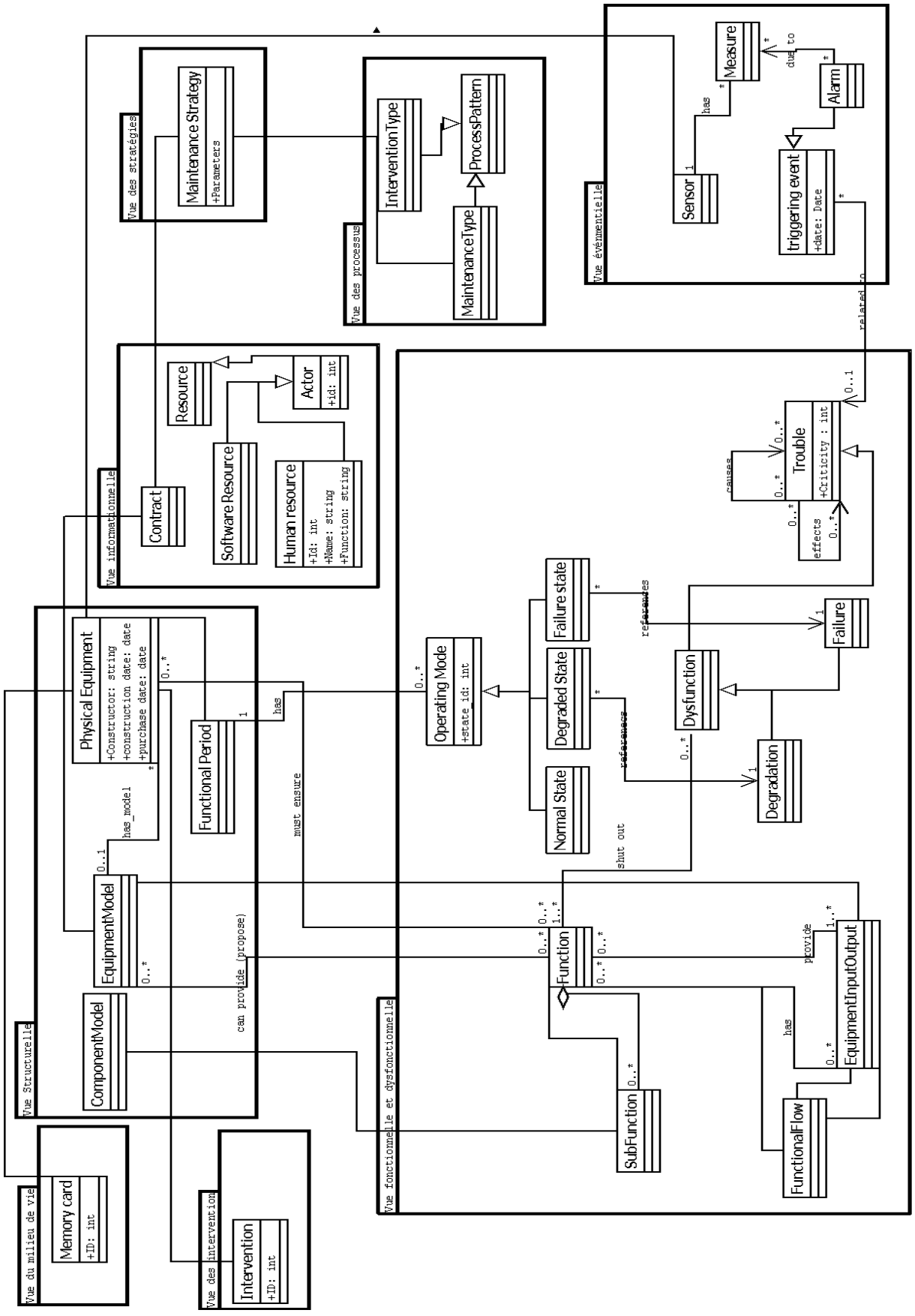


Figure 4-6 les différents points de vue dans une vue globale de IMAMO

Pour alléger la lecture de la thèse et diminuer la densité des informations, nous nous limitons, dans cette section, à la présentation détaillée de la vue structurelle. Ainsi, la vue des processus sera aussi développée dans le chapitre 5 qui décrit l'exploitation de l'ontologie dans les fonctionnalités fournies par la plateforme de s-maintenance. Les autres vues ainsi que leurs détails sont présentés dans l'*annexe ONTOL*. Nous notons aussi que les vues présentées dans la Figure 6 ne sont pas détaillées et ne contiennent que quelques échantillons de concepts et de relations.

La Figure 4-7 présente le diagramme de classe associé à cette vue structurelle. Cette dernière présente la composition de l'équipement, ainsi que les différents modes d'exploitation qu'un équipement physique peut avoir. La description de l'ensemble des concepts de cette vue est résumé dans le dictionnaire des données présenté dans le tableau 4-3.

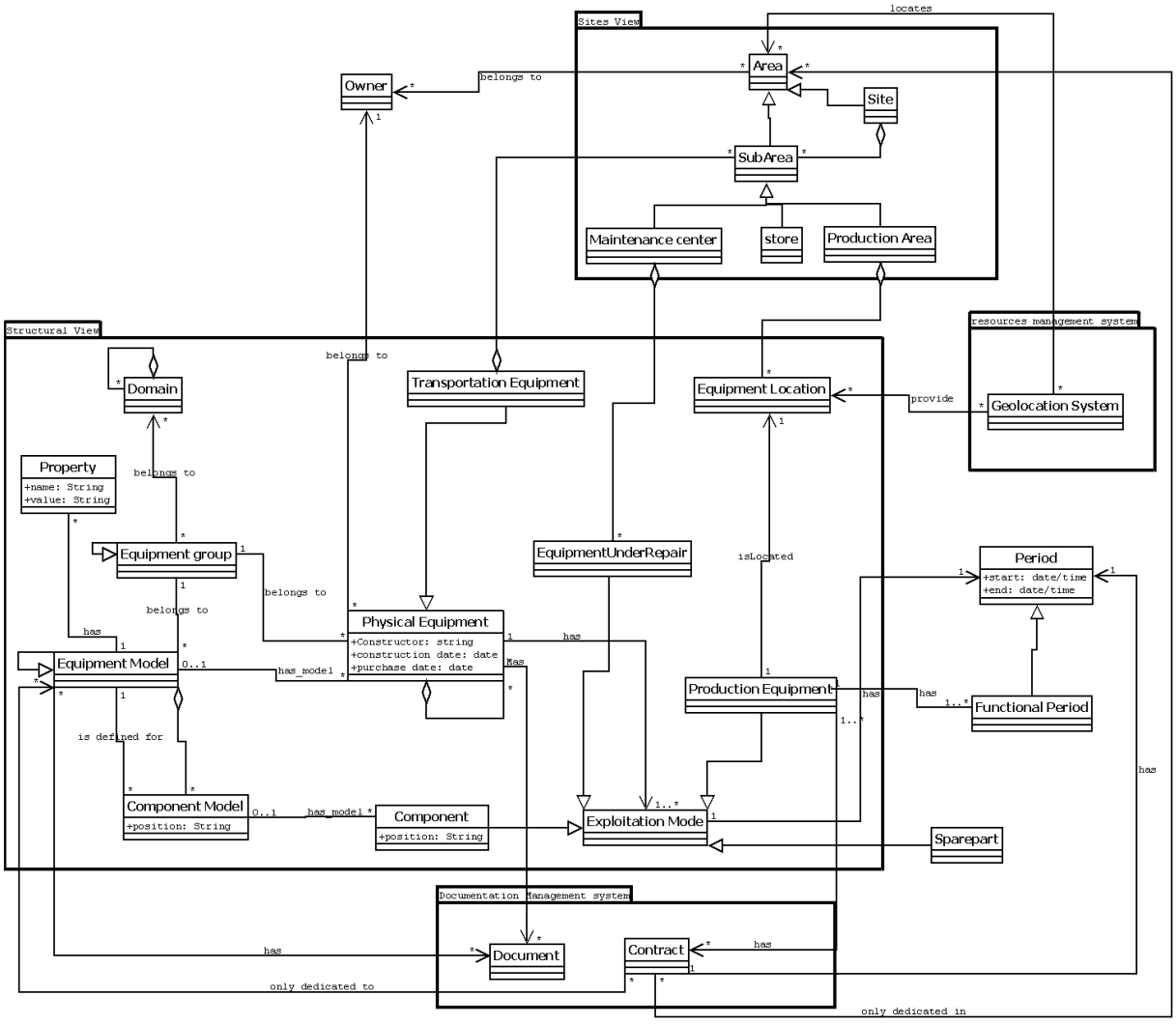


Figure 4-7 Vue structurelle dans IMAMO

Tableau 4-3 Dictionnaire de données de la vue structurelle

Concept	Termes français	Termes anglais	Description
Domain	Domaine	Domain ; Field;	Un domaine particulier de connaissances ou d'expertise (par exemple électrique).
physical equipment	Équipement	Physical equipment ; Asset; Physical- product; Machine; Device; Item;	Tout élément, composant, mécanisme, sous-système, unité fonctionnelle, équipement ou système qui peut être considéré individuellement. Un nombre donné d'équipements, par exemple un ensemble d'équipements, ou un échantillon, peut lui-même être considéré comme un équipement.
transportation equipment	Équipement de transport	Transportation equipment;	Un équipement physique spécifique de transport. Un moyen de transport qui peut contenir une ou plusieurs zone (s) de production, un ensemble d'équipes de maintenance, et un ensemble de magasins. Par exemple: Un bateau de pêche au large des côtes a sa propre décomposition (moteur permettant de se déplacer ...) ainsi qu'une zone de production qui nettoie le poisson et les congèle.
maintenance tool	Outils de maintenance	Maintenance tool;	Un équipement physique spécifique utilisé comme outil pour l'exécution des activités de maintenance. Ce type d'équipement physique doit être maintenu aussi.
equipment model	Modèle d'équipement	Equipment model; As-designed-product; Model	Vue conceptuelle de la composition physique de l'équipement. Il est composé par les différentes modèles des composants d'un équipement physique.
equipment group	Groupe de l'équipement ; Famille d'équipment ;	Equipment group ; Equipemnt family ;	un ensemble d'équipement ayant une conception de base commune, mais comportant des types différents à l'intérieur desquels existent des versions et des variantes. (exemple : voiture).
component model	Modèle de composant	Component model;	Vue conceptuelle d'un composant (par exemple le modèle de moteur électrique).
Component mode;	Mode composant	Component mode;	C'est un mode d'exploitation spécifique qui peut être joué par un équipement physique. Il a la particularité d'être au sein de l'équipement physique supérieure (par exemple moteur 3X57H).
exploitation mode	Mode d'exploitation	Exploitation mode;	C'est une abstraction d'un rôle joué par l'équipement. Elle présente les états d'exploitation qui peuvent prendre un équipement physique. Un équipement peut être exploité comme un composant, un équipement de production, une pièce de rechange ou être en cours de réparation.
equipment under repair	Équipement en cours de réparation	Equipment under repair;	Mode d'exploitation spécifique affecté à un équipement physique alors qu'il est au cours de réparation ou situé dans un centre de maintenance pour être réparé.
production equipment mode	Équipement en mode production	Production equipment mode;	Mode d'exploitation spécifique affecté à un équipement physique alors qu'il est exploité dans une des tâches de production et / ou situé dans une zone de production.
Spare part mode	Équipement en mode pièce de rechange	Spare part mode;	Mode d'exploitation spécifique affecté à un équipement physique destiné à remplacer un équipement physique correspondant, afin de rétablir la fonction originale fournie

			par l'équipement physique. Généralement, il est situé dans un magasin.
equipment location	Emplacement de l'équipement	Equipment location;	Position d'un équipement physique dans une zone de production (pour localiser et suivre les positions de l'équipement).
Area	Zone	Area;	Région géographique spécifique (pour gérer multi site)
sub area	Sous zone	Sub area;	Région qui constitue une partie d'une zone.
site	Site ; parc ;	Site ; Park ;	Une zone spécifique avec des limites définies composée d'un ensemble de sous-zones.
maintenance center	centre de maintenance	Maintenance center; Maintenance workshop;	Sous-zone spécifique ou les tâches de maintenance sont effectuées.
store	Magasin	Store;	Sous-zone spécifique réservée pour le stock des équipements physiques qui vont être utilisés dans le futur (pièces de rechange).
production area	zone de production	Production area;	Sous-zones spécifique réservée pour les taches de production.
owner	Propriétaire ;	Owner ;	Propriétaire de l'équipement. Ceci permet de gérer les cas d'externalisation de la fonction de maintenance ainsi que la
Period	Période	Period;	Intervalle de temps.
functional period	période fonctionnelle	Functional period;	Période spécifique durant lequel un équipement physique doit assurer certaines fonctions.

Grâce à l'activité d'acquisition nous avons constaté que dans le référentiel de produit présenté dans sa thèse, Gzara classifie les produits selon différents niveaux d'abstraction, auxquels sont associés des connaissances différentes (Gzara, 2000). Gzara considère que le concept produit peut faire référence soit à des objets virtuels (le produit objet de l'activité de conception) soit à des objets physiques (le produit objet d'une livraison au client). Etant donné que les équipements industriels à maintenir sont considérés comme des produits, lors de la création de cette vue, nous avons adopté cet esprit de différenciation entre le niveau virtuel et le niveau physique. Ceci peut être remarqué avec les deux concepts *« Equipment model »* qui fait référence à l'objet virtuel et *« Physical Equipment »* qui fait référence à l'objet physique. L'exemple présenté dans la Figure 4-8 illustre cette démarche.

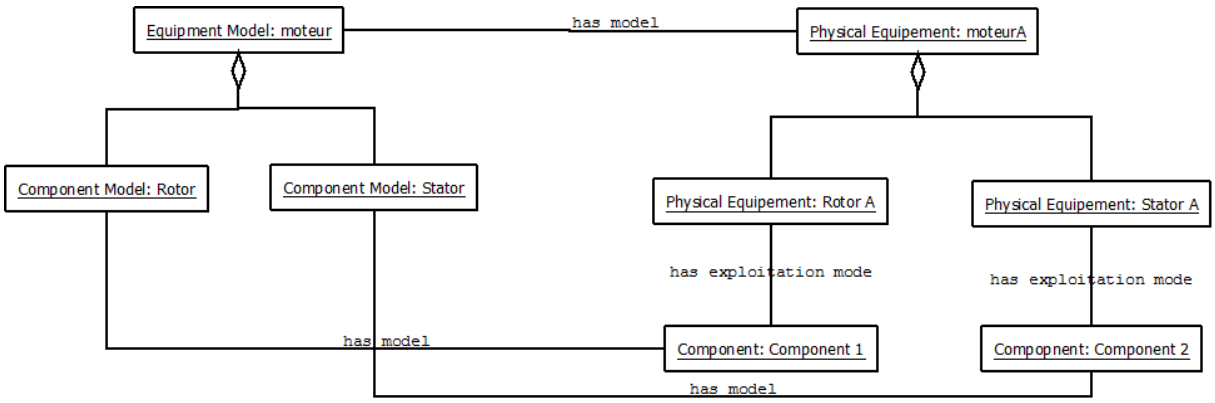


Figure 4-8 Exemple d'instanciation des concepts *« equipment model »* et *« physical equipment »*

Avec cette conceptualisation nous pouvons instancier un équipement physique à partir d'un modèle d'équipement, un mécanisme se chargera de copier et de lier chaque composant à ce qui correspond au nouveau modèle. Nous pouvons, aussi, décomposer un équipement sans définir son modèle. Cependant, il doit appartenir à une famille (un groupe). Nous pouvons définir plusieurs groupes d'équipements afin de les organiser par famille et sous-famille ainsi que définir le domaine auquel appartient celle-ci. Par exemple, « moteur » est une instance du concept « Equipment Group ». Nous pouvons instancier que le moteur électrique hérite de moteur et appartient au domaine électrique. Nous pourrions ainsi faire une étude concernant tous les moteurs électriques et des équipements gérés par la plateforme.

Un modèle d'équipement possède sa propre décomposition grâce à des modèles de composants (ce qui en fait un concept plus performant par rapport au concept « Equipment Group ») et la possibilité de lier les modèles avec un modèle fonctionnel, dysfonctionnel et des documents (grâce aux relations entre les concepts). Un modèle peut hériter d'un autre (exemple : les modèles pousseur et tireur héritent du modèle actionneur pneumatique). Ainsi, il hérite des liens qui lui sont associés (documents, décomposition, etc.).

Un modèle de composant (le concept « component model ») représente le rôle de composant que va jouer un équipement dans un autre équipement (supérieur). Pour cela, nous définissons un modèle de composant sur un modèle d'équipement (décomposition) et nous définissons également quel modèle d'équipement pourra être utilisé pour remplir ce rôle de composant.

Il est possible également de définir la différence entre deux composants de même nature sur un même modèle d'équipement, grâce à une localisation ou une attribution.

Par exemple, une machine est équipée de deux capteurs de température, à l'entrée et à la sortie. Ces deux capteurs appartiennent au même modèle « Capteur De Température ». En permettant d'ajouter la position (entrée, sortie) du modèle d'équipement qui sera utilisée comme composant, nous pouvons faire la différence entre ces deux capteurs.

Il en va de même pour l'attribution des composants. Par exemple, deux actionneurs pneumatiques utilisés sur une même machine : un en tireur et l'autre en pousseur.

C'est dans le cadre de la séparation des niveaux physique et virtuel, nous avons adopté le principe de « tout est équipement physique ». Ainsi, réellement, nous ne considérons pas l'existence physique des pièces de rechange ou des composants. En effet, ces derniers sont considérés comme des modes d'exploitation virtuels de l'équipement physique. Dans le cadre physique, un composant est un objet qui équipe un objet supérieur. Donc, un ensemble d'équipements physiques compose un équipement physique supérieur, ou nous pouvons dire qu'un équipement physique est composé d'un ensemble d'équipements physiques.

Par conséquent, afin de connaître à quel type d'équipement physique nous avons affaire, il suffit de connaître le rôle qu'il joue ou a joué grâce au concept virtuel des changements de type (Exploitation mode). Par exemple, l'équipement physique Rotor A231 jouait le rôle de pièce de rechange « spare part » dans le magasin E13 (instance du concept « store ») depuis son achat. Or, depuis son installation la semaine dernière sur le moteur

67UA (instance de « physical equipment »), il a un nouveau mode d'exploitation : une instance du concept « component ».

Ce concept de mode d'exploitation, « exploitation mode », permet de conserver l'historique de l'exploitation de l'équipement physique même lorsque l'équipement change de lieu de production. Par conséquent, il joue un rôle important vis-à-vis de la gestion du cycle de vie.

Le concept « Production Equipment » représente la vue des équipements vis-à-vis des opérateurs de production. Cette vue est différente de celle des opérateurs de maintenance. En effet, les équipements de production restent identiques du point de vue production, même si des éléments ou équipements complets sont changés après la maintenance.

Par exemple, un robot a été remplacé par un autre identique mais avec une immatriculation différente et pourtant la production considère qu'il s'agît du même robot. Il est absolument nécessaire pour la maintenance de savoir de quel équipement physique il s'agit.

Ainsi, nous définissons sur ces équipements de production des périodes de fonctionnement durant lesquelles ils doivent remplir certaines fonctions. Ces équipements sont également localisés grâce au concept « Equipment Location ». La traçabilité des déplacements des équipements de production permet de garantir celle des équipements physiques qui jouent le rôle de production. Un équipement de production n'existe pas s'il n'est pas réalisé/concrétisé par un équipement physique. Par exemple, un robot effectue des tâches d'emballage sur une chaîne de production. Changer le robot par un autre identique ne change pas l'équipement de production (il garde sa localisation et ses fonctions à remplir et reste le même vis-à-vis de la production). Si le robot est retiré physiquement et qu'il n'y en a plus, on ne parle plus d'équipement de production, il disparaît car il n'est plus réalisé par le robot mais le robot physique existe toujours (comme pièce de rechange, autre équipement de production, équipement en réparation ou même composant).

Enfin, cette vue d'ontologie conceptualisée d'une façon spécifiée permet de gérer la maintenance de tous types d'équipements. De plus, elle permet aussi de fournir différents mécanismes à valeurs ajoutées dans la plateforme de s-maintenance. En effet il est possible de :

- Instancier un équipement avec ses composants.
- Re-classifier dynamiquement les composants des équipements (avoir différents niveaux de nomenclature de l'équipement).
- Filtrer et classifier les équipements par rapport à un domaine.
- Contrôler la décomposition d'un équipement physique par rapport aux modèles des équipements.
- Classifier les équipements selon le mode d'exploitation.
- Tracer le cycle de vie des équipements physiques.
- Localiser les équipements physiques (suivi spatial).
- Tracer l'exploitation d'un équipement de production.
- Evaluer l'état de santé de l'équipement grâce à des indicateurs d'exploitation pour le recyclage et la réutilisation.

4.4- Formalisation

Pour que le modèle conceptuel soit transformé en un modèle formel ou semi-compatible, il doit alors être formalisé en utilisant un système de programmation orienté-cadre (Frame-Oriented Programming)[43] ou d'un système de représentation de logique de description.

Avec le modèle UML, nous ne pouvons pas savoir si la conception de l'ontologie a correctement modélisé les connaissances. Toutefois, l'expressivité des modèles UML construits peut conduire à des conséquences implicites qui peuvent passer inaperçues par le concepteur dans des diagrammes complexes et provoquer diverses formes d'incohérences ou de redondances dans le modèle ontologique (Berardi, Calvanese, & De Giacomo, 2005). Par conséquent, il serait hautement souhaitable de détecter automatiquement les propriétés formelles pertinentes du modèle de l'ontologie UML telles que les incohérences et les redondances mentionnées ci-dessus. Pour rendre le modèle opérationnel, il doit donc être formalisé (Van Der Straeten, Simmonds, Mens, & Jonckers, 2003).

Nous avons donc décidé d'utiliser une variante de description logique comme langage de représentation de IMAMO étant donné ses propriétés de possibilités de décidabilité, de subsomption, et d'inférence.

Afin d'encoder des diagrammes de classe de IMAMO dans *ALCQHI*, nous avons utilisé la même approche que dans (Berardi, Calvanese, & De Giacomo, 2005). Alors que dans le modèle UML de notre ontologie, toutes les associations sont binaires, elles peuvent donc être traduites de la même manière qu'une agrégation (Obrst, Werner, Inderjeet, Steve, & Smith, 2007), avec les assertions supplémentaires pour une association *ASSOC* entre les classes de C1 et C2:

.

Par exemple, l'association entre la relation Has-Equipment-Model et les concepts Physical-Equipement et le Equipement-Model sera traduit comme suit :

Nous notons que dans la plupart des cas, lors de la définition du nom de la relation, nous composons celui-ci par une combinaison d'un verbe avec les deux noms des concepts connexes, comme dans le cas de l'exemple présenté ci-dessus.

4.5- Implémentation

Nous avons ensuite traduit le modèle IMAMO formalisé en PowerLoom. Malgré la disponibilité actuelle de la version bêta 4.0, nous avons choisi de travailler avec la version stable 3.2.0. En outre, il convient de noter qu'un

[43] http://www.ai.sri.com/pub_list/236

exportateur PowerLoom pour l'éditeur Protégé a été mis en œuvre. Il peut écrire des ontologies en utilisant le langage du Frame-Protégé de PowerLoom.

Par ailleurs, l'interface graphique PowerLoom (ou éditeur de connaissances), un client graphique basé sur Java pour PowerLoom, est désormais une caractéristique standard et disponible avec PowerLoom à partir de la version 4.0 (PowerLoom, 2011) ce qui facilitera la manipulation de la base de connaissances définie sous ce langage.

Dans cette section, nous présentons une partie de la vue structurelle de l'équipement implémentée sous PowerLoom. L'ontologie complète sous PowerLoom est présentée dans l'annexe OPL. Chaque class UML dans le modèle conceptuel est traduite comme un concept PowerLoom avec la commande *« DEFCONCEPT »*. Les associations et les attributs de ces classes sont traduits comme des relations PowerLoom en utilisant les commandes *« DEFRELATION »* et *« DEFFUNCTION »*.

```
(DEFMODULE "/PL-KERNEL-KB/PL-USER/ONTOLOGIE-MAINTENANCE"
  :DOCUMENTATION "Module for Maintenance"
  :INCLUDES ("PL-USER"))

(IN-MODULE "/PL-KERNEL-KB/PL-USER/ONTOLOGIE-MAINTENANCE")
(IN-DIALECT :KIF)
(DEFCONCEPT COMPONENT)
(DEFCONCEPT PHYSICAL-EQUIPMENT)
(DEFRELATION PHYSICAL-EQUIPMENT-ID ((?C PHYSICAL-EQUIPMENT) (?ID STRING)))
(DEFRELATION PHYSICAL-EQUIPMENT-TYPE ((?C PHYSICAL-EQUIPMENT) (?TYPE STRING)))
(DEFRELATION PHYSICAL-EQUIPMENT-CONSTRUCTOR ((?C PHYSICAL-EQUIPMENT) (?CONSTRUCTOR STRING)))
(DEFRELATION PHYSICAL-EQUIPMENT-ACTUAL-LOCATION ((?C PHYSICAL-EQUIPMENT) (?ACTUAL-LOCATION
STRING)))
(DEFRELATION PHYSICAL-EQUIPMENT-HABITUAL-LOCATION ((?C PHYSICAL-EQUIPMENT) (?HABITUAL-LOCATION
STRING)))
(DEFRELATION PHYSICAL-EQUIPMENT-CONSTRUCTION-DATE ((?C PHYSICAL-EQUIPMENT) (?CONSTRUCTION-DATE
STRING)))
(DEFRELATION PHYSICAL-EQUIPMENT-PURSHASE-DATE ((?C PHYSICAL-EQUIPMENT) (?PURSHASE-DATE
STRING)))
(DEFRELATION PHYSICAL-EQUIPMENT-FUNCTIONNING-START-DATE ((?C PHYSICAL-EQUIPMENT)
(?FUNCTIONNING-START-DATE STRING)))
(DEFFUNCTION FUNCTIONNAL-DEGREE ((?C PHYSICAL-EQUIPMENT))
  :-> (?N INTEGER))
(DEFCONCEPT EQUIPMENT-MODEL)
(DEFRELATION EQUIPMENT-HAS-TOP-MODEL ((?E PHYSICAL-EQUIPMENT) (?MG EQUIPMENT-MODEL)))
(DEFRELATION EQUIPMENT-MODEL-INHERITS ((?MG1 EQUIPMENT-MODEL) (?MG2 EQUIPMENT-MODEL)))
(DEFRELATION EQUIPMENT-COMPOSED ((?E PHYSICAL-EQUIPMENT) (?COM PHYSICAL-EQUIPMENT)))
```

4.6- Evaluations

Nous rappelons que l'activité d'évaluation dans METHONTOLOGY est une activité de support qui doit être appliquée dans les principales activités de développement. Etant donné l'importance de cette activité dans le

processus de développement, nous la présentons dans une section séparée en agrégeant les différentes évaluations faites au long du processus de développement de IMAMO.

Cette activité d'évaluation permet de nous guider pour proposer une ontologie de qualité aussi bien conceptuelle, opérationnelle et fonctionnelle.

Toutefois, ces différents niveaux d'évaluation ne permettent pas de garantir l'adéquation entre les attentes de l'utilisateur lors de l'exploitation de cette ontologie. C'est à l'usage, que les performances ou les manques de l'ontologie transparaissent. L'évaluation d'une ontologie n'est actuellement pas entièrement satisfaisante. Il existe des métriques permettant de quantifier celle-ci, mais qui ne peut pas attester de la qualité d'utilisation.

Par contre, sensibilisé à ce problème d'évaluation, Brank et al ont réalisé une étude des approches d'évaluation et ont en identifié quatre types, à savoir :

- Approche basée sur la comparaison de l'ontologie à un Standard d'or *(Golden Standard en anglais)* ce qui nécessite l'existence de celui-ci.
- Approche basée sur l'utilisation de l'ontologie dans une application et l'évaluation des résultats.
- Approche basée sur la comparaison avec une source de données dans le domaine couvert par l'ontologie.
- Approche basée sur l'évaluation par humains qui tentent de déterminer dans quelle mesure l'ontologie répond à un ensemble de critères prédéfinis, normes, exigences, etc.

Brank et al ont affirmé que le choix d'une méthode d'évaluation appropriée dépend de la finalité de l'évaluation, sur l'application dans laquelle l'ontologie est utilisée et sur quel aspect de l'ontologie est évaluée. En revanche, une autre étude, présentée par Obrst et al. (Obrst, Werner, Inderjeet, Steve, & Smith, 2007), décrit l'évolution des stratégies et souligne les techniques actuelles d'évaluation d'ontologie, par exemple les critères, les questions et les aspects. Toutefois, il n'existe pas de manière standardisée, objective et largement acceptée des méthodes d'évaluation d'ontologie.

Nous prendrons appui sur les types d'approches identifiés par Brank et al pour donner des éléments de réponses, les plus complètes permettant d'évaluer l'ontologie IMAMO parmi les quatre approches existantes.

Par ailleurs, en dépit de l'absence d'un standard d'or dans la maintenance tout en sachant que nous avons effectué déjà une cartographie *(mapping)* entre IMAMO et le modèle MIMOSA-Cris (fait référence à la troisième approche identifiée par Brank et al), notre évaluation de IMAMO est composée de deux étapes principales, à savoir :

- Evaluation de la qualité du modèle conceptuel par rapport à des métriques.

- Evaluation métier axée sur la valeur ajoutée de l'ontologie en faisant appel respectivement à l'approche d'évaluation par un humain et l'approche par une application :

    o Vérification des fonctionnalités de IMAMO via une méthode questions / réponses.

o Évaluation de l'applicabilité, et de l'exploitation des connaissances en exécutant des requêtes et en évaluant leurs résultats.

4.6.1- Qualité du modèle conceptuel de IMAMO

Conformément à Tartir et al. (Tatir & Budak Arpinar, 2007), l'évaluation de la qualité d'une ontologie est importante pour plusieurs raisons : permettre au développeur de reconnaître automatiquement les zones qui pourraient avoir besoin d'être plus booster, et de savoir quelles parties de l'ontologie pourraient causer des problèmes. En effet, différentes dimensions sont disponibles pour évaluer la qualité d'une ontologie. Nous sommes intéressés par les métriques de qualité présentées par (Tartir, Arpinar, Moore, Sheth, & Aleman-Meza, 2005). Par conséquent, nous utilisons la métrique d'évaluation de schéma pour évaluer le succès du modèle conceptuel de IMAMO sur le niveau d'organisation des classes, la profondeur, l'équilibre, la richesse, la largeur et la hauteur de l'arbre d'héritage de l'ontologie qui peuvent jouer un rôle dans l'évaluation de la qualité.
Pour comprendre les mesures utilisées et la discussion ci-dessous, il est important de connaître les éléments suivants extraits de (Tartir, Arpinar, Moore, Sheth, & Aleman-Meza, 2005) :
Structure de l'ontologie (schema): un schéma d'ontologie est un sextuple $O := \{C, P, A, H^C, prop, att\}$, composé de deux ensembles disjoints *C et P* dont les éléments sont appelés respectivement des concepts et des relations, une hiérarchie de concepts H^C, H^C est une relation transitive dirigée $H^C \subseteq C \times C$ appelée aussi taxonomie de concept. $H^C(C_1, C_2)$ signifie que C_1 est un sous concept de C_2 (C1 hérite de C2). Ainsi, une fonction $prop$: $P \rightarrow C \times C$, qui relie les concepts sous forme non taxonomique. La fonction att: $A \rightarrow C$ reliant les concepts avec des valeurs réels.

Le modèle conceptuel de IMAMO contient 187 Relations (P), 103 Concepts (classes) (C), 60 Sous-classes (SC) and 40 Attribues (att).

Tableau 4-4 Qualification de la qualité du modèle conceptuel de IMAMO

Nom de métrique	Description	Formule	Résultat obtenu	Interprétation
Richesse des relations	Cette métrique reflète la diversité des relations et de la mise en place des relations dans l'ontologie. Une ontologie qui contient de nombreuses relations autres que celles des sous-classes est plus riche qu'une taxonomie qui a seulement des relations d'héritage (sous-classes). Formellement, la richesse des relations (RR) d'un schéma est définie comme le ratio du nombre de relations (P) définies dans le schéma, divisée par la somme du nombre de sous-classes (SC) (qui est le même que le nombre de relations d'héritage), plus le nombre de relations.	$RR = \frac{\quad}{\quad}$	IMAMO_RR = 187/(187+60) = 0.75 = 75%	Ce résultat obtenu dépasse largement la moyenne. Cela signifie que notre ontologie n'est pas à caractère hiérarchique. Ce n'est pas seulement une hiérarchie de classes, mais elle maintient un équilibre entre les relations d'héritage et les associations. Ceci est dû à l'inclusion des concepts du domaine et prouve que le modèle ontologique est orientée métier et répond aux besoins métiers de la maintenance.
Richesse de l'héritage	Cette mesure décrit la répartition des informations entre les différents niveaux de l'arbre de l'héritage de l'ontologie (c.à.d. arbre de classification des concepts) ou le niveau de déploiement des classes parentes. Ceci est une bonne indication de la façon dont les connaissances sont regroupées en différentes catégories et sous-catégories dans l'ontologie. Cette mesure permet de distinguer une ontologie horizontale d'une autre verticale ou une ontologie avec différents niveaux de spécialisation. Une ontologie horizontale (ou plate) est une ontologie ayant un petit nombre de niveaux d'héritage, chaque classe ayant un nombre relativement important de sous-classes. En revanche, une ontologie verticale contient un grand nombre de niveaux d'héritage où les classes ont un petit nombre de sous-classes. Formellement, la richesse de l'héritage (RI) de l'ontologie est définie comme le nombre moyen de sous-classes par classe.	$IRs= \frac{\quad}{\quad}$	IMAMO_$\Sigma H^C(C^1, C^i)$ = 60; IMAMO_IRs = 60/103 = 0.58.	Ce résultat est proche de la moyenne (0,5). Cela montre que dans le contexte de détails des connaissances, notre ontologie maintient un équilibre entre la généralité et l'explicité. Le modèle ontologique est hybride, il n'est ni verticale (ce qui pourrait refléter un type très détaillé de la connaissance que l'ontologie représente) ni horizontal (ce qui signifie qu'il représente un large éventail de connaissances générales). Nous considérons ce résultat comme un objectif atteint, parce que notre premier objectif était de construire un modèle ontologique générique pour le domaine de la maintenance. Cependant, il a été simultanément assez fort pour couvrir possible les nombreux aspects (concepts) de la maintenance.
	le nombre d'attributs définis pour chaque classe peuvent indiquer la qualité de la conceptualisation de l'ontologie et la quantité d'informations relatives pour instancier les			Le résultat obtenu montre la pauvreté du modèle conceptuel de IMAMO en termes d'attributs. Le résultat donne comme une moyenne de 0,38 attributs par concept qui est très faible. Lors de la construction de IMAMO on avait la conscience de cela vu que

Richesse des Attributs	données. Généralement, plus le nombre d'attributs définie augmente plus les connaissances de l'ontologie augmentent. Formellement, la métrique richesse des attributs (AR) est définie comme le nombre moyen d'attributs par classe. Elle est calculée comme le nombre d'attributs de toutes les classes (att), divisé par le nombre de classes (C).	AR= ——	IMAMO_AR = 40 /103 = 0.38	nous avons porté notre intérêt sur la présentation des concepts et leurs relations et non pas sur ses détaillées par l'identification des attributs. Bien que nous savons que ces détails sont très intéressants pour la qualité et la richesse de l'ontologie, ils ne sont pas simple à identifier parce que l'ontologie doit utiliser des termes expressifs et sans ambiguïté. Dans ce contexte, cette tâche est plus compliquée pour être générique que dans le contexte des concepts. Cependant, pour notre travail futur, nous envisageons la collaboration avec des experts métier afin d'identifier les attributs qui peuvent être génériques pour des concepts de maintenance dans IMAMO (C.à.d.: détails sur les classes du modèle conceptuel IMAMO).

4.6.2- Évaluation métier de l'ontologie

4.6.2.1- Vérification des fonctionnalités d'exploitation

Cette évaluation est basée sur l'approche «question / réponse » (Question asking en anglais) pour vérifier si certaines des fonctionnalités de l'ontologie sont possibles sur IMAMO. Ces questions sont résumées dans le tableau 4-5, où nous essayons de répondre et d'argumenter en faveur du type de question «Est-ce que IMAMO permet ...? ». Les questions sont classées en deux catégories. Premièrement, des questions métiers en relation avec les quatre couches du processus de maintenance identifiées par (Rasovska I. , 2006) concernant l'analyse de l'équipement, le diagnostic et l'expertise, la gestion des ressources et la stratégie de maintenance. La deuxième catégorie porte sur les capacités techniques de l'ontologie.

Tableau 4-5 Fonctionnalités assurées par IMAMO

Catégorie	Fonctionnalités	Est-ce que IMAMO permet ...?
Questions métiers	Décrire un équipement comme il est	Oui, grâce à la vue structurelle de l'équipement.
	Tracer l'historique de l'équipement	Oui, il est possible de récupérer les informations sur les usages actuels et passés de l'équipement physique grâce aux concepts *« exploitation mode » et « functional period »*. De plus, il permet de récupérer les différents modes d'exploitation au cours du cycle de vie d'un équipement grâce aux concepts *«operating mode »*, «triggering event», intervention et « life record ».
	Gérer les données de maintenance	Oui, il est possible de relier les informations avec le processus de maintenance grâce aux concepts des processus, le concept « trouble » et les concepts de ressources à savoir «intervention report», «work request», « diagnosis » et autres.
	Couverture étendue sur le cycle de vie	Oui, Il est possible de relier les informations de l'équipement avec les différentes phases du cycle de vie grâce aux concepts « life record », « equipment model» and « period ».
	Traiter automatiquement les données pour les événements	Oui, grâce à la manipulation de la vue événementielle par le classificateur de description de PowerLoom.
	Gestion des ressources	Oui, grâce à la vue informationnelle qui englobe tout les types de ressources.
	Gestion du pronostic et monitoring	Oui, grâce à la manipulation de la vue événementielle et la vue informationnelle ainsi que la vue fonctionnelle et dysfonctionnelle.
	Gestion du diagnostic	Oui, grâce à la manipulation de la vue événementielle et la vue informationnelle ainsi

		que la vue fonctionnelle et dysfonctionnelle.
	Gestion des stratégies	Oui, grâce à la manipulation de la vue de stratégie, la vue des interventions, la vue des processus et la vue informationnelle.
Questions techniques	Exécutable	Oui, possible grâce à l'API de PowerLoom.
	Charger les données	Oui, possible grâce à l'API de PowerLoom.
	Consistance	Oui, grâce au classificateur de description de PowerLoom.
	Équivalences	Oui, grâce au classificateur de description de PowerLoom.
	Reclassification	Oui, grâce au classificateur de description de PowerLoom.
	Inférence	Oui, grâce au classificateur de description de PowerLoom.
	Importer/Exporter des données	Oui, possible grâce à l'API de PowerLoom et grâce au PowerLoom GUI.
	Importer multiple modèles sous une seule source	Oui, possible grâce à l'API de PowerLoom et grâce au PowerLoom GUI.
	Fusionner des modèles	Oui/Non, ceci dépend de l'outil utilisé.
	Faire de simples calculs	Oui, grâce à la manipulation des concepts ainsi que leurs attributs et relations par le classificateur de description de PowerLoom.
	Restreindre toutes les classes	Oui, grâce au classificateur de description de PowerLoom.

4.6.2.2- Mise en exploitation de IMAMO

Nous avons testé une partie de IMAMO dans un environnement réel par son intégration dans la plateforme logicielle em@web qui gère la maintenance de l'équipement SISTRE un « Système Industriel Supervisé de TRansfErt de palettes » situé dans l'atelier du département AS2M de l'institut FEMTO-ST (voir Figure 4-9).

Dans cette section, nous présentons les tests d'IMAMO également effectués sur SISTRE. Ce dernier représente un système de production flexible et est composé de cinq postes de travail robotisés desservis par un système de transfert de palettes organisés en anneaux double (interne et externe). Chaque station est équipée d'actionneurs pneumatiques (pousseurs, les extracteurs et des indexeurs) et les actionneurs électriques (stoppeurs), ainsi qu'un certain nombre de capteurs inductifs (capteurs de proximité). Un module d'induction de lecture / écriture permet l'identification et la localisation de chaque palette et fournit des informations relatives à l'opération nécessaire à une station. Le déplacement des palettes est assuré par le frottement sur les courroies qui sont gérées par des moteurs électriques. Chaque palette porte une étiquette magnétique utilisée comme une mémoire embarquée qui

peut être lu grâce à chaque poste de travail à des modules magnétiques de lecture / écriture (Balogh) et permet la mémorisation de la séquence d'assemblage de produits.

Figure 4-9 The pallet transfer system "SISTRE"

Ainsi, ces étiquettes permettent la détermination de la trajectoire de la palette à travers le système. Les palettes sont véhiculées sur la bague intérieure qui permet de transiter entre les différentes stations. Lorsque la palette doit être traitée par un robot à la station de travail (informations lues sur l'étiquette de la palette), ce dernier est dévié sur la bague extérieure où se situe le poste de travail approprié. Ce dernier est donc situé sur la couronne extérieure et contient des actionneurs pneumatiques et électriques (extracteur, pousseur, indexeur, stoppeur) ainsi que des capteurs inductifs (Rasovska I. , 2006).

Le code PowerLoom suivant décrit l'instanciation de l'équipement physique SISTRE et de l'équipement modèle PLATEFORME, aussi l'instanciation des associations `PHYSICAL-EQUIPMENT-CONSTRUCTOR` "Bosch" de SISTRE, `EQUIPMENT-HAS-TOP-MODEL` de SISTRE qui est une PLATEFORME et l'association `EQUIPMENT-COMPONENT-COMPOSED` afin de décrire la composition de l'équipement physique SISTRE:

```
(ASSERT (PHYSICAL-EQUIPMENT SISTRE))
(ASSERT (PHYSICAL-EQUIPMENT-CONSTRUCTOR SISTRE "Bosch"))
(ASSERT (EQUIPMENT-MODEL PLATEFORME))
(ASSERT (EQUIPMENT-HAS-TOP-MODEL SISTRE PLATEFORME))
(ASSERT (EQUIPMENT-COMPOSED SISTRE ROBOT))
…
(ASSERT (EQUIPMENT-COMPOSED SISTRE CONVOYEUR))
(ASSERT (EQUIPMENT-COMPONENT-COMPOSED SISTRE CAMERA-DE-SERVEILLANCE))
(ASSERT (COMPONENT ENTRETOISE))
(ASSERT (EQUIPMENT-COMPONENT-COMPOSED CONVOYEUR ENTRETOISE))
(ASSERT (EQUIPMENT-COMPOSED CONVOYEUR COURROIE))
```

L'ensemble des concepts définis et les assertions (les instances) présentent la base de connaissances du domaine de la maintenance. Différentes règles peuvent être ajoutées afin d'enrichir cette base de connaissances et de donner plus de précision sur les concepts, tels que la définition de la règle CRITICAL-COMPONENT, définissant un composant critique comme un équipement physique ayant un degré fonctionnel (une propriété dans les équipement physiques définie comme un attribut numérique pour spécifier l'importance de cet équipement ou composant dans le fonctionnement de l'équipement ou le processus industriel) supérieure à 6:

```
(DEFCONCEPT CRITICAL-COMPONENT ((?P PHYSICAL-EQUIPMENT))
 :<<=>> (AND (PHYSICAL-EQUIPMENT ?P) (> (FUNCTIONNAL-DEGREE ?P) 6)))
```

L'expression précédente est équivalente aux deux expressions suivantes:

```
(DEFCONCEPT CRITICAL-COMPONENT ((?P PHYSICAL-EQUIPMENT))
(DEFRULE CRITICAL-COMPONENT (> (FUNCTIONNAL-DEGREE ?P) 6))
```

L'API Java de PowerLoom permet d'interroger la base de connaissances. Ceci permet de vérifier la cohérence du modèle ontologique et de vérifier l'exactitude des réponses de la requête. Dans cet exemple, nous demandons au moteur PowerLoom de fournir la liste de tous les équipements physiques. La réponse donnée n'est pas seulement l'équipement physique SISTRE mais ses composants aussi. Dans IMAMO, un équipement physique peut être composé d'éléments d'équipement physique.

```
PL-USER |= (load "ontologie-maintenance.plm")
PL-USER |= (in-module "ONTOLOGIE-MAINTENANCE")
ONTOLOGIE-MAINTENANCE |= (RETRIEVE ALL (PHYSICAL-EQUIPMENT ?PE))

There are 28 solutions:
  #1: ?PE=COURROIE
  #2: ?PE=CONVOYEUR
  #3: ?PE=DETECTEUR
  #4: ?PE=ACTIONNEUR
  #5: ?PE=SISTRE
  …
  #26: ?PE=BAL0
  #27: ?PE=TAP-EXT
  #28: ?PE=TAP-IN
```

Considérons un autre exemple sur une petite partie du modèle conceptuel d'IMAMO. Nous définissons une association entre les concepts `PHYSICAL-EQUIPMENT` et `INTERVENTION`; cette association est transformée comme la relation `Has-Intervention` in PowerLoom.

```
(DEFRELATION Has-Intervention ((?C PHYSICAL-EQUIPMENT) (? INTERVENTION)))
```

Connaissant la relation (i.e. association) entre `PHYSICAL-EQUIPMENT` et `DATA-ACQUISITION-SYSTEM` qui est reliée à `MEASURE`, ce dernier est relié `TRIGGER-EVENT` relié à `WORK-REQUEST` relié `aussi à INTERVENTION`.

Lors de l'interrogation du moteur PowerLoom pour obtenir toutes les interventions concernant l'équipement physique TEST02, le même résultat peut être obtenu avec deux requêtes différentes.

Première requête:

```
ONTOLOGIE-MAINTENANCE |= (RETRIEVE ALL (Has-Intervention TEST02 ?I))
There is 2 solutions:
  #1: ?I=INTERVENTION-5
  #2: ?I=INTERVENTION-9
```

Deuxième requête:

```
ONTOLOGIE-MAINTENANCE |= (RETRIEVE ALL (EXISTS (?D)
(AND    (EQUIPMENT-SCADA ?D TEST02)
       (EXISTS     (?M)     (AND          (DATA-ACQUISITION-SYSTEM-CATCH-MEASURE     ?D     ?M)
       (EXISTS          (?T)          (AND          (MEASURE-TRIGGER-EVENT          ?M          ?T)
       (EXISTS     (?WR)     (AND                          (WORK-REQUEST-ORIGIN     ?WR     ?T)
       (HAS-WORK-REQUEST ?I ?WR)))                                                          ))
       )) )) )
There is 2 solutions:
  #1: ?I=INTERVENTION-5
  #2: ?I=INTERVENTION-9
```

La seconde requête prouve que le lien entre le concept `PHYSICAL-EQUIPMENT` et le concept `INTERVENTION` peut être supprimé à cause de l'existence d'une relation implicite entre ces deux concepts. Ainsi, nous pouvons identifier la redondance présentée dans le modèle UML de l'ontologie grâce au moteur de raisonnement de PowerLoom.

4.6.2.3- Etudes de cas pour l'interopérabilité sémantique

En montrant deux exemples de cas d'utilisation, simples et concrets, concernant l'exploitation d'IMAMO relatif à l'interopérabilité sémantique, sachant que les cas complexes de cette interopérabilité sont gérés par le composant « médiateur » de la plateforme. Le premier exemple montre comment l'ajout dynamique de règles à la base de connaissances peut résoudre le problème de l'interopérabilité. Le second, quant à lui, montre comment l'interopérabilité est toujours assurée, même si IMAMO évolue localement.

4.6.2.3.1- Première étude de cas

Concernant le premier exemple, nous considérons deux sites : un site de production ayant un système de gestion des équipements appelé (PS) et un site de maintenance ayant un système de gestion de maintenance appelé (MS). Le premier système utilise le modèle MIMOSA-CRIS et le deuxième système utilise IMAMO. L'équipement SISTRE est situé dans le site de production. Cet équipement est inconnu dans le site de maintenance. SISTRE est tombé en panne, par conséquent, le manager de PS envoie une demande d'intervention au manager de MS via la plateforme de maintenance. Les experts et les techniciens qui utilisent MS ont besoin de toutes les informations sur la composition de cet équipement et sur les interventions précédentes effectuées sur l'équipement.

On rappelle qu'un équipement est modélisé par le concept ASSET dans le modèle MIMOSA-CRIS. Ainsi, dans IMAMO un équipement est défini par le concept Physical-Equipment.

Nous montrons, par une séquence d'instructions PowerLoom, comment le moteur de raisonnement peut résoudre ce problème et comment il parvient à obtenir les informations nécessaires demandées par le MS.

Tout d'abord, nous commençons par demander au système PowerLoom gérant l'ontologie IMAMO utilisée par le système MS de lister tous les équipements connus par ce système. D'où, savoir si SISTRE est connu comme équipement dans ce site ou pas:

```
ONTOLOGIE-MAINTENANCE |= (RETRIEVE ALL (PHYSICAL-EQUIPMENT ?PE))
There are 3 solutions:
  #1: ?PE=TEST02
  #2: ?PE=TEST01
  #3: ?PE=TEST
ONTOLOGIE-MAINTENANCE |= (ASK (PHYSICAL-EQUIPMENT SISTRE))
UNKNOWN
ONTOLOGIE-MAINTENANCE |= (ASK (not (PHYSICAL-EQUIPMENT SISTRE)))
UNKNOWN
```

Ainsi, le système MS ne sait pas que SISTRE est un équipement. Afin de garantir un échange compréhensible entre les systèmes PS et MS, il suffit d'ajouter à la base de connaissances une règle d'équivalence définissant ASSET comme un concept équivalent au concept PHYSICAL-EQUIPEMENT. Cela permettra au système MS d'obtenir toutes les informations sur toutes les instances du concept ASSET dans PS.

```
(DEFCONCEPT PHYSICAL-EQUIPMENT ((?A ASSET)))
(DEFCONCEPT ASSET ((?PE PHYSICAL-EQUIPMENT)))
```

Après la définition des règles d'équivalence, nous demandons au système MS si SISTRE est un équipement physique, le moteur de raisonnement répond par oui comme le montre la requête suivante:

```
ONTOLOGIE-MAINTENANCE |= (ASK (PHYSICAL-EQUIPMENT SISTRE))
TRUE
```

Ainsi, lorsque le système MS considère qu'un équipement physique comme ASSET, il peut obtenir par conséquent les informations concernant la composition de SISTRE.

```
ONTOLOGIE-MAINTENANCE |= (retrieve all (EQUIPMENT-COMPOSED SISTRE ?x))
There are 4 solutions:
#1: ?X=ROBOT
  #2: ?X=ACTIONNEUR
  #3: ?X=DETECTEUR
  #4: ?X=CONVOYEUR
```

La requête suivante permet de récupérer les interventions précédentes sur SISTRE:

```
ONTOLOGIE-MAINTENANCE |= (RETRIEVE ALL (EXISTS (?D) (AND (EQUIPMENT-SCADA ?D SISTRE)
(EXISTS (?M)
             (AND       (DATA-ACQUISITION-SYSTEM-CATCH-MEASURE   ?   D   ?M)   (EXISTS   (?T)
       (     AND         (MEASURE-TRIGGER-EVENT    ?M    ?T)                  (EXISTS    (?WR)
       (    AND        (WORK-REQUEST-ORIGIN    ?WR    ?T)        (HAS-WORK-REQUEST    ?I    ?WR)
       )  )   ))   )) )) )
There is 1 solution:
  #1: ?I=INTERVENTION-2
```

La requête suivante est presque la même requête que la suivante, mais elle fournit aussi les informations concernant l'origine des interventions:

```
ONTOLOGIE-MAINTENANCE   |=   (RETRIEVE   ALL   (AND       (EQUIPMENT-SCADA   ?D   SISTRE)
(AND  (DATA-ACQUISITION-SYSTEM-CATCH-MEASURE ?D ?M)                                    (AND
(MEASURE-TRIGGER-EVENT ?M ?T)                                    (AND    (WORK-REQUEST-ORIGIN
?WR ?T)                                  (HAS-WORK-REQUEST ?I ?WR)))) ))
```

```
There is 1 solution:
  #1: ?D=SCADA-1, ?M=M-1, ?T=TEMPERATURE-ELEVEE, ?WR=WORK-REQUEST-2, ?I=INTERVENTION-2
```

Les deux requêtes précédentes montrent l'efficacité du moteur de raisonnement de PowerLoom pour assurer l'interopérabilité sémantique entre les systèmes ayant des modèles différents.

4.6.2.3.2- Deuxième étude de cas

Concernant le second exemple de l'interopérabilité sémantique, nous considérons deux sites : S1 et S2 ayant deux systèmes de gestion adoptant l'ontologie IMAMO. Chaque système dans chaque site enrichit l'ontologie au niveau local afin de l'adapter à leurs besoins spécifiques.

Dans ce cas d'utilisation, nous prenons une partie de l'ontologie concernant les ressources matérielles présentée dans la Figure 4-10. Supposons que cette partie d'ontologie ne considère pas le concept lubrifiant dans les ressources matérielles.

Afin d'adapter l'ontologie à leurs besoins, les utilisateurs du système de maintenance de S1 ajoutent ce concept au modèle de données utilisé par leur système comme le montre la Figure 4-11. Ils définissent le concept ajouté à la base de connaissances par le biais de PowerLoom comme suit:

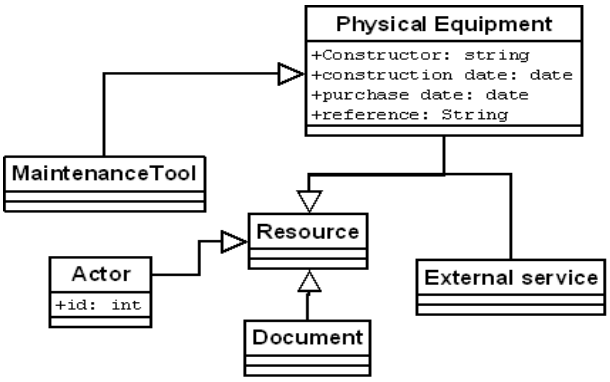


Figure 4-10 Une partie de l'ontologie des ressources matérielles

```
(DEFCONCEPT LUBRICANT (?C RESSOURCE))
(DEFCONCEPT RESSOURCE ::<<=>> OR (ACTOR T? OR (PHYSICAL-EQUIPMENT ?PE OR (DOCUMENT ? D OR
(EXTERNAL-SERVICE ?ES LUBRICANT ?L )))))
(DEFRELATION LUBRICANT-HAS-SPECIFICITY (LUBRICANT ?SPECIFICITY STRING))
(ASSERT LUBRICANT (And (Not ?ES EXTERNAL-SERVICE)(AND (NOT ?T TOOL)(AND(NOT ?PE PHYSICAL-
EQUIPMENT)(AND (NOT ?D DOCUMENT)(NOT ?A ACTOR))))))
```

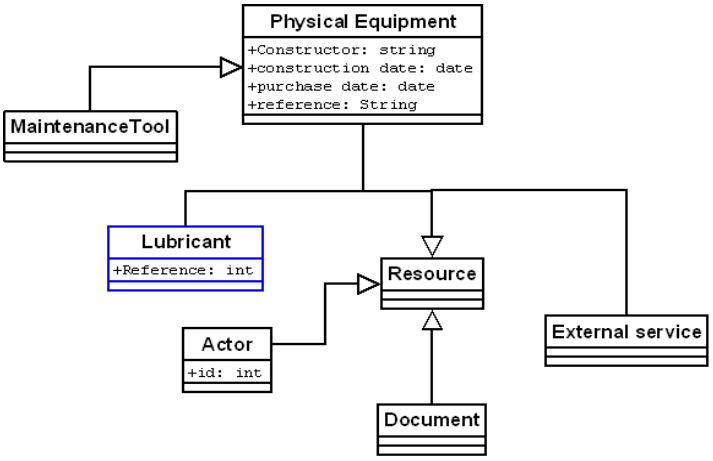


Figure 4-11 Ontologie de ressources matérielles de S1

La même action a été faite dans le site S2. Les utilisateurs du système de gestions de S2 ont ajouté le concept consommable « consumable » qui est le même que «lubrifiant» dans le domaine de maintenance tel que présenté dans la Figure 4-12. Le concept ajouté est défini dans la base de connaissances par le biais PowerLoom comme suit:

```
(DEFCONCEPT CONSUMABLE (?C RESSOURCE))
(DEFCONCEPT RESSOURCE ::<<=>> OR (ACTOR T? OR (PHYSICAL-EQUIPMENT ?PE OR (DOCUMENT ? D OR
(EXTERNAL-SERVICE ?ES CONSUMABLE ?C )))))
(DEFRELATION CONSUMABLE-HAS-SPECIFICITY (CONSUMABLE ?SPECIFICITY STRING))
(ASSERT CONSUMABLE (And (Not ?ES EXTERNAL-SERVICE)(AND (NOT ?T TOOL)(AND(NOT ?PE PHYSICAL-
EQUIPMENT)(AND (NOT ?D DOCUMENT)(NOT ?A ACTOR))))))
 (ASSERT (CONSUMABLE OIL))
```

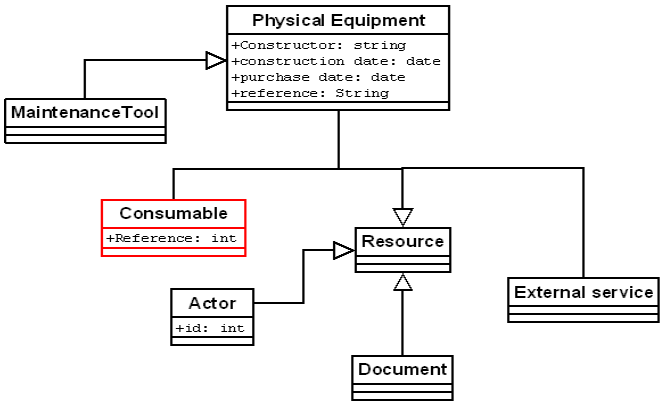


Figure 4-12 Ontologie de ressources matérielles de S2

Par conséquent, nous lançons quelques requêtes pour vérifier si le moteur de raisonnement de PowerLoom peut assurer l'interopérabilité entre S1 et S2 lors d'un échange concernant la ressource matérielle huile « oil » entre les deux systèmes des deux sites.

```
ONTOLOGIE-MAINTENANCE |= (ASK (RESSOURCE OIL))
TRUE
ONTOLOGIE-MAINTENANCE |= (RETRIEVE ALL (RESSOURCE ?MS))
There is 1 solution:
  #1: ?MS=OIL
ONTOLOGIE-MAINTENANCE |= (ASK (CONSUMABLE OIL))
TRUE
ONTOLOGIE-MAINTENANCE |= (ASK (LUBRICANT OIL))
TRUE
ONTOLOGIE-MAINTENANCE |= (RETRIEVE ALL (CONSUMABLE ?CN))
There is 1 solution:
  #1: ?CN=OIL
ONTOLOGIE-MAINTENANCE |= (RETRIEVE ALL (LUBRICANT ?LN))
There is 1 solution:
  #1: ?LN=OIL
```

Les requêtes présentées ci-dessus montrent que le conflit sémantique peut être résolu en exploitant le moteur de raisonnement PowerLoom. Le raisonnement appliqué à la base de connaissances contenant IMAMO peut déduire l'équivalence entre les deux concepts « Lubricant » et « Consumable». Cette équivalence déduite permet un échange interopérable entre les deux systèmes de S1 et S2.

4.6.2.4- Etude de cas sur l'exploitation de connaissances

Dans cette section, nous présentons un exemple simple décrivant l'exploitation des connaissances existantes afin d'en extraire de nouvelles connaissances. Cette étude de cas porte sur les connaissances des composants réutilisables après le démontage de SISTRE. En effet, les connaissances nécessaires pour trouver les composants réutilisables existent dans la base des connaissances mais de manière implicite. Ainsi, elles ne sont pas exploitées.

Par conséquent, il suffit d'éditer une nouvelle règle définissant un composant réutilisable. Puis, via le moteur de raisonnement PowerLoom, nous pouvons récupérer la liste des composants réutilisables dans Sistre. Les commandes suivantes définissent un composant réutilisable comme un composant qui dispose d'une période de fonctionnement de moins de 60000 heures ou moins de trois modes de fonctionnements de type panne. Nous considérons que la période de fonctionnement d'un composant est la même que celle de l'équipement physique. La Figure 4-13 montre une partie de l'ontologie utilisée pour définir cette règle.

```
DEFRELATION REUSABLE-COMPONENT-OF ((?PE PHYSICAL-EQUIPMENT) (?COM PHYSICAL-EQUIPMENT))  :=>
       (OR
              AND(
                     (HAS-EXPLOITATION-MODE ?COM ?EM)
                     AND ((HAS-FUNCTIONAL-PERIOD ?EM ?FP)
                     (> (SUM (NUMBER-HOURS ?FP ?NH))60000)
                     )              )
              (AND (HAS-PERIOD ?EM ?P)
              AND ((DURING ?P ?OM)
                     AND((TRUE(FAILURE-STATE ?OM)) (>(COUNT(STATE_ID ?OM ?X))3)
                     )              )))
```

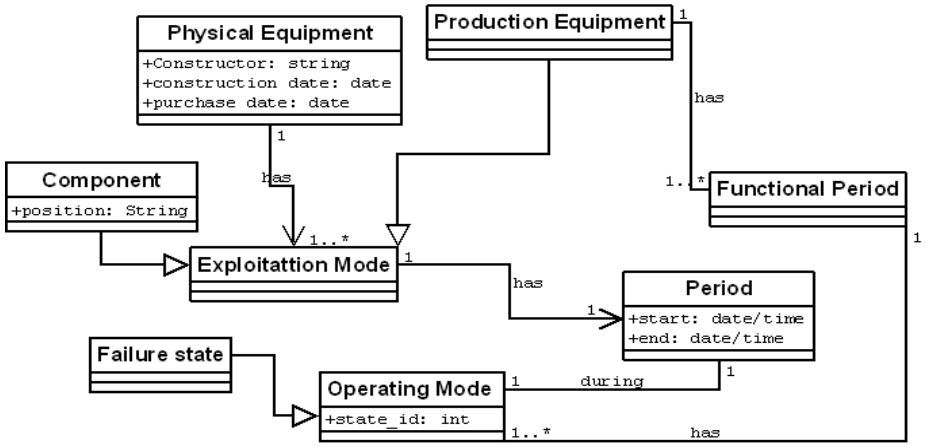


Figure 4-13 La partie de l'ontologie utilisée dans la défintion de composants réutilisables

Après avoir défini la règle des composants réutilisable, la requête suivante permet d'obtenir la liste de tous les composants réutilisables de SISTRE:

```
ONTOLOGIE-MAINTENANCE |= (RETRIEVE ALL (REUSABLE-COMPONENT-OF SISTRE ?RC))
There is 1 solution:
  #1: ?RC= ACTIONNEUR
```

D'où, nous notons que «*ACTIONNEUR* » est le seul composant réutilisable de Sistre. Cette information était inconnue, mais grâce à la règle définie - qui peut être utilisée pour tout équipement physique - de nouvelles connaissances sur la réutilisation des composants peuvent être extraites de la base de connaissances. Ce type de connaissances extraites donne une valeur ajoutée au système de maintenance afin d'appuyer les décisions et réduire les coûts.

4.7- Directives pour la maintenance de l'ontologie

Selon Yildiz (Yildiz, 2006), la maintenance et le changement des ontologies peuvent prendre plusieurs formes : la modification, le *versioning* et l'évolution.

Il est à noter que la définition des opérations de maintenance des ontologies n'est pas facile (Yildiz, 2006), vu que tous les effets résultants possibles des composants d'une ontologie doivent être pris en compte lors d'un changement. Klein identifie trois types de changements (Klein, 2004) :

- Les changements conceptuels (par l'évolution des relations et des concepts),
- La spécification (par l' ajout de nouvelles propriétés [attributs] à un concept),
- La représentation (par la formalisation etou les phases de mise en œuvred'un autrelangage de représentation).

Par conséquent, l'état présent de IMAMO peut évoluer en fonction de ces trois types de changement: conceptuelle et spécification. Ainsi, comme invoqué dans la section d'évaluation, l'une des principales évolutions

nécessaires pour IMAMO est l'ajout d'attributs pour les concepts. En outre, comme signalé dans la section d'intégration, d'autres ontologies peuvent être intégrées et réutilisées pour enrichir IMAMO.

Dans le cadre de ces objectifs, une initiative a été lancée afin de créer un site web pour IMAMO. Cela permettra à l'ontologie d'être partagée avec les communautés universitaires et industrielles. Il permettra aussi d'adopter l'ontologie sur une grande échelle et ainsi initier l'évolution et la maintenance collective.

D'autre part, Klein définit la gestion des versions d'une ontologie, comme «la capacité à gérer les changements de l'ontologie et de leurs effets par la création et le maintien de différentes variantes de l'ontologie» (Klein, 2004). En effet, cette forme de changement est très importante dans le cycle de vie de IMAMO considérant que les principaux avantages de cette ontologie sont sa réutilisation et son interopérabilité induite dans la plateforme. Cette fonctionnalité est essentielle tant que l'ontologie est générique et sera utilisée par différentes applications dans le domaine de maintenance. Deux cas sont à noter :

- Premièrement, de nouvelles versions d'IMAMO peuvent être partagées entre toutes ces applications.
- Deuxièmement, chaque application de maintenance développe sa propre version locale d'IMAMO.

Malgré la différence entre ces versions, l'interopérabilité entre ces versions doit être assurée. En effet, ceci est en concordance avec le principe du médiateur proposé dans l'architecture de la plateforme et son principe de fonctionnement afin d'assurer l'alignement des différentes versions de l'ontologie.

5. Conclusion

Le rôle important de la base de connaissances dans la plateforme de s-maintenance tourne autour de deux axes principaux à savoir l'interopérabilité sémantique et l'exploitation et la réutilisation des connaissances. Pour avoir une base de connaissances solide et partageable ayant une plus large utilisation que les bases de connaissances traditionnelles, les ontologies sont présentées comme la solution potentielle qui explique la conceptualisation du monde réel par des concepts fondamentaux.

En Constatant l'absence d'une ontologie opérationnelle de maintenancemalgré la présence de standards dans le domaine comme MIMOSA-CRIS et de modèle de connaissance issu de PROTEUS, nous avons proposé dans ce chapitre la construction d'une ontologie du domaine pour qu'elle soit la base sur laquelle se fonde la plateforme de s-maintenance. Nous avons appelé l'ontologie proposée IMAMO (Industrial MAintenance Management Ontology).

Un état de l'art sur les méthodologies de création d'ontologies effectué par rapport à un ensemble de critères dont par exemple le type, la grandeur de l'ontologie, le degré de formalisation et la stabilité de la méthodologie, nous avons choisi la méthodologie METHONTOLOGY. Cette méthode réputée par sa stabilité et sa maturité par rapport aux autres méthodologies est bien structurée et couvre tout le cycle de vie de développement d'une ontologie.

Un état de l'art sur les différents langages et outils nous a amené à choisir le langage de description PowerLoom ainsi que son API java pour l'activité de développement de IMAMO.

En outre, dans le cycle de vie de IMAMO les activités de support « acquisition des connaissances » et « intégration » nous ont permis de bien opérer les activités de développement « spécification » et « conceptualisation ». Pour ces activités de support nous nous sommes basés sur les travaux antérieurs concernant les standards et les projets effectués dans le domaine de la maintenance comme MIMOSA-CRIS, PROTEUS, le modèle-SMAC ou ceux qui ont des liens avec la maintenance comme le modèle SOM dans le projet PROMISE ainsi que les ontologies dans la maintenance de logiciel

Par rapport à l'activité de conceptualisation, nous avons choisi le diagramme de classe UML pour une conceptualisation semi formelle motivée par l'expressivité de ce langage.

Ainsi, pour avoir une représentation plus lisible, et un modèle compréhensible, notamment sur le plan des relations entre concepts, nous avons décomposés le diagramme de classe en huit vues selon leurs points de focalisation, à savoir : la vue structurelle, la vue fonctionnelle et dysfonctionnelle, la vue événementielle qui présente les événements déclencheurs lancés après les échecs et / ou la dégradation, liée à au diagnostic et l'expertise; la vue informationnelle (documents, humains, logiciels, outils, indicateurs, etc.), la vue interventionnelle, la vue stratégique, la vue des processus et finalement la vue mémorielle.

Pour transformer le modèle conceptuel en un modèle formel, nous avons utilisé un système de présentation qui a permis d'encoder les diagrammes de classer dans ALQHI (une variante de description logique), après cela, nous avons implémenté ce modèle formel en utilisant la version 3.2.0 du langage PowerLoom pour sa stabilité par rapport à la dernière version de ce langage.

D'autre part, même si l'activité d'évaluation supporte la proposition d'une ontologie de qualité aussi bien conceptuelle, opérationnelle et fonctionnelle, néanmoins, ces différents niveaux d'évaluation ne permettent pas de garantir l'adéquation entre les attentes de l'utilisateur lors de l'exploitation de cette ontologie. En effet, c'est à l'usage, que les performances ou les manques de l'ontologie transparaissent.

Ainsi, sachant qu'il n'existe pas de manière standardisée, objective et largement acceptée des méthodes d'évaluation d'ontologie, nous avons pris appui sur les quatre types d'approches identifiés par Brank et al pour donner des éléments de réponses, les plus complètes permettant d'évaluer l'ontologie IMAMO.

Premièrement, une évaluation de la qualité du modèle conceptuel par rapport à des métriques qualifiant la conceptualisation de l'ontologie nous a permis de conclure que notre ontologie n'est pas une hiérarchie en tenant un équilibre entre les relations d'héritages et les autres types de relations ; elle est hybride en tenant un équilibre entre la généralité et l'explicité et elle est pauvre en terme d'attributs .

Deuxièmement, une évaluation métier axée sur la valeur ajoutée de l'ontologie en faisant appel respectivement à l'approche d'évaluation par humain et l'approche par application. La première a porté sur la vérification des fonctionnalités de IMAMO via une méthode questions / réponses, nous a permis d'observer que IMAMO répond aux exigences métier de la maintenance. Ainsi, nous avons pu observer la richesse en termes de manipulations possibles grâce au classificateur de PowerLoom. Quant à la deuxième, qui a porté sur l'évaluation de l'applicabilité et de l'exploitation des connaissances en exécutant des requêtes et en évaluant leurs résultats, elle

nous a permis d'observer les possibilités de garantir une interopérabilité sémantique soit en cas de *« versionning »* d'ontologie soit par l'ajout de règles d'équivalence dans le cas d'ontologies différentes.

Lors de la dernière activité de développement dans METHONTOLOGY qui est l'activité de maintenance, nous avons discuté l'évolution de IMAMO que se soit au niveau de spécification ou au niveau du *versionning.*

Après avoir développé cette ontologie, les questions qui se posent sont *« comment cette ontologie sera exploitée pour fournir les services dynamiques envisagés par la plateforme de s-maintenance ?» et « quelles sont ses valeurs ajoutées pour l'utilisateur ?».* Les réponses à ces questions feront l'objet du prochain et dernier chapitre.

Chapitre 5 Développement et validation des fonctionnalités de la plateforme de s-maintenance

1. Introduction

Une des caractéristiques de la plateforme de s-maintenance est de proposer des services dynamiques et d'assurer des fonctionnalités (auto-traçabilité, auto-apprentissage, autogestion, etc.) susceptibles de faire évoluer l'intelligence et les comportements de la plateforme en tenant compte de l'aspect dynamique des connaissances. En effet, la dynamicité de la connaissance est un enjeu essentiel en ingénierie des connaissances. Bachimont (Bachimont, 2004) souligne que l'ingénierie des connaissances manipule des inscriptions de connaissances qui permettent elles-mêmes « l'expression, la transmission et l'appropriation de la connaissance proprement dite ». Cette inscription de connaissances doit se prêter à une interprétation critique et à une (re)construction de la connaissance.

Ce chapitre est consacré à la mise en place de fonctionnalités de remise à jour de connaissances, en fonction des interactions entre composants et expérience des utilisateurs, plus précisément des traces numériques d'interaction entre utilisateur et plateforme. Ces traces nous permettront de comprendre les comportements des utilisateurs : chaque utilisateur doit mentionner pour quoi l'a-il fait ? Par conséquent, nous envisagerons de formaliser les décisions prises par rapport aux contraintes, afin de comprendre et réutiliser les expériences passées. Ces traces d'activités sont assurées par les applications (les services) de la plateforme. Elles permettront de formaliser dans la base de connaissances l'expérience associée à l''exécution des activités, ses entrées et ses sorties ainsi que les contraintes subies. Ces connaissances dans la base de connaissances permettent à ces applications (application de diagnostic, application de pronostic, etc.) d'avoir un retour d'expérience sur leurs fonctionnements. Ce retour d'expérience exploité par ces applications aide à proposer des services dynamiques faisant évoluer leur comportement en fonction de leurs expériences.

Les traces numériques d'interaction associées à un modèle, seront définies par Mille comme des traces modélisées. Alain Mille dans (Laflaquière, Prié, & Mille, 2008) défend que les traces puissent prétendre au statut d'inscription de connaissance. Il dit que : « en s'ouvrant à l'interprétation d'un observateur, les traces modélisées ne sont pas seulement une présentation de l'activité pour elle-même, elles deviennent potentiellement une ressource de cette activité au sein de l'environnement observé. »

Partant de ce postulat, nous développons les fonctionnalités d'auto-traçabilité, d'auto-apprentissage et d'autogestion en prenant appui sur ces inscriptions de connaissances, ceci à l'aide d'une ingénierie susceptible d'exploiter l'aspect dynamique des connaissances, et de réutiliser ces inscriptions à partir de l'expérience passée de ses utilisateurs, l'ingénierie des traces.

L'ingénierie des traces, faisant partie de l'ingénierie des connaissances, fournies une nouvelle forme de réutilisation d'expérience et nous semble la mieux adaptée pour fournir un support aux fonctionnalités d'auto-X par l'aspect dynamique des connaissances utilisées dans la plateforme et par conséquent des services.

A ces fins, nous proposons d'élaborer un système à base de traces (SBT) permettant de tracer les activités effectuées via la plateforme. En effet, l'analyse de ces traces a pour objectif de fournir un retour d'expérience, sous forme de règles de connaissances aussi bien pour l'utilisateur de la plateforme que pour les applications qu'elle intègre.

Ceci est réalisé via un SBT adapté à la s-maintenance, son architecture sera fourni dans la deuxième section. L'architecture de ce système prend appui sur l'ontologie de maintenance, et plus particulièrement la partie réservée au processus relatif aux différentes activités dans la plateforme. En effet, dans ce chapitre, nous faisons un zoom sur certains composants de l'architecture de s-maintenance : base de connaissances, le raisonneur, le coordinateur et le gestionnaire de traces modélisées.

La collaboration entre ces composants assurera les fonctionnalités d'auto-traçabilité et d'auto-apprentissage qui seront développées successivement dans les sections 3 et 4.

La section 5 porte sur l'exploitation des sorties (outputs) du SBT ainsi formé au profit de l'utilisateur et de la plateforme. La fonctionnalité d'autogestion sera développé et contribuera à créer un service de diagnostic dynamique.

Finalement, nous étudions dans la section 6 l'impact et la valeur ajoutée de ces fonctionnalités sur les performances de la maintenance et ceci par le biais d'un calcul d'indices de temps et de coûts. Une étude de cas en rapport avec l'entreprise TEM sera simulée.

2. Système à base de trace dans la plateforme de s-maintenance

2.1- Système à base de traces (SBT)

Une trace d'interaction est définie comme un objet contenant la représentation d'une partie de l'expérience entre un système et ses utilisateurs. De manière générale, une trace d'interactions peut contenir tout ce qui est en rapport avec l'activité menée par les utilisateurs sur un système (Cram, Jouvin, & Mille, 2007). Champin et al (Champin, Prie, & Mille, 2004) définissent la trace comme étant une séquence d'états et de transitions représentant l'activité de l'utilisateur. Il existe différents termes qui se réfèrent au concept de trace d'interactions comme les logs, les historiques, les traînées et les tracks (Mille, 2006). Laflaquière et al (Laflaquière, Prié, & Mille, 2008) considèrent que ces traces sont un support potentiel d'une expérience d'utilisation.

Ces traces d'interactions peuvent être réutilisées à deux fins : l'assistance et l'analyse. L'analyse des traces consiste à analyser l'activité qui a généré les interactions (Mille, 2006) sachant qu'il existe différentes techniques d'analyse comme la visualisation des données tracées (Rossi, Lechevallier, & El Golli, 2005), la transformation des traces (Georgeon, Mille, & Bellet, 2006), la détection de séquences fréquentes, etc.

En effet, la réutilisation des traces d'interactions est une approche de la réutilisation de l'expérience fondée sur des expériences passées. Ce type d'approche rentre dans le cadre de la deuxième génération de système expert qui ne se base pas que sur des connaissances générales mais qui dispose aussi d'un nouveau type de connaissance qui est l'expérience (Cram, Jouvin, & Mille, 2007).

Les traitements à appliquer aux traces, sont formalisés à l'aide de Systèmes à Base de Traces (SBT) (Cram, Jouvin, & Mille, 2007), système qui s'appuie sur trois phases principales : la collecte, souvent suivie d'une étape de prétraitement, l'analyse et l'exploitation (Bousbia, 2011). Afin de faciliter la mise en place d'un SBT, Settouti

et al, ont proposés dans (Settouti L. , Prié, Mille, & Marty, 2006) une architecture composée de trois systèmes interdépendants, les systèmes de collecte, de transformation et de visualisation (voir Figure 5.1).

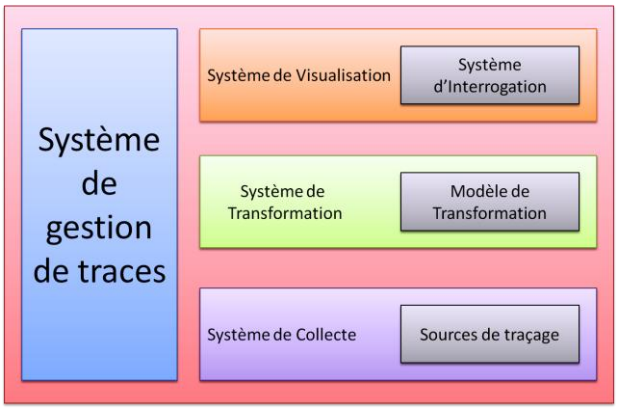


Figure 5-1 Architecture d'un SBT

Le système de collecte capte les interactions par l'intermédiaire de sources de traçage en créant une première trace. Ce système structure les traces collectées suivant une hiérarchie de classes observées appeler le modèle de trace (Cram, Jouvin, & Mille, 2007).

Le système de transformation constitue le cœur du SBT. Il permet de générer de nouvelles connaissances à partir des traces collectées. Le choix du modèle de transformation à appliquer dépend de l'objectif d'utilisation de cette trace.

L'ensemble des traces collectées et transformées est alors accessible par l'intermédiaire d'un système de visualisation (Bousbia, 2011) permettant d'exploiter, analyser et interpréter celles-ci (Cram, Jouvin, & Mille, 2007) et (Settouti L. , Prié, Mille, & Marty, 2006).

2.2- Système à base de traces adapté à la s-maintenance

Notre objectif est l'extraction des règles de connaissances à partir de l'activité de la plateforme. Le cas des activités portant sur les tâches décisionnelles prises par l'utilisateur semble intéressant. A cet effet, nous proposons un système à base de traces qui ne se limite pas uniquement à la visualisation, mais qui a l'ambition d'interpréter ces traces via les connaissances de la plateforme et ses modules d'apprentissage.

Pour répondre à cet objectif, l'architecture du SBT sera composée de deux systèmes principaux, un pour l'auto-traçabilité et l'autre d'auto-apprentissage. Le système d'auto-apprentissage est lui-même composé de deux systèmes, un concernant la collecte de trace et l'autre sa transformation.

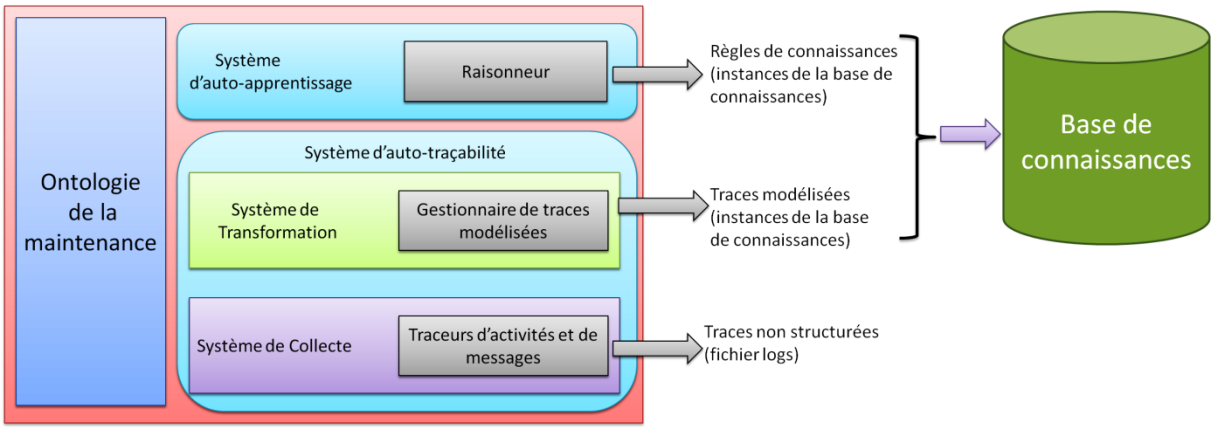


Figure 5-2 SBT adapté à la s-maintenance

Ce SBT prend appui sur l'ontologie du domaine de maintenance pour modéliser les traces, les interpréter et en extraire de la connaissance. Comme le montre la Figure 5.2, l'ontologie servira à tout niveau.

Le système de collecte est dirigé par le composant traceur d'activités et de messages qui collecte les traces d'interactions des activités et fournit des fichiers logs contenant des traces non structurées.

Soulignons que le système de transformation est dirigé par le composant gestionnaire des traces modélisées qui transforme les traces non structurées en traces modélisées à partir de l'ontologie, ensuite, il les enregistre comme instance dans la base de connaissances.

Remarquons que, nous avons ajouté aux systèmes de visualisation d'interrogation, un système d'interprétation et d'apprentissage. Ce système permet d'inférer de nouvelles règles de connaissances sur les activités des processus gérées par la plateforme, à partir des traces déjà enregistrées dans la base de connaissances.

La Figure 5.3 illustre le fonctionnement étape par étape de notre SBT en présentant les entrées et les sorties des différentes fonctionnalités impliquées dans ce système.

- **Etape 0 :** Grâce aux connaissances enregistrées dans la base de connaissances concernant l'exécution des processus (composés d'activités), **la fonctionnalité de gestion** orchestre l'exécution des opérations de la plateforme (labels 0, 0'et 0'').
- **Etape 1 :** Lors des interactions des opérateurs de maintenance avec la plateforme et l'exécution des activités dans celle-ci, ***la fonctionnalité de traçabilité*** collecte les traces d'interactions et les enregistre dans des fichiers logs (labels 0'' et 1).
- **Etape 2 :** ***la fonctionnalité de transformation*** récupère les fichiers logs, elle extrait des traces modélisées par un mapping avec le modèle de trace (l'ontologie de maintenance dans notre cas) et finalement elle enregistre ces traces modélisées comme de nouvelles instances dans la base de connaissances (labels 1 et 2).
- **Etape 3 :** ***la fonctionnalité d'interprétation et d'apprentissage*** récupère les instances concernant les traces modélisées des processus et les analyse en vue d'extraire de nouvelles connaissances sur ces processus sous forme de règles (label 3 et 4).

- **Etape 4 :** ***la fonctionnalité de validation et d'adaptation*** traite les règles issues de l'apprentissage pour les modéliser de façon qu'elles respectent la structure de la base de connaissance avant de les enregistrer dedans comme de nouvelles instances (label 4 et 5).

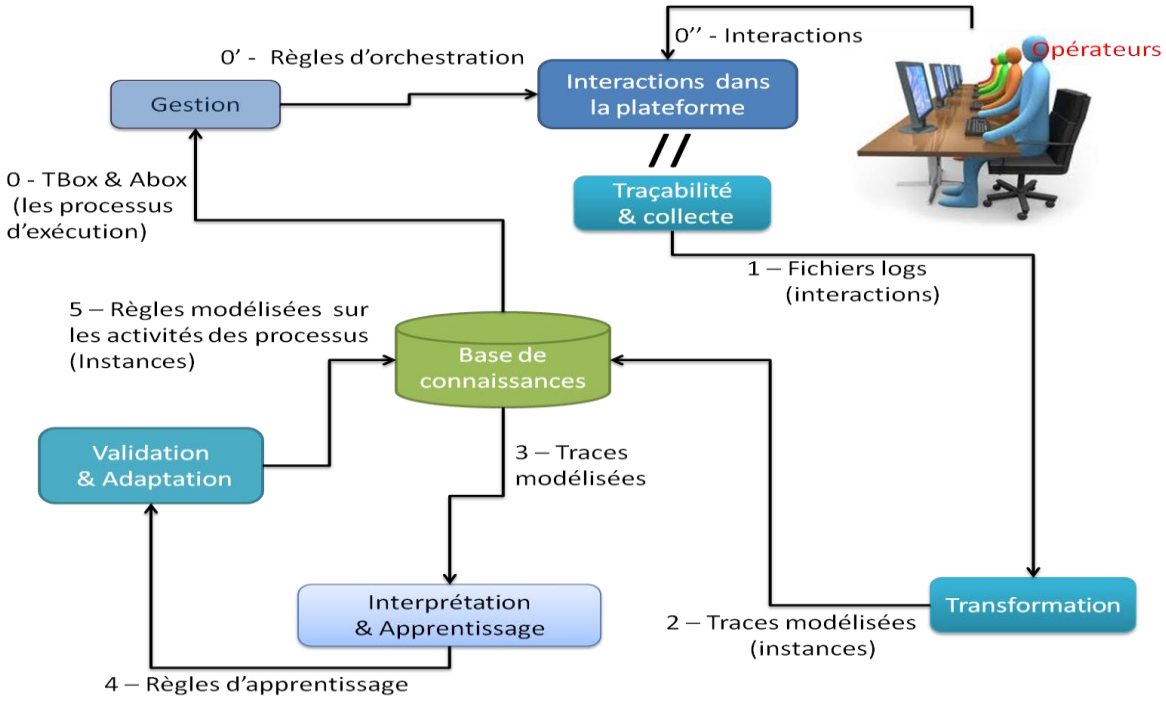


Figure 5-3 Boucle fermée du fonctionnement du SBT

3. Système d'auto-traçabilité : Collecte et transformation de traces

Le processus de traçabilité consiste à collecter l'ensemble des traces enregistrables en utilisant un environnement informatique sauvegardant les activités des opérateurs de maintenance. L'enregistrement de ces traces peut se référer à différents formats comme les fichiers logs, les *tracks* ou les *trails*. Selon Cram et al, les logs désignent des fichiers dans lesquels sont stockées au format textuel des traces de chaque requête ou opération effectuée sur un système, ou chaque erreur qu'il rencontre. Les logs représentent l'histoire d'interaction, mais comporte un certain nombre de défauts lors de la réutilisation de l'expérience (Cram, Jouvin, & Mille, 2007). Les *trails* sont notamment utilisés dans un contexte professionnel pour désigner la traçabilité d'un système, où le système a été conçu avec un mécanisme d'écoute sur lequel il est possible de se greffer. Quant aux *tracks*, ils désignent généralement des objets intentionnellement structurés pour représenter l'expérience d'une activité passée (Choquet & Iksal, 2007).

Selon (Bousbia, 2011) la collecte des données fournit des données qualifiées de « traces brutes » ou encore « traces primitives », difficilement exploitables en tant que telles. La création des traces à partir de ces logs est un processus complexe qui nécessite de nombreuses opérations (filtrage, recomposition en sessions, etc.). En effet, les logs peuvent contenir de grands volumes d'informations, parfois sans le contexte dans lequel les informations

ont été générées. Les données stockées ne sont pas structurées et pour la plupart inadaptées à l'objectif souhaité (Cram, Jouvin, & Mille, 2007).

Pour pouvoir mettre en pratique la réutilisation de l'expérience d'interaction, il faut donc pouvoir modéliser les traces (Cram, Jouvin, & Mille, 2007). Dans ce cadre, Settouti et al définissent formellement une trace comme « une collection d'observés qui peuvent être temporellement situés ». Un observé peut être considéré comme tout élément de l'environnement de l'utilisateur (une entité, une action) qui fait sens dans l'observation de son activité (Settouti L.-S. , Prié, Marty, & Mille, 2007).

Selon (Cram, Jouvin & mille, 2007), chaque observé de la trace est une instance de classe d'observé appartenant à une structure hiérarchique de classes d'observées appelées le modèle de trace. En outre, les observés de la trace sont reliés entre eux via les relations instances définies dans le modèle de trace. Ce modèle de trace peut être lié à une ontologie définissant les interactions utilisateur/système.

Pour tracer une activité à partir de la plateforme de s-maintenance, nous devons associer un modèle de trace. Or, nous savons que le modèle de processus de IMAMO (vue de processus) peut être exploité en tant quemodèle de trace pour les activités. Cette vue de processus sera développée dans la section 3.2 juste après la section 3.1 portant sur le fonctionnement du processus de traçabilité du système d'auto-traçabilité.

3.1- Processus de traçabilité du système d'auto-apprentissage

Le système d'auto-traçabilité consiste à collecter l'ensemble des traces enregistrables en utilisant un environnement informatique sauvegardant l'activité de l'opérateur de maintenance. Cet environnement informatique doit donc suivre des processus d'exécution dans la plateforme de s-maintenance. Comme déjà mentionné, les fichiers logs sont un bon outil pour sauvegarder les traces mais ils sont contraints par la complexité de leur structure et de leur analyse. Pour cela, dans le processus de traçabilité (collecte et transformation) portant sur le fonctionnement de ce système que nous présentons à la Figure 5-5, nous exploitons les fichiers logs ainsi que les traces modélisées. Donc, il y a une transformation du fichier logs à une trace modélisée par différents modules et finalement ces traces modélisées sont enregistrées dans la base de connaissance. La Figure 554 fournie un exemple d'instances de traces modélisées enregistrées dans la base de connaissances en langage PowerLoom.

```
(Assert (Activity A235))
(Assert (has-Reference-step A235 Validate-Alarm))
(Assert (ActivityInPutOutPut AIO569))
(Assert (has-Reference-Transition AIO569 Alarm.Measure))
(Assert (has-Input A235 AIO569))
(Assert (ActivityInPutOutPut AIO589))
(Assert (has-Reference-Transition AIO589 Alarm-Valid))
(Assert (has-Output A235 AIO589))
```

Figure 5-4 Exemple de mise à jour de la base de connaissances

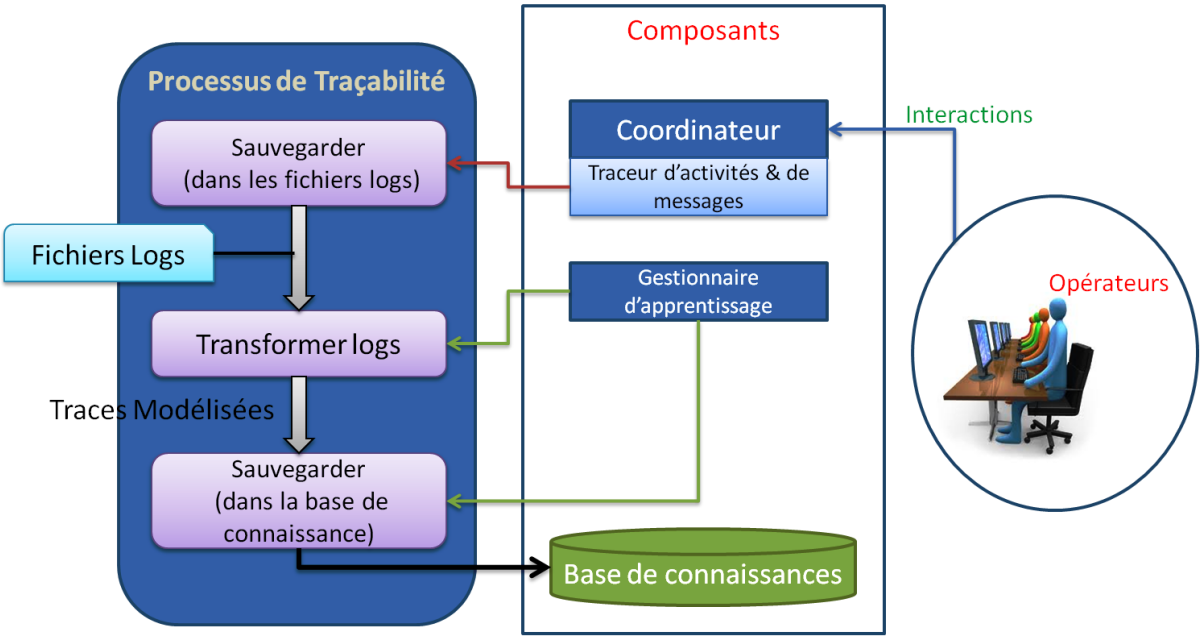


Figure 5-5 Activités du processus de traçabilité

La Figure 5-5, illustre que le *«Traceur d'activités et de messages »* est le sous composant du *Coordinateur* qui prend en charge la collecte des traces dans les fichiers logs pour les transférer ensuite au composant « Gestionnaire de traces modélisées). Ce dernier prend en charge la transformation des traces brutes des fichiers logs en traces exploitables par l'extrait des concepts du modèle de trace. De plus, ce modèle permet la structuration et la sauvgarde dans la base de connaissances comme de nouvelles instances.

3.2- Modèle de trace dans IMAMO

Cram et al affirment que les processus d'interaction entre l'humain et la machine se font à plusieurs niveaux : des relations de composition peuvent exister entre les interactions de différents niveaux (e.g., un projet est fait de réunions, des réunions sont composées de prise de parole) (Cram, Jouvin, & Mille, 2007). Hilbert et Redmiles classent ces interactions selon six niveaux d'abstraction (Hilbert & Redmiles, 2000) allant des manipulations de la souris ou du clavier jusqu'aux traces à analyser aux niveaux des tâches relatifs au domaine. Le niveau le plus abstrait (niveau 6) décrit des interactions relatives aux tâches (activités) réalisées (e.g., faire un diagnostique, corriger un bug, etc.). Le niveau que nous exploitons dans notre SBT car l'objectif du processus de traçabilité est d'apprendre à partir des activités des processus. La collecte sera donc faite sur ce sixième niveau. Ainsi, le modèle de trace que nous devons mettre en place pour le processus de transformation doit porter sur les activités des processus de maintenance gérées par les utilisateurs de la plateforme.

En effet, l'idée de conceptualiser les processus et les activités dans la « vue des processus de IMAMO » est motivée par le besoin d'avoir une conceptualisation des interactions métier pour les enregistrer dans la base de connaissances et les exploiter dans la phase d'analyse des traces grâce au moteur de raisonnement.

3.2.1- Vue de processus

La Figure 5-6 présente une partie de cette vue de processus et le tableau 1 présente le dictionnaire de données (la vue globale des processus est présentée dans l'annexe ONTOL ainsi que le dictionnaire de données détaillé).

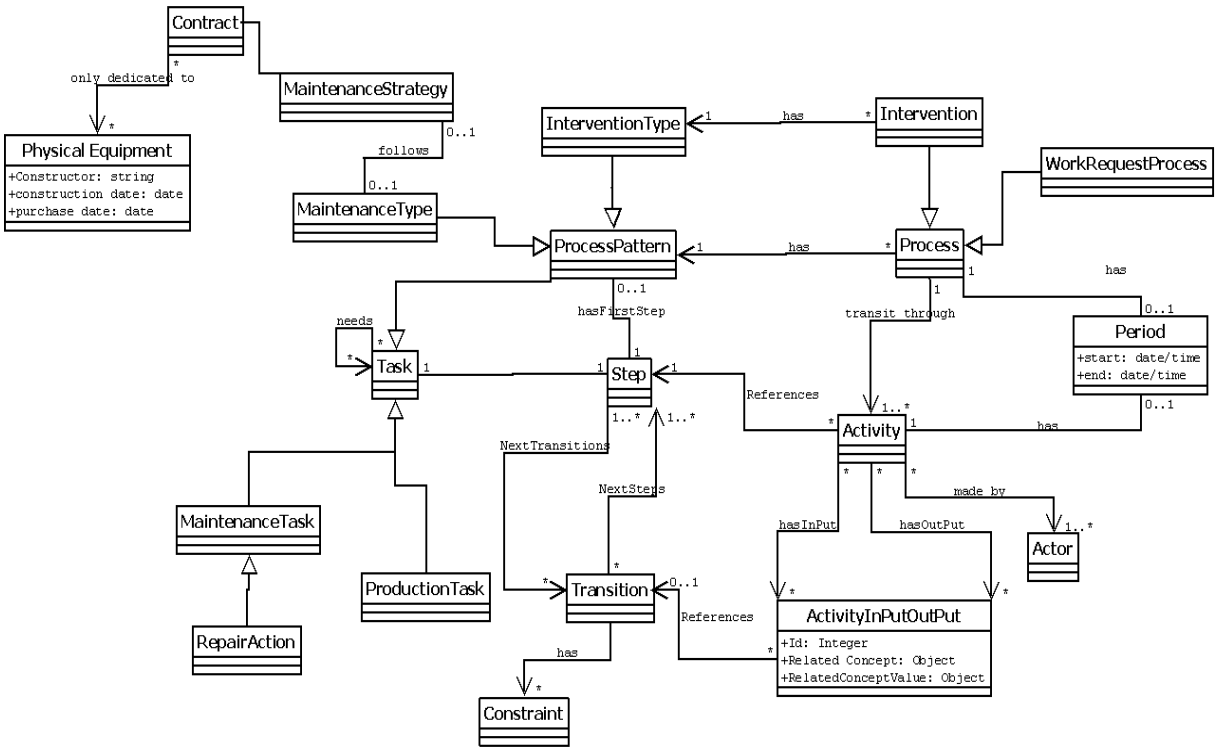


Figure 5-6 Vue des processus dans IMAMO

Tableau 5-1 Dictionnaire de données de la vue des processus

Concept	Termes français	Termes anglais	Description
process;	Processus;	Process;	Séquence d'activités interdépendantes et liées, qui, à chaque étape, consomme une ou plusieurs ressources pour convertir les entrées en sorties, tout en respectant un modèle de processus (le modèle de processus présente le modèle général du processus). Ces sorties de transition servent comme entrées de transition pour l'étape suivante jusqu'à ce qu'un but connu soit atteint.
process pattern;	Modèle de processus; Pattern de processus;	Process pattern;	Motif qui décrit une série d'actions. Un modèle est une description d'une solution générale à un problème commun ou une question à partir de laquelle une solution détaillée à un problème spécifique peut être déterminée.
maintenance type;	Type de maintenance;	Maintenance type;	Type spécifique de modèle de processus concernant la méthode pour effectuer la maintenance et l'orchestration des processus utilisés en vue d'atteindre les objectifs de la stratégie de maintenance. Il ya 12 types possibles de maintenance qui sont: maintenance préventive, maintenance planifiée, maintenance prédéterminée, maintenance basée sur les conditions, maintenance prédictive, maintenance corrective,

			maintenance à distance, maintenance différée, maintenance immédiate, maintenance en ligne, maintenance sur site et maintenance de l'opérateur. (la traduction en anglais est la suivante : Preventive maintenance, Scheduled maintenance, Predetermined maintenance, Condition based maintenance, Predictive maintenance, Corrective maintenance, Remote maintenance, Deferred maintenance, Immediate maintenance, On line maintenance, On-site maintenance and Operator maintenance.)
maintenance task;	Tâche de maintenance;	Maintenance task;	Type de tâche spécifique concernant une partie des travaux de maintenance (par exemple, réparation, remplacement, inspection, lubrification, etc.)
intervention type;	Type d'intervention;	Intervention type;	Type spécifique de modèle de processus. Il présente le modèle générique d'une intervention.
task;	Tâche;	Task;	Partie de travail effectuée par un acteur dans le cadre de ses fonctions.
repair action;	Action de reparation;	Repair action;	Type spécifique de tâches de maintenance définie comme une action physique prises pour rétablir la fonction requise d'un équipement physique défectueux.
production task;	Tâche de production;	Production task;	Type de tâche spécifique qui se termine par un produit discret ou un résultat qui est observable et mesurable.
constraint;	Contrainte;	Constraint;	Condition restrictive pour contrôler les transitions entre les étapes.
activity;	Activité;	Activity;	Unité d'organisation à exécuter une action spécifique. Une activité est l'exécution d'une tâche que ce soit une activité physique ou l'exécution de code. Il présente l'activité exercée par l'acteur dans le monde réel.
step;	Étape;	Step;	Manœuvre effectuée dans le cadre de progression vers l'état d'avancement d'un processus. Elle est référencée par une activité.
work request process;	Processus de demande d'intervention;	Work request process;	Type spécifique de processus lancé automatiquement ou par un acteur lors de la réception d'une demande de travail. Il permet de gérer la demande de travail jusqu'au résoudre le problème d'origine de l'événement déclencheur et la fin du processus d'intervention, y compris l'édition du rapport d'intervention.
Transition	Transitions ;	Transitions ;	Passage d'une étape à une autre dans le cadre d'un processus. Elle constitue une étape intermédiaire qui est envisagée comme un simple intermédiaire entre deux étapes dans un processus. C'est une connexion entre deux étapes. Une transition fait passer d'une étape à une autre représente la réponse au événement particulier. Une transition peut être un ensemble de valeurs ou événement déclencheur.
AcitivityInPutOutPut	Entrée Sortie d'une activité ;	Activity Input Output	Les paramètres, les valeurs et/ou les événements d'entrées ou déclencheurs de l'exécution d'une activité. Ces paramètres peuvent être les sorties de l'exécution d'autres activités. Le résultat d'exécution d'une activité (la sortie) est l'entrée d'une autre activité. Par ailleurs, une ActivityInPutOutPut est le lien de passage d'une activité à une autre. Par conséquent, chaque ActivityInPutOutPut peut faire référence à une Transition.

Un processus ***(Process)*** est une séquence d'activités ***(Acitivity)*** interdépendantes et liées faisant référence à des étapes ***(Step)***, qui, à chaque activité, consomme une ou plusieurs ressources pour convertir les entrées en sorties ***(ActivityInPutOutPut)***, tout en respectant un modèle de processus ***(Process Pattern)***. Ces sorties faisant

référence à des transitions ***(Transition),*** servent comme entrées qui elles aussi font référence à des transitions pour l'activité suivante jusqu'à ce qu'un but connu soit atteint (la clôture du processus).

Chaque *«Process»* fait référence à un *« Process Pattern »*, chaque *« Activity »* fait référence à un *« Step »* et chaque *« ActivityInPutOutPut »* fait référence à une *« Transition »*. Les instances des concepts processus et activité sont les interactions réelles faites (les traces enregistrées) dans la plateforme de s-maintenance et tracées par le processus de traçabilité. En outre, chaque activité est effectuée par un ou plusieurs acteurs (utilisateurs de la plateforme) qui sont définis par les instances du concept *« Actor »*. En conclusion, comme indiqué dans la Figure 5-7, le modèle de traces peut être résumé dans les concepts Process, Activity, Actor et ActivityInPutOutPut ainsi que les relations qui les structurent.

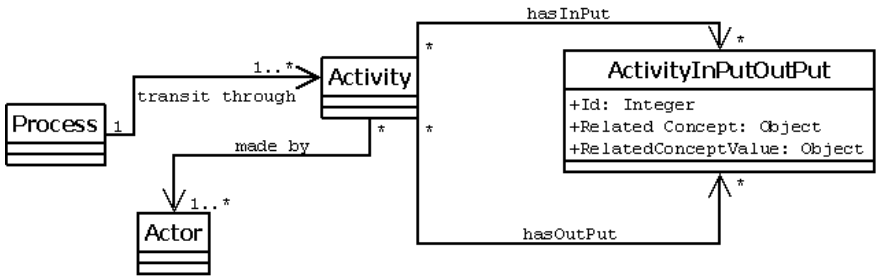


Figure 5-7 Modèle de Traces

3.2.2- Exemple illustratif

Pour mieux comprendre ce modèle, nous fournissons un exemple sur la maintenance conditionnelle. En effet, «Maintenance conditionnelle» est une instance du concept *« Maintenance Type »* qui est un sous-concept de *« Process Pattern »*. En outre, toutes les étapes et les transitions du processus de la «Maintenance conditionnelle» sont aussi instanciées.

Dans la Figure 5-8, nous présentons une illustration schématique de l'instanciation du concept *« Process Pattern »*. Nous faisons un focus sur le processus de gestion (présentation du «Process Pattern» de traitement de la demande de travail « work request ») qui fait référence à une étape (*Step*) dans le processus de maintenance conditionnelle (instance de Process Pattern).

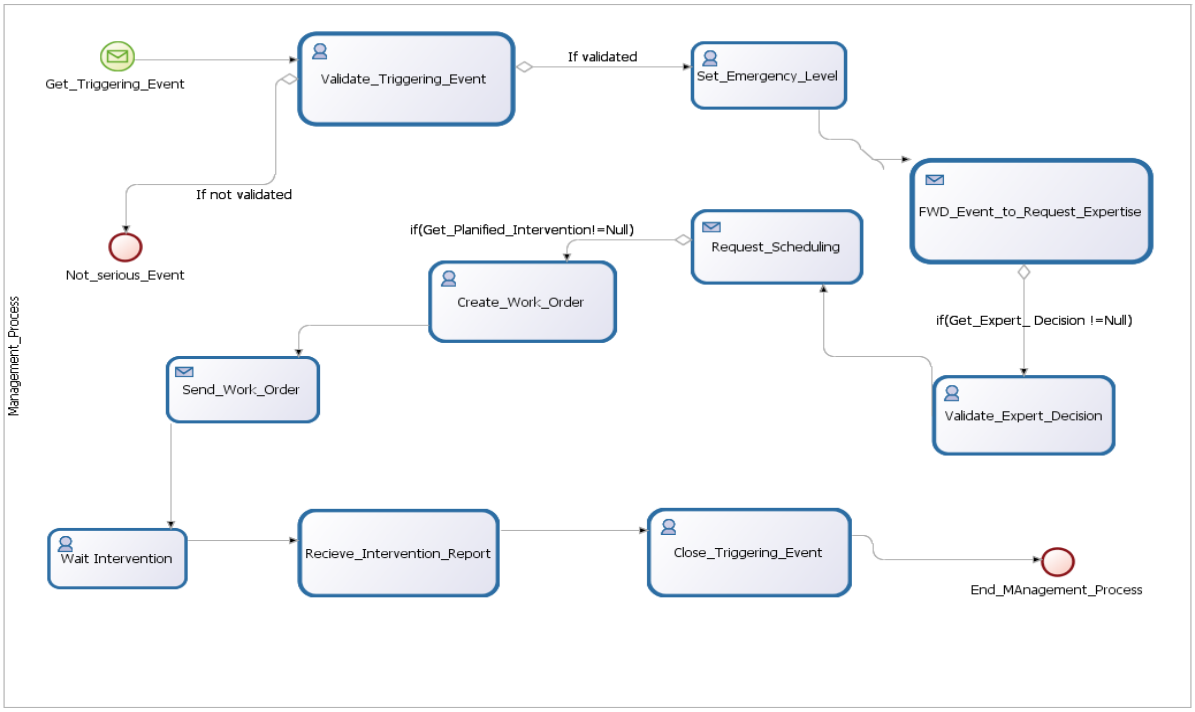


Figure 5-8 Exemple schématique d'une instance de *« Process Pattern »* (*management process*)

Les instructions PowerLoom suivantes présentent une partie de la base de connaissances concernant les processus et leurs instances.

```
(DEFCONCEPT Task)
(DEFCONCEPT Process-Pattern (?T Task))
(DEFCONCEPT Maintenance-Type (?PT Process-Pattern))
(DEFCONCEPT Step)
(DEFCONCEPT First-Step (?S Step))
(DEFCONCEPT Transition)
(DEFCONCEPT Next-Step (?S Step))
(DEFCONCEPT Constraint)

(Defrelation (StepInPut ((?T Transition) (?NS Step)))
(Defrelation (hasFirstStep ((?MT Maintenance-Type) (?FS First-Step)))
(Defrelation (StepOutPut ((?FS Step) (?T Transition)))
(Defrelation (Refrences ((?S Step) (?T Task)))

(Assert (Maintenace-Type Conditional-Maintenance)
(Assert (Maintenace-Type Predective-Maintenance)
(Assert (Maintenace-Type Systematic-Maintenance)
(Assert (Maintenace-Type Systematic-Maintenance)
(Assert (Step Monitoring-Process)
(Assert (Step Prognostic)
(Assert (Step Scheduling-Process)
(Assert (Management-Process)
```

```
(Assert (Transition Alarm)
(Assert (Transition RUL)
(Assert (Transition Notification-Date)

(ASSERT (hasFirstStep Conditional-Maintenance Monitoring-Process))
(ASSERT (hasFirstStep Predective-Maintenance Prognostic))
(ASSERT (hasFirstStep Systematic-Maintenance Scheduling-Process))

(ASSERT (StepOutPut Monitoring-Process Alarm))
(ASSERT (StepOutPut Prognostic RUL))
(ASSERT (StepOutPut Scheduling-Process Notification-Date))

(ASSERT (StepInPut RUL Management-Process))
(ASSERT (StepInPut Alarm Management-Process))
(ASSERT (StepInPut Notification-Date Management-Process))
```

Ainsi, nous rappelons que PowerLoom a un optimiseur de requêtes statiques et dynamiques qui est similaire aux optimiseurs utilisés dans les systèmes de bases de données. PowerLoom possède également une interface de base de données relationnelle qui lui permet d'utiliser la puissance de traitement des grandes bases d'instances (Chalupsky, MacGregor, & Russ, 2010). Ces avantages permettent la création de différentes vue de données, comme dans les bases de données grâce à des requêtes de jointure entre des concepts et des relations. Une vue est constituée d'une requête stockée accessible comme une table virtuelle composée de l'ensemble de résultats d'une requête (Bertino, 1992). Le tableau 5-2 présente un exemple d'une vue joignant chaque «Maintenance Type » avec sa première étape appelée « First Step » ainsi que la «Transition» entre la première et la deuxième étape («Next Step») dans IMAMO.

Tableau 5-2 Exemple du pattern "Maintenance Type"

Maintenance Type	**First Step**	**Transition**	**Next Step**
Conditional Maintenance	Monitoring process	Alarm	Management Process
Predictive maintenance	Prognostic	RUL	Management Process
Systematic maintenance	Scheduling	Notification (Date)	Management Process

Notre étude de cas se focalisera sur ce processus de gestion "Management Process" dans la maintenance conditionnelle pour illustrer le fonctionnement du système d'auto-apprentissage ainsi que la fonctionnalité d'autogestion dans la plateforme de s-maintenance.

Ainsi, comme hypothèse de travail, nous considérons que la plateforme de s-maintenance est utilisée pour gérer la maintenance des machines et des équipements industriels sur différents sites distribués un peu partout en France. Dans ces différents sites (parcs d'équipement), certains équipement font partie du même groupe d'équipement et partagent ainsi un même modèle d'équipement et de composants critiques de même capteurs à surveiller.

4. Système d'auto-apprentissage : Interprétation et Analyse des traces

Une fois les données de traces sont structurées et sauvegardées comme nouvelles instances dans la base de connaissance, l'étape d'apprentissage est lancée pour interpréter celles-ci et apprendre. Le processus d'auto-apprentissage est le processus de fonctionnement qui automatise le module d'apprentissage et d'interprétation dans l'étape 3. Selon Nilsson, l'apprentissage, comme l'intelligence, couvre un large éventail de processus qui est difficile à définir précisément. Nilson reprend sa définition de l'apprentissage les phrases du dictionnaire : « acquisition des connaissances, compréhension des compétences, instruction ou expérience par l'étude », et « la modification d'une tendance comportementale par l'expérience » (Nilsson, 1998). L'apprentissage automatique (machine learning en anglais) se réfère à un système capable d'acquérir et d'intégrer automatiquement des connaissances. La capacité des systèmes à apprendre de l'expérience, de la formation, de l'observation analytique, et d'autres moyens, résulte d'un système qui peut s'auto-améliorer en permanence et de ce fait, présente l'efficience et l'efficacité.

Un système d'apprentissage automatique commence habituellement avec une certaine connaissance organisée afin qu'il puisse interpréter, analyser et tester les connaissances acquises (worldofcomputing, 2011). Mettre en place des systèmes apprenants automatiquement sans intervention humaine, c'est mettre en place des systèmes capables d'auto-apprendre (Xu, Xu, Xiao, Zhou, Liu, & Jiang, 1995). Ainsi, tant que les techniques d'apprentissage automatique sont considérées comme le cœur de tout processus d'auto-apprentissage, un système d'auto-apprentissage est caractérisé par l'auto-ajustement de ses comportements en fonction des connaissances apprises (Lowe & Shirinzadeh, 2005).

Rappelons que la plateforme de S-maintenance est un système auto-apprenant (assure la fonctionnalité d'auto-apprentissage), avec la possibilité de faire évoluer son degré d'intelligence grâce à la dynamique de sa base de connaissances.

La fonctionnalité d'auto-traçabilité dans la plateforme se focalise sur l'ensemble de concepts suivant : *Process Pattern* (modèle de processus), *Process* (processus), *Step* (étape), *Activity* (activité), *Transition* et *Constraint* (contrainte) ainsi que les différentes relations entre ces concepts et leurs instances. Par conséquent, il sera possible pour le processus d'apprentissage de comprendre les différentes relations et interactions pour conclure certaines dépendances permettant de générer des règles pour des nouveaux modèles de processus.

Quoique, même après le prétraitement et la structuration dans le processus de traçabilité, les traces collectées sont généralement volumineuses et parfois non expressives. De ce fait, le processus d'apprentissage (d'interprétation) devient difficile pour l'analyste à cause de la complexité de la tâche. Pour cela, le module d'apprentissage associé à cette phase d'analyse, traite les traces en utilisant différentes méthodes d'analyse et d'apprentissage dont les méthodes statistiques et de fouille de données (data mining) qui restent les plus utilisées (Hilbert & Redmiles, 2000).

Dans cette étude, nous adoptons l'approche arbre de décision pour sa facilité à manipuler des données « symboliques[44] » et numériques. Nous rappelons succinctement qu'un arbre de décision affecte une classe (ou sortie) à un modèle d'entrée en filtrant le modèle à travers des tests dans un arbre de décision ce qui permet d'obtenir et que nous obtenons des règles mutuellement exclusives et exhaustives (Nilsson, 1998).

Nous avons adopté l'algorithme C4.5 (Quinlan, Induction of Decision Trees, 1990) et (Quinlan, C4.5: Programs for Machine Learning, 1993) issus de la plateforme Weka[45] pour créer un arbre de décision basé sur un ensemble de données étiquetées d'entrée. Le système C4.5 se compose de quatre programmes principaux: 1) le générateur d'arbres de décision ('C4.5') qui construit l'arbre de décision, 2) le générateur de règle de production ('c4.5rules') qui génère de règles de production à partir de l'arbre non élagué , 3) l'interpréteur d'arbres de décision qui classifie les éléments en utilisant un arbre de décision et 4) l'interpréteur de règles de production qui classifie les éléments en utilisant un ensemble de règles.

Les sorties du C4.5 constituent le bon arbre de décision, les tables d'erreurs de formation et des tests et la matrice de confusion. Dans l'architecture de la plateforme de s-maintenance que nous avons proposée, cet algorithme est inclu dans le composant *« Raisonneur »*.

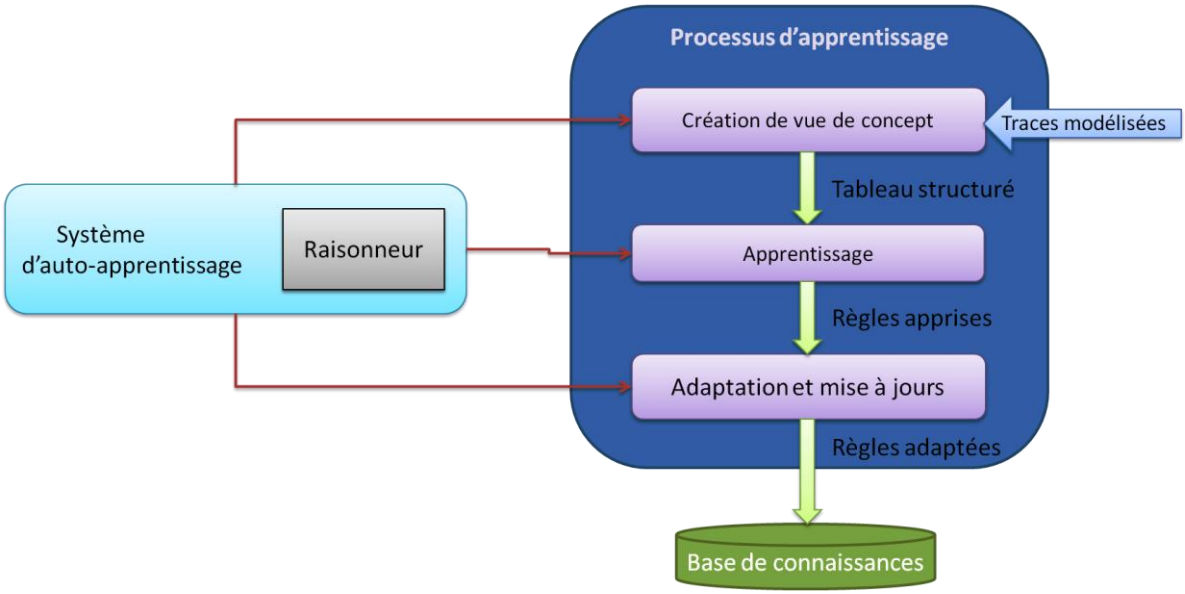


Figure 5-9 Activités du processus d'apprentissage

Tel que mentionné par Xu et al (Xu, Xu, Xiao, Zhou, Liu, & Jiang, 1995), un processus d'apprentissage doit satisfaire à la condition de cohérence qui signifie que le résultat d'apprentissage doit être compatible avec les connaissances d'origine déjà existantes dans la base de connaissances. Notre processus d'apprentissage est principalement basé sur les modèles de connaissances de IMAMO et les connaissances de la base de connaissances de la plateforme de s-maintenance. Comme indiqué dans la Figure 5-9, la première étape dans ce processus porte sur la création à partir de la base de connaissances de données contenant les connaissances

[44] *Les données sont symboliques* dans chaque case du tableau de données symboliques, on ne trouve pas nécessairement une seule valeur pour qu'elle soit quantitative ou qualitative.

[45] http://wekaclassalgos.sourceforge.net/

nécessaires à analyser. Cette étape est effectuée afin de présenter les connaissances nécessaires sous une forme structurée (tableau avec schéma spécifique). Puis, à partir de la deuxième vue, les différents fichiers nécessaires pour l'algorithme C4.5 sont créés de façon automatique dans la deuxième étape. Cette étape d'apprentissage consiste à exécuter l'algorithme C4.5 par le composant *« Raisonneur »*. Selon les pourcentages de la classification correcte ainsi que les pourcentages d'erreur relative, les résultats d'apprentissage sont validés ou pas. Finalement, s'ils sont validés, la troisième étape consiste à adapter les règles résultantes de cet apprentissage et par la suite à mettre à jour de la base de connaissances.

4.1- Création des vues

Comme présenté dans le modèle de processus dans IMAMO, chaque *«Process»* fait référence à un *« Process Pattern »* et chaque *« Activity »* fait référence à un *« Step »*. Les instances des concepts processus et activité sont les interactions réelles faites (les traces enregistrées) dans la plateforme de s-maintenance et tracées par le processus de traçabilité. En outre, chaque activité est effectuée par un ou plusieurs acteurs (utilisateurs de la plateforme) qui sont définis par les instances du concept *« Actor »*.

Les connaissances à la base de l'exercice d'apprentissage sont les instances des concepts activité et / ou processus. L'apprentissage, dans un soucis de génération a été appliquée aux concepts activité et/ou processus, et non pas à leurs instances.

L'intérêt de transformer et d'adapter les traces modélisées enregistrées dans la base de connaissances est de permettre au processus d'apprentissage analysant ces traces d'avoir des traces dans une forme adaptée à sa manière de fonctionnement.

Par conséquent, dans un premier temps, le composant *raisonneur* construit une vue contenant les instances, puis dans un second temps, il génère une seconde vue contenant des instances de concepts généraux, chaque instance d'activité ou de processus est remplacée par l'instance qui y fait référence sur les concepts *« Process Pattern »* (modèle de processus) ou *« Step »* (étape). Les tableaux 5-3 et 5-4 illustrent cette création de vues avec explication à l'appui. Il est à noter que les schémas de données des vues sont préalablement conçus dans le système en respectant le modèle ontologique de IMAMO. Les schémas proposés sont inspirés de la table de transition d'état qui est un tableau montrant dans quel état une machine à états finis se déplacera vers, en se basant sur l'état actuel et les inputs (Breen, 2005).

Le schéma de la première vue est présenté par un 5-uplet contenant l'activité concernée avec ses entrées/sorties, les activités sont les suivantes dans le processus de gestion d'un équipement physique :

< *ActivityInPut.Id, Activity, ActivityOutPut.Id, NextActivity, RelatedPhysicalEquipment* >.

Tableau 5-3 . Vue de premier niveau du processus d'apprentissage (vue physique)

ActivityInPut.Id	Activity	ActivityOutPut	NextActivity.Id	RelatedPhysicalEquipment
234	A982P22	536	A1236P22	P456
536	A1236P22	589	A2356P22	P456
789	A254P11	465	A269P11	Pu342
865	A11P18	965	A36P18	I231
965	A36P18	136	A39P18	I231
136	A39P18	245	A85P18	I231

Le tableau 5-3 ne fournit aucune information exploitable telle quelle. Nous allons donc raisonner au niveau des instances de concepts modèle et non au niveau des instances des concepts physique, grâce à l'ontologie IMAMO.

En ce qui concerne le deuxième niveau de vue, le schéma de données est principalement composé par le 8-uplet suivant :

< *ActivityInPut.Concept, ActivityInPut.Val, Step, ActivityOutPut.Concept, ActivityOutPut.Val, Constraint, NextStep, ConcernedEquipmentModel*>.

Cette deuxième vue est générée à partir de la première.
Step et *NextStep* dans cette deuxième vue correspondent respectivement aux *Activity* et *Next Activity* dans la première vue. *ActivityInPut.*
Concept et *ActivityInPut.Val* présentent les valeurs des propriétés (attributs) des instances du concept *ActivityInPut.*
La même chose est appliquée pour les instances du concept *ActivityOutPut.* En ce qui concerne *Constraint*, ceci est généré à partir de la *Transition* qui correspond au concept *ActivityInPutOutPut*.
ConcernedEquipmentModel présente le concept "*Equipment Model"* faisant référence au concept *"Physical Equipment"*.

Tableau 5-4 Vue de deuxième niveau du processus d'apprentissage

ActivityInPut.Concept	ActivityInPut.Val	Step	ActivityOutPut.Concept	ActivityOutPut.Val	Constraint	NextStep	Concerned Equipment Model
Alarm.Measure.Value	46	Validate Alarm	Alarm.validated	True	Alarm.Measure.Value>42	Set Emergency Level	Pusher
Alarm.validated	True	Set Emergency Level	Alarm.Emergency	Very High	Alarm.Measure.Value>42 And Alarm.Validate=True	Request exepertise	Pusher
Alarm	44	Validate	Alarm.validated	False	Alarm.Value>4	Close	Puller

		Alarm			2	Work request	
Measure	41	Control Data	Condition	Not Violated	Measure.Value> 42	Control Data	Indexer
Measure	42	Control Data	Condition	Not Violated	Measure.Value> 42	Control Data	Indexer
Measure	43	Control Data	Condition	Violated	Measure.Value> 42	Create Alarm	Indexer

4.2- Apprentissage

4.2.1- Construction de l'ensemble d'apprentissage

A partir de la deuxième vue, nous construisons l'ensemble d'apprentissage de C4.5. Cet ensemble est constitué par une collection de séquences de données. Chaque séquence a un *tuple* de valeurs pour un ensemble fixe d'attributs (ou variables indépendantes) A = {A1, A2,… An} et un attribut de classe (ou variable dépendante). Un attribut Aa est décrit comme continu ou discret selon ses valeurs qui sont numériques ou nominales (Kohavi & Quinlan, 2002).

Dans notre étude de cas nous avons pris comme variable à expliquer (variable de classe) l'attribut ActivityOutPut.Val et comme ensemble de variables indépendantes (variables explicatives) *{ActivityInPut.Concept, ActivityInPut.Value, ActivityOutPut.Concept, constraint, ConcernedEquipmentModel}.*

La Figure 5-10 montre un exemple de l'ensemble d'apprentissage soumis à C4.5 avec un validateAlarm.names définissant les attributs et un fichier validateAlarm.data recensant les instances.

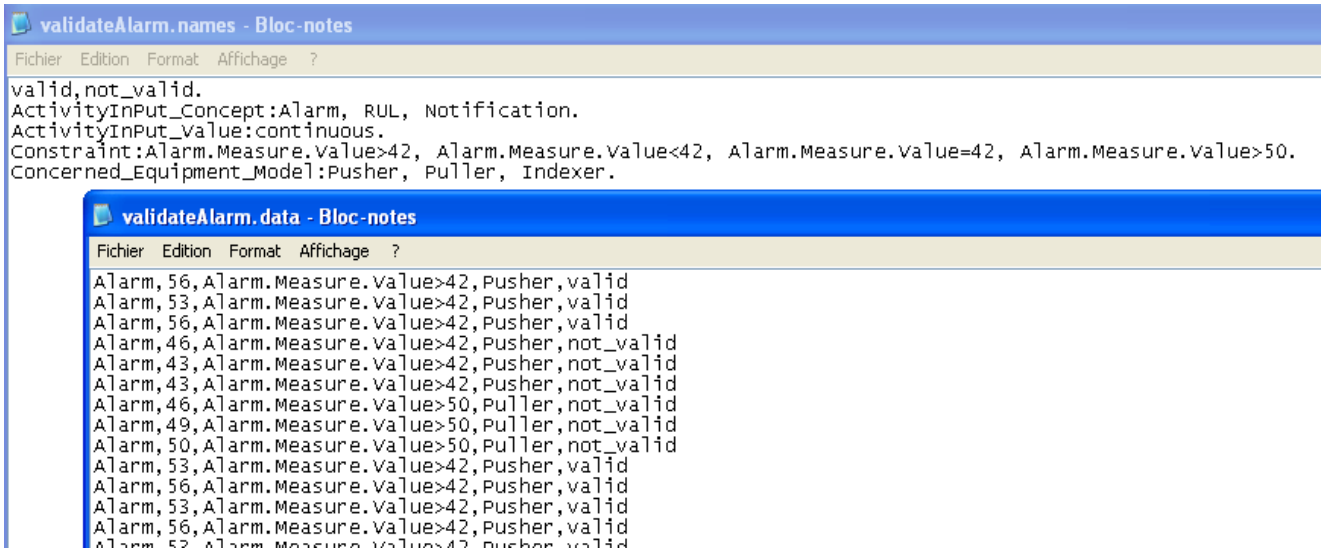


Figure 5-10 Exemple de fichiers C4.5 pour le STEP « validate.Alarm »

4.2.2- Lancement de C.4.5 et validation

Le lancement automatique de l'algorithme C4.5 peut se faire soit périodiquement (par exemple tous les mois), ou en fonction d'un seuil identifié sur le nombre d'instances dans la base de connaissances (par exemple l'algorithme sera lancé tous les 100 nouveaux instances d'un concept). Notre choix s'est porté sur la deuxième alternative car le nombre d'instances ne varie pas de façon uniforme dans le temps. En effet, nous pouvons avoir par exemple seulement 9 instances pendant un mois et 100 instances en deux jours. Ce nombre est vérifié par le composant raisonneur à chaque fois qu'une nouvelle instance est ajoutée à la base de connaissances. La Figure 5.11 montre un exemple d'arbre de décision obtenu suite à l'exécution de C4.5.

La validation des résultats d'apprentissage se fera suivant quatre indicateurs à savoir : les instances correctement classés (*Correctly Classified Instances), Kappa statistic*, l'erreur relative absolue (*Relative absolute error)* et l'erreur relative quadratique (*relative squared error*). Kappa *statistic* est une mesure de chance corrigée d'un accord entre les classes estimées et les classes vraies. Il est calculé en prenant l'accord attendu par chance loin de l'accord observé et en divisant par le maximum d'accord possible. Les taux d'erreur sont utilisés pour la prévision numérique plutôt que la classification. En prévision numérique, les prévisions ne sont pas seulement bonnes ou mauvaises, l'erreur a une grandeur, et ces mesures le reflètent (Bouckaert, 2010).

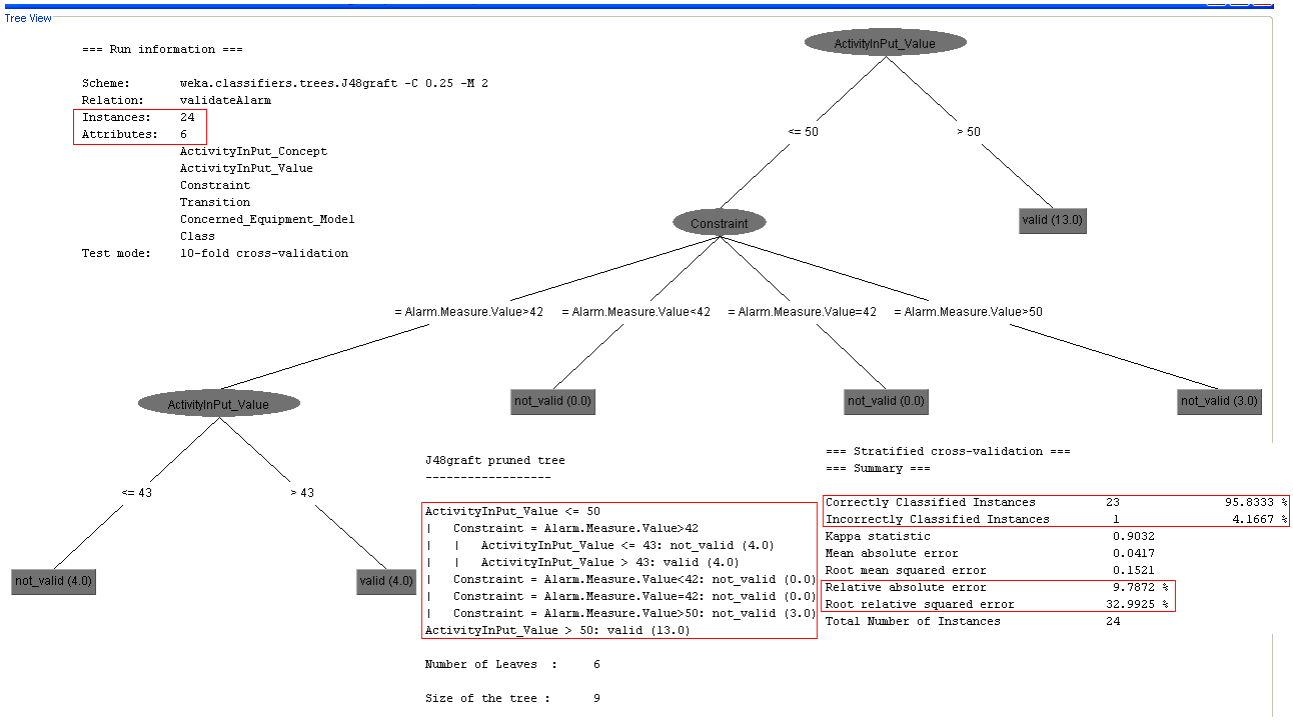


Figure 5-11 Résultats de la deuxième exécution de l'algorithme d'apprentissage

Les décisions se font automatiquement, nous avons adopté une politique de validation en définissant des seuils de confiance élevés comme configuration de base (Tableau 5), mais qui peuvent être modifiés par l'administrateur de la plateforme. En outre, nous notons que tous les seuils doivent être respectés, pour valider les résultats d'apprentissage.

Tableau 5-5 Seuils de validation des indicateurs

Indicator	Threshold
Correctly Classified Instances	> = 90%
Kappa statistic	> = 0.5
Relative absolute error	< = 10%
relative squared error	< 50%

4.3- L'adaptation des règles et mise à jour de la base de connaissances

L'adaptation, présente la troisième activité dans le processus d'apprentissage. L'objectif de cette activité est d'adapter les règles génèrées de l'activité d'apprentissage et de les enregistre comme de nouvelles instances dans la base de connaissances. Le fonctionnement de cette activité suit le processus présenté dans la Figure 5.12.

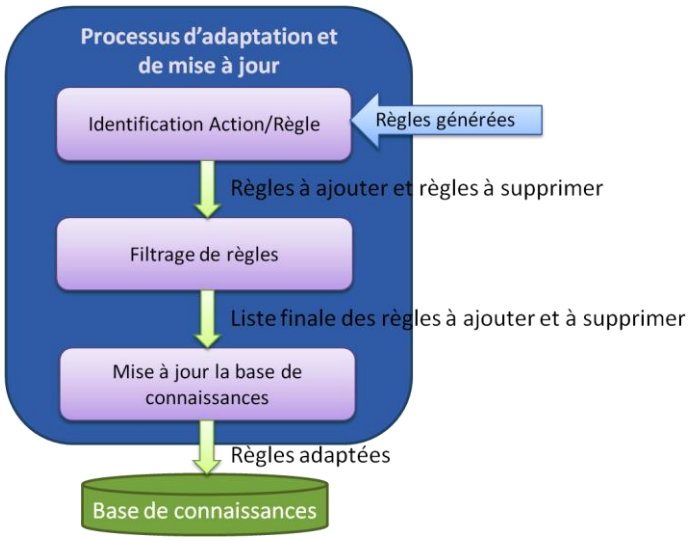


Figure 5-12 Processus d'adaptation et de mise à jour

La première activité dans ce processus porte sur l'identification des actions à faire par rapport à chaque règle générée (règles à ignorer, règles à supprimer, règles à ajouter). La deuxième activité porte sur le filtrage des règles à ajouter afin d'identifier les possibilités de subsomption. Après ceci, la dernière activité consiste à mettre à jour la base de connaissance par l'ajout des nouvelles règles, la suppression ou la modification des règles déjà existantes. En plus, nous rappelons que cette mise à jour concerne principalement les instances des concepts « Step», « Transition » et « Constraint ». La modification au niveau de cette vue de l'ontologie provoque un changement dans les modèles de processus (process pattern).

4.3.1- Identification actions par rapport aux règles générées

La Figure 5-13 fournit un exemple de résultat d'apprentissage des règles. Le *raisonneur* doit traduire ces règles sous une forme respectant la vue du processus de l'ontologie IMAMO. Le résultat d'apprentissage est composé par des règles définies et exprimées par les attributs des fichiers C4.5. Par conséquent,

le *raisonneur* prend avantage du fait que la structure de tous les fichiers de données est la même pour toutes les instances du concept *Step* et que les attributs présentent les concepts de l'ontologie. Ainsi, les règles qui en résultent sont exprimées en utilisant les concepts de l'ontologie et de leurs instances ou des termes connexes. L'exemple suivant fournit des règles sur les transitions possibles et leurs contraintes relatives, liées à «Valider alarme » une instance de concept *Step*.

```
J48graft pruned tree
------------------

ActivityInPut_Value <= 50
|   Constraint = Alarm.Measure.Value>42
|   |   ActivityInPut_Value <= 43: not_valid (16.0)
|   |   ActivityInPut_Value > 43: valid (4.0)
|   Constraint = Alarm.Measure.Value<42: not_valid (0.0)
|   Constraint = Alarm.Measure.Value=42: not_valid (0.0)
|   Constraint = Alarm.Measure.Value>50: not_valid (14.0)
ActivityInPut_Value > 50: valid (48.0)

Number of Leaves  :     6

Size of the tree :      9
```

Figure 5-13 Exemple de règles résultantes de l'apprentissage

Le résultat obtenu doit être analysé en utilisant la vue du second niveau liée à l'étape d'analyse dans l'activité d'apprentissage. Cette vue permet de comprendre le niveau de la relation entre les attributs utilisés dans les règles et les concepts. Par exemple, concernant la vue de second niveau, dans le cas du *Step «Valider alarme »*, *ActivityInPutValue* présente la valeur de *Alarm.Measure.Value*.

Afin d'interpréter les règles d'apprentissage générées, le *raisonneur* fait un *mapping* entre ces règles, la vue de deuxième niveau et la base de connaissance. Il construit un tableau d'interprétation ayant comme schéma *le 4-uplet (Step, contrainte classifié, Transition associée apprise, Contrainte existante).* Le tableau comportera donc, le *Step* sur lequel l'apprentissage était effectué, la contrainte classifiée associée à cet apprentissage, la valeur de la transition résultante de cette contrainte et finalement la contrainte déjà associée au *Step* dans la base de connaissance. La structure de ce tableau permettra de comparer les contraintes existantes et les contraintes classifiées par l'apprentissage, et permettra aussi la création dans la base de connaissances des relations entre ces nouvelles règles et les valeurs des transitions associées. Le tableau 6 présente le tableau d'interprétation du résultat d'apprentissage au sujet du *Step «Valider alarme »*.

Tableau 5-6 Table d'interprétation

Step	Extracted Constraint	Learned Transition	Existed constraint
Validate-Alarm	Alarm.Measure.Value >50	Alarm-Valid	Alarm.Measure.Value >50
Validate-Alarm	Alarm.Measure.Value <=43	Alarm-Not-Valid	Alarm.Measure.Value >42
Validate- Alarm	Alarm.Measure.Value >43	Alarm-Valid	Alarm.Measure.Value >42
Validate-Alarm	Null	Null	Alarm.Measure.Value <42
Validate-Alarm	Null	Null	Alarm.Measure.Value =42
Validate-Alarm	Alarm.Measure.Value <=50	Alarm-Not-Valid	Alarm.Measure.Value >50

A partir de ce tableau, grâce à la comparaison des règles apprises et des règles existantes, le *raisonneur* identifie les actions à faire par rapport à chaque règle apprise, soit une action d'ignorance ou de modification.

Par exemple dans les règles comparées dans le tableau 6, nous pouvons identifier les actions suivantes :

- *Ignorer les règles* contradictoires
    - La contrainte *Alarm.Measure.Value <=50* extraite lors de l'apprentissage est contradictoire à la contrainte existante Alarm.Measure.Value >50.
- *Modifier les règles existantes*:
    - Alarm.Measure.Value >42, Alarm.Measure.Value <42, Alarm.Measure.Value =42 seront remplacées par les règles Alarm.Measure.Value <=43 and Alarm.Measure.Value >43.

A partir de ces actions identifiées et les valeurs des transitions apprises, le raisonneur définit les règles et les relations à mettre en place dans la base de connaissances. A partir des actions identifiées pour notre cas d'étude (tableau 6), deux actions seront définit :

- Eliminer les relations avec les contraintes existantes à modifier (entre *Transition et Constraint*)
    - Alarm.Measure.Value >42
    - Alarm.Measure.Value <42
    - Alarm.Measure.Value =42
- Ajouter les nouvelles relations pour les règles apprises
    - Règle 1: Alarm.Measure.Value >43 (autogéré) (has Transition ← Valid_Alarm)
    - Règle 2: Si Alarm.Measure.Value <=43 (autogéré) (has Transition ← Not_Valid_Alarm)
    - Règle 3: Si Alarm.Measure.Value >50 (autogéré) (has Transition ← Valid_Alarm)

4.3.2- Filtrage des règles

Après ce *mapping* règles/actions, une activité de filtrage des règles à ajouter est lancée. Cette étape consiste à vérifier s'il y a des règles contradictoires ou s'il y a des règles plus générales que d'autres entre ces nouvelles règles (vérifier les possibilités de subsomption). Par exemple, la règle 1 est plus générique que la règle 3, d'où la règle la plus générale sera adoptée. Par conséquent, à la fin de cette activité deux règles sont validées à savoir règle 1 et la règle 2 ainsi que les règles qui seront modifiées.

4.3.3- Mise à jour de la base de connaissances

Comme défini dans l'ontologie, nous dissocions les modèles abstraits des modèles physiques. Nous avons un modèle abstrait relatif aux processus (représenté par les concepts S*tep, Transition et Constraint)* et un modèle opérationnel (représenté par les concepts*Activity, et ActivityInPutOutPut)* instanciés lors de l'exécution de la plateforme. La mise à jour de la base de connaissances concerne les instances du modèle abstrait de processus. Un exemple de mise à jour est illustré dans les Figures 5-14 et 5-15 fournissant un aperçu en amont et en aval de la base de connaissances ainsi que les opérations effectuées lors de sa mise à jour.

```
(DEFRELATION Confirmed-Self-Managed((?C Constraint) (?Self-Managed Boolean)))
(Assert (Step Validate-Alarm))
(Assert (Transition Alarm-Not-Valid))
(Assert (Transition Alarm-Valid))
(Assert (Constraint Alarm.Measure.Value>42))
(Assert (Constraint Alarm.Measure.Value<42))
(Assert (Constraint Alarm.Measure.Value=42))
(Assert (has-constraint Alarm-Not-Valid  Alarm.Measure.Value<42))
(Assert (has-constraint Alarm-Valid  Alarm.Measure.Value>42))
(Assert (has-constraint Alarm-Not-Valid  Alarm.Measure.Value=42))
(Assert (Confirmed -Self-Managed  Alarm.Measure.Value=42 False))
(Assert (Confirmed -Self-Managed  Alarm.Measure.Value<42 False))
(Assert (Confirmed -Self-Managed  Alarm.Measure.Value>42 False))
```

Figure 5-14 Base de connaissances initiale

Selon les règles apprises, le *raisonneur* commence par rétracter les instances à supprimer, puis ajouter les nouvelles instances. La base de connaissances mise à jour dans la Figure 5.19 montre les relations rétractées entre les transitions `Alarm-Valid` et `Alarm-Not-Valid` avec certaines contraintes, l'instanciation de nouvelles contraintes et de leurs relations avec les transitions `Alarm-Valid` et `Alarm-Not-Valid` et enfin l'affectation de la valeur « `True` » à l'attribut `Confirmed-Self-Managed` de ces contraintes. La Figure 5-16 présente la nouvelle base de connaissances résultante après la mise à jour.

Modifier régles existantes

```
(Retract (has-constraint Alarm-Valid  Alarm.Measure.Value>42))
(Retract (has-constraint Alarm-Not-Valid  Alarm.Measure.Value<42))
(Retract (has-constraint Alarm-Not-Valid  Alarm.Measure.Value=42))
(Assert (Constraint Alarm.Measure.Value>43))
(Assert (Constraint Alarm.Measure.Value<=43))
```

Ajouter nouvelles régles

```
(Assert (has-constraint Alarm-Valid  Alarm.Measure.Value>43))
(Assert (has-constraint Alarm-Not-Valid  Alarm.Measure.Value<=43))
(Assert (Confirmed-Self-Managed  Alarm.Measure.Value<=43 True))
(Assert (Confirmed-Self-Managed  Alarm.Measure.Value>43 True))
```

Figure 5-15 Base de connaissances mise à jour par le *GeP*

Base de connaissance après mise à jour :

```
(Assert (Step Validate-Alarm))
(Assert (Transition Alarm-Not-Valid))
(Assert (Transition Alarm-Valid))
(Assert (Transition Alarm.Measure))
(Assert (Constraint Alarm.Measure.Value>43))
(Assert (Constraint Alarm.Measure.Vale<=43))
(Assert (has-constraint Alarm-Valid  Alarm.Measure.Value>43))
(Assert (has-constraint Alarm-Not-Valid Alarm.Measure.Value<=43))
(Assert (Confirmed-Self-Managed Alarm.Measure.Value<=43 True))
(Assert (Confirmed -Self-Managed Alarm.Measure.Value>43 True))
```

Figure 5-16 Nouvelle base de connaissances

Ainsi, à l'issue de cette mise à jour de la base de connaissances, le modèle processus est modifié. La nouvelle version du processus de gestion dans la Figure 5-17 est bien évoluée par rapport au premier processus illustré dans la Figure 5-8.

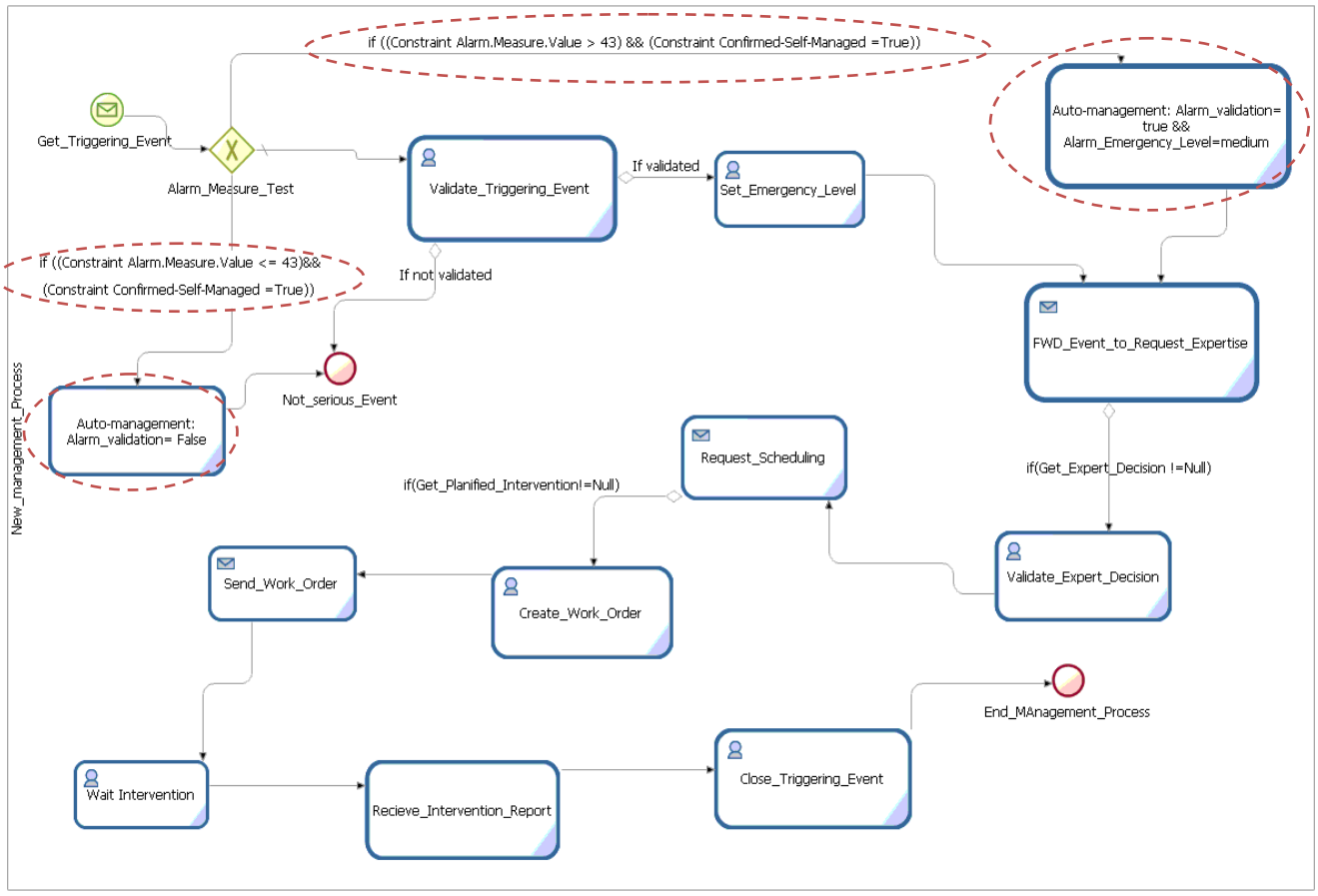


Figure 5-17La nouvelle version du processus de gestion

5. Exploitation des règles issues de l'apprentissage

Dans cette section nous présentons deux cas d'exploitation des règles issues de l'apprentissage. La première porte sur la fonctionnalité d'autogestion et la deuxième sur un service de diagnostic dynamique. L'exploitation de ces règles ajoutées dans la base de connaissances peut se faire de deux manières différentes, soit par autogestion des processus dans la plateforme, soit par les services fournis par les applications qu'elle intègre. La différence entre un service dynamique et la fonctionnalité d'autogestion est que cette dernière porte sur l'autogestion (exploitation des règles d'apprentissage) de l'activité effectuée par l'utilisateur humain et que le premier porte sur l'exploitation du service lui-même des règles d'apprentissage sur les activités effectuées par les applications.

5.1- Fonctionnalité d'autogestion

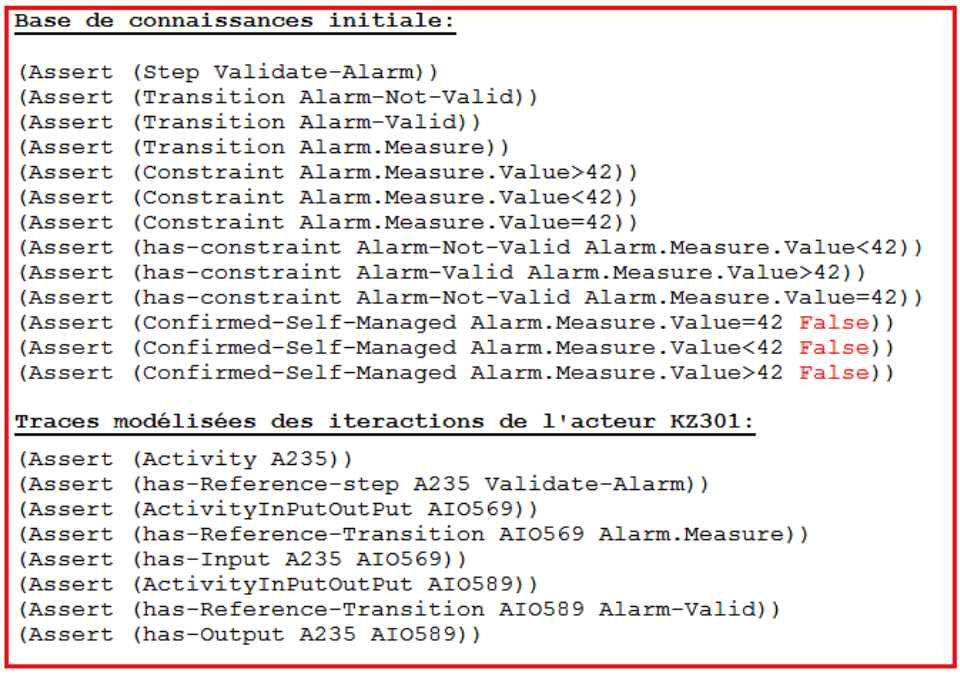
La fonctionnalité d'autogestion dans la plateforme de s-maintenance est assurée par le composant *GeP (gestionnaire de processus).* Elle permet d'exploiter les connaissances dynamiques apprises pour gérer de façon automatique les activités sur les quelles la fonctionnalité d'auto-apprentissage a été mise. L'exécution de ces activités en prenant appui sur ces règles de connaissances se fera sans intervention d'opérateurs.

En effet, la Figure 5-14 contient des éléments de la base de connaissances, *Confirmed-Self-Managed* est un attribut 46 booléen du concept «Constraint ». Sachant qu'une règle apprise porte sur une contrainte de transition entre activité, cet attribut est mis à «true» lors de la mise à jour de la base de connaissances.

La Figure 5-18 montre une vue de trace d'exécution extraite de la base de connaissance de la plateforme concernant quelques activités avant la mise à jour de la base de connaissance suite à l'auto-apprentissage sur ces activités. Dans cette phase, l'exécution et l'instanciation des ces activités sont assurées par des acteurs de maintenance utilisant la plateforme.

Après un auto-apprentissage validé sur une instance particulière du concept Step, les activités faisant références à ce Step (instances du concept Activity), ainsi que les entrées et sorties de ces activités (les instances du concept AcitivityInPutOutPut) seront autogérées (c.à.d. lancement automatiquement de l'exécution et affectation automatique des valeurs) par le composant *GeP*.

Dans la Figure 5-19, nous fournissons une vue extraite de la base de connaissances de la plateforme sur des activités et des transitions gérées automatiquement par le *GeP* avec une affectation automatique des valeurs de sorties de l'activité grâce aux règles apprises. L'exemple dans cette vue concerne une activité faisant référence au *Step « Validate-Alarm »* dans le processus de management illustré dans la Figure 5-17.



```
Base de connaissances initiale:

(Assert (Step Validate-Alarm))
(Assert (Transition Alarm-Not-Valid))
(Assert (Transition Alarm-Valid))
(Assert (Transition Alarm.Measure))
(Assert (Constraint Alarm.Measure.Value>42))
(Assert (Constraint Alarm.Measure.Value<42))
(Assert (Constraint Alarm.Measure.Value=42))
(Assert (has-constraint Alarm-Not-Valid Alarm.Measure.Value<42))
(Assert (has-constraint Alarm-Valid Alarm.Measure.Value>42))
(Assert (has-constraint Alarm-Not-Valid Alarm.Measure.Value=42))
(Assert (Confirmed-Self-Managed Alarm.Measure.Value=42 False))
(Assert (Confirmed-Self-Managed Alarm.Measure.Value<42 False))
(Assert (Confirmed-Self-Managed Alarm.Measure.Value>42 False))

Traces modélisées des iteractions de l'acteur KZ301:

(Assert (Activity A235))
(Assert (has-Reference-step A235 Validate-Alarm))
(Assert (ActivityInPutOutPut AIO569))
(Assert (has-Reference-Transition AIO569 Alarm.Measure))
(Assert (has-Input A235 AIO569))
(Assert (ActivityInPutOutPut AIO589))
(Assert (has-Reference-Transition AIO589 Alarm-Valid))
(Assert (has-Output A235 AIO589))
```

Figure 5-18 Une vue de la base de connaissances avant l'auto-apprentissage

[46] Nous rappelons que dans PowerLoom, les attributs des concepts sont présentés comme des relations entre le concept et le nom de l'attribut et son type.

```
Base de connaissance après mise à jour :

(Assert (Step Validate-Alarm))
(Assert (Transition Alarm-Not-Valid))
(Assert (Transition Alarm-Valid))
(Assert (Transition Alarm.Measure))
(Assert (Constraint Alarm.Measure.Value>43))
(Assert (Constraint Alarm.Measure.Vale<=43))
(Assert (has-constraint Alarm-Valid  Alarm.Measure.Value>43))
(Assert (has-constraint Alarm-Not-Valid Alarm.Measure.Value<=43))
(Assert (Confirmed-Self-Managed Alarm.Measure.Value<=43 True))
(Assert (Confirmed -Self-Managed Alarm.Measure.Value>43 True))

Traces modélisées des interactions du GEP :
(Assert (Activity A425))
(Assert (has-Reference-step A425 Validate-Alarm))
(Assert (ActivityInPutOutPut AIO692))
(Assert (has-Reference-Transition AIO692 Alarm.Measure))
(Assert (has-Input A425 AIO692))
(Assert (ActivityInPutOutPut AIO891))
(Assert (has-Reference-Transition AI891 Alarm-Valid))
(Assert (has-Output A425 AIO891))
```

Figure 5-19 Vue de la base de connaissances après auto-apprentissage et avec autogestion

5.2- Service dynamique

Nous illustrons sur les Figures suivantes le cas du service de diagnostic qui évolue dynamiquement avec la dynamicité de la base de connaissance à chaque fois une nouvelle règle apprise concerne l'une des activités gérées par lui est ajoutée dans la base. La Figure 5-20 illustre le processus de diagnostic initial définit dans la base de connaissance.

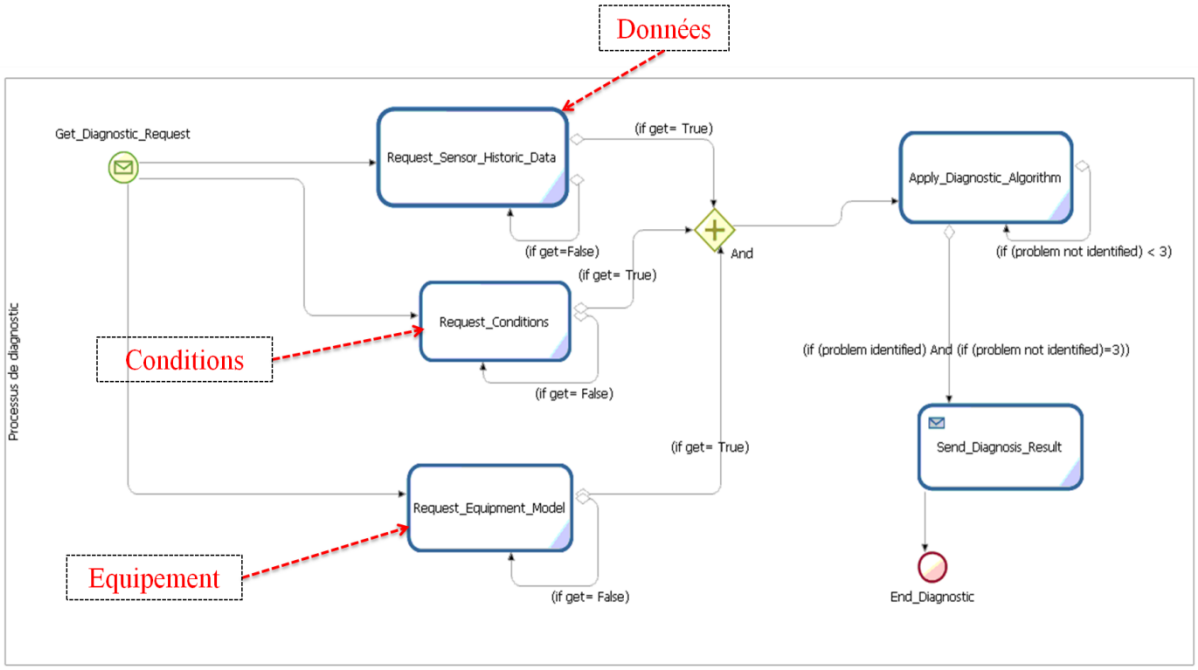


Figure 5-20 Processus initial du service de diagnostic

Ainsi, après un auto-apprentissage sur l'étape (activité modèle) *« Apply_Diagnostic_Algorithm »*, une nouvelle règle sur les entrées (Données, Condition, Modèle équipement) et la sortie (problème identifié) est rajoutée dans la base de connaissances.

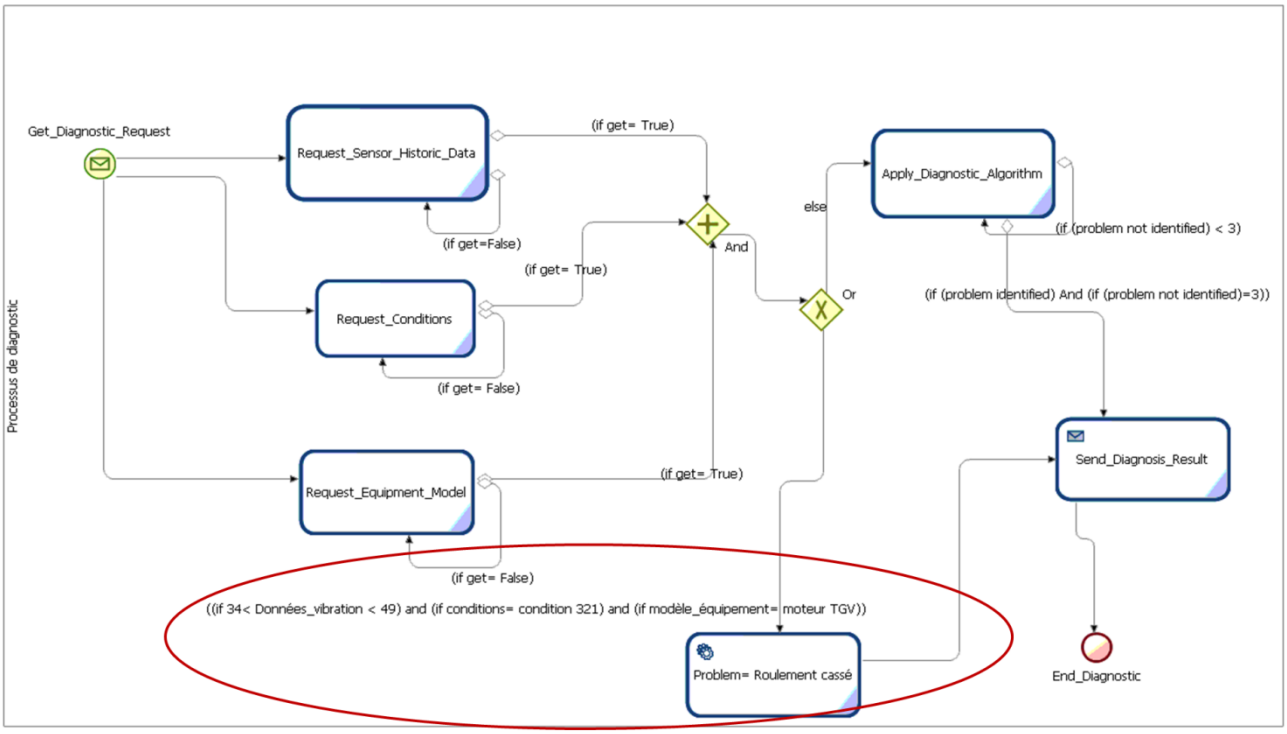


Figure 5-21 Nouveau processus du service de diagnostic après un premier auto-apprentissage

Par exemple, l'apprentissage sur les entrées sorties du *step « Apply_Diagnostic_Algorithm »* a permis de déduire que pour les moteurs de TGV (modèle d'équipement) , quand les données du capteur de vibration sont entre 34 et 49 et que la condition associée est la condition 321 donc le problème identifié est « roulement cassé ». Cette nouvelle règle est ajoutée donc à la base de connaissances. Lors des nouvelles exécutions du service de diagnostic, ce dernier suit le nouveau processus issu de cet apprentissage (voir Figure 5-21).

Ainsi, à chaque fois que de nouvelles règles sont apprises, le service de diagnostic suit le nouveau processus résultant de ce dernier apprentissage. La dyanmicité du service est toujours en évolution avec l'avancement de l'apprentissage et de l'exécution.

La Figure 5.22 fournit une autre version du processus du service de diagnostic suite à une nouvelle règle d'apprentissage portant sur les actionneurs. En effet, pour les équipements ayant comme modèle d'équipement actionneur, si les données du capteur de fin de course sont égales à faux et si la condition associée est la condition 2574 alors le problème identifié est « vieillissement des joints des vannes ».

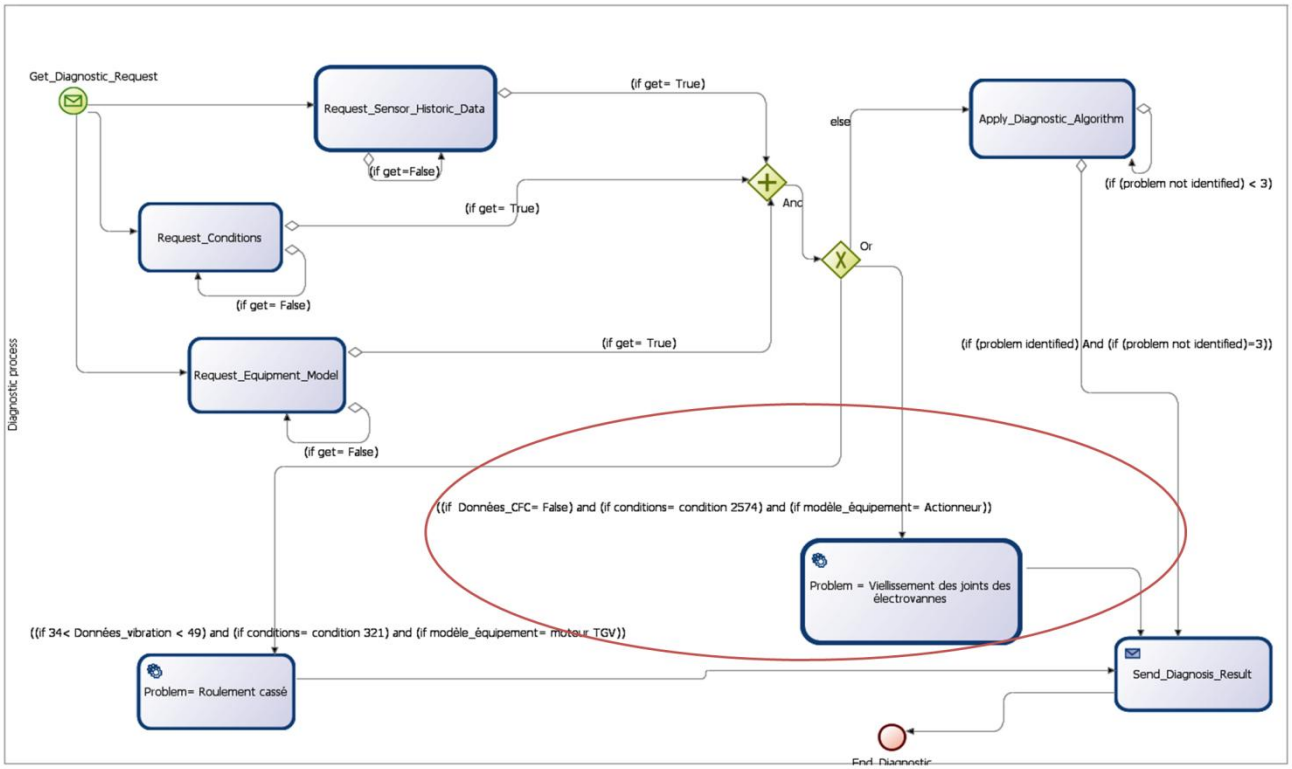


Figure 5-22 Nouveau processus du service de diagnostic issu d'un nouvel auto-apprentissage

Après avoir appliqué ces fonctionnalités sur un cas d'étude, et montrer la faisabilité de ce système à base de traces, nous étudierons l'impact de celles-ci sur les performances de la fonction de maintenance.

6. Impact de ces fonctionnalités sur les performances de la maintenance

El Aoufir et al, définissent l'efficience de la maintenance par son aptitude à améliorer les objectifs fixés (disponibilité, sécurité, qualité, …) tout en réduisant le coût global de la maintenance. Ils affirment que la mesure de la performance du processus maintenance est liée à la définition et à la mesure d'indicateurs de performance techniques (exemple : la disponibilité) et d'indicateurs économiques (les coûts directs et indirects de la maintenance) (EL AOUFIR & BOUAMI, 2005).

L'objectif principal des systèmes de maintenance est de faire évoluer cette efficience en améliorant la mesure de performance du processus de maintenance.

Dans ce contexte, nous allons faire une étude sur l'impact des fonctionnalités de la plateforme de s-maintenance sur les performances techniques et économiques de la maintenance. Cette étude portera sur deux types d'indicateurs de temps et de coûts.

6.1- Performance technique : indicateurs de temps

Nous nous intéresserons à l'indicateur de temps global du processus MTTR permettant de calculer le temps global du processus de maintenance (Stanley, 2011). Il existe d'autres indicateurs, comme le MTBF[47] facilement calculable dans la plateforme de maintenance, mais que nous n'utiliserons pas dans notre cas de simulation de fonctionnement de la plateforme.

Le MTTR *(mean time to repair)* est appelé aussi indice de maintenabilité. La maintenabilité s'entend, pour une entité utilisée dans des conditions données, comme la probabilité pour qu'une activité donnée de maintenance puisse être effectuée sur un intervalle de temps donné, lorsque la maintenance est assurée dans des conditions données et avec l'utilisation de procédures et moyens prescrits (Stanley, 2011).

Le MTTR est calculé en additionnant les temps actifs de maintenance (comportant les temps de localisation de défaillance, de diagnostic, d'intervention, de contrôles et d'essais) ainsi que les temps annexes de maintenance (comportent les temps de détection, d'appels à la maintenance, d'arrivée de la maintenance, de préparation de la logistique d'intervention, etc.), le tout divisé par le nombre d'interventions.

De plus, il est à noter que les indicateurs de maintenance se basent généralement sur les données réelles et non pas sur des estimations. La plateforme permet d'effectuer des calculs réels ainsi que des calculs d'indicateurs estimés (par exemple grâce à l'attribut « Estimated-Execution-Duration » du concept « Step ») ce qui permettra de comparer ces différentes valeurs entre elles.

Par conséquent, le temps estimé d'exécution d'un processus est égal au temps d'exécution estimé d'un « process pattern » que nous notons TexPp (Temps exécution Process pattern).

$$TexPp(pp) = \sum (pp.Act_i.estimated\_excution\_duration))$$

D'autre part, l'exécution d'un processus de maintenance dure un certain temps, avec un début (period.start) et une fin (period.end). Un processus peut être démarré par un utilisateur ou bien par un autre processus. Ce temps d'exécution représente la somme des temps d'exécutions réels des activités (calculer grâce au concept « period » et ses attributs (start et end)) appartenant à chaque processus. On le note TexP (Temps exécution processus) :

$$TexP(p) = (p.Period.End - p.Period.Start)$$
$$= \sum (TexA(\ p.Act_i\ )) = \sum (p.Act_i.Period.End - p.Act_i.Period.Start)$$

Ainsi, pour calculer les temps estimés et réels de maintenance sur une période donnée, il suffit de multiplier les deux indicateurs précédents par le nombre de processus instanciés tout au long de cette période.

Par conséquent, le temps d'exécution estimé (Tee) pour une période A donnée se calcule par :

$$Tee\ (A) = K * TexPp(MT)$$

[47] Le MTBF (Mean Time Between Failures), appelé aussi indice de fiabilité, désigne le temps moyen entre défaillances consécutives d'un type d'équipement.

K présente le nombre de processus instanciés et faisant référence aux instances du concept« Maintenance Type » (MT) qui est un sous concept de « process pattern »[48].

Ainsi le Temps d'exécution réel (Ter) est noté par:

$$Ter\ (A) = H * TexP(p)$$

H est le nombre de processus instanciés du processus (p) instanciés et faisant référence aux instances du concept« Maintenance Type » durant la période l'année A.

Les deux temps calculés précédemment permettent d'évaluer l'erreur entre le temps estimé et le temps réel pour une période donnée. En fonction de ce taux d'erreur, nous pouvons remettre en cause le calcul d'estimation de temps. Cet indicateur que nous appelons erreur d'estimation de temps d'exécution (EeTe) est noté par:

$$EeTe = Ter - Tee$$

6.2- Performance économique : indicateurs de coût

En ce qui concerne les performances économiques, elles sont liées directement aux coûts de la maintenance. La norme NF X60-0201 définit les coûts de maintenance comme étant *les coûts directement imputables à la maintenance*. Elle précise que *ces coûts peuvent s'analyser par nature (personnel, outillages et équipements de maintenance, produits et matières consommées, sous-traitance, autres) et par destination (préparation, documents techniques, interventions, suivi et gestion, magasinage, formation, autres, ...).* Ces coûts sont généralement appelés coûts directs.

La Figure 5-23 montre qu'il existe des coûts indirects en opposition avec les coûts directs et qui se répercutent sur les coûts cumulés de maintenance. Ces coûts indirects sont composés de :

- coûts d'indisponibilité qui incluent les coûts de pertes de production, la non qualité, les surcoûts de production, les pénalités contractuelles, … consécutifs à une défaillance.
- coûts de défaillance qui intègrent les coûts de maintenance corrective et les coûts d'indisponibilité consécutifs à la défaillance.

En effet, les coûts directs d'un processus dépendent des coûts des activités appartenant à celui-ci et les coûts indirects dépendent directement de son temps d'exécution.

[48] Remarque : Les processus complexes sont composés par des activités qui peuvent faire références à des « steps » qui font références à d'autres processus (exemple un processus de maintenance conditionnelle fait appel à un processus de diagnostique et un processus d'intervention. Par conséquent, pour éviter toutes erreurs de calcule, nous considérons pour tous les calcules dans cette section que les processus faisant références à des « process pattern » de type « maintenance type ».

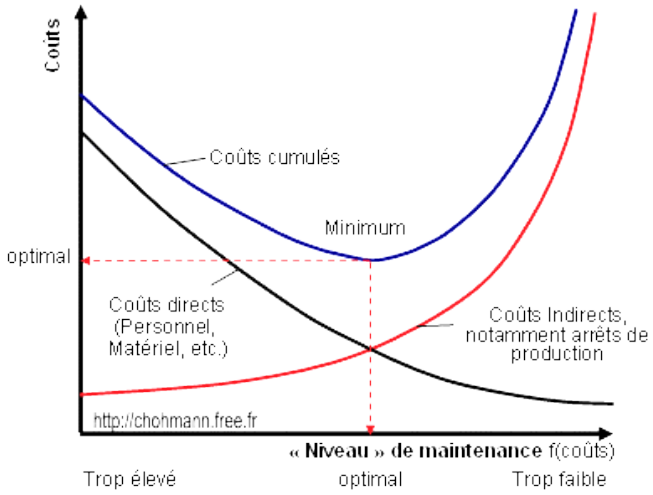


Figure 5-23 Coûts de la maintenance[49]

Lorsque la plateforme a appris une règle permettant de résoudre automatiquement un problème aucun coût ne sera occasionné pour cette activité, au contraire d'une activité impliquant un utilisateur. Nous partons de l'hypothèse suivante pour estimer les coûts directs et indirects d'une activité. Nous classons donc ces activités en deux catégories :

- Activité à coût : une activité ayant une sortie « AcitivityInputOutPut » faisant référence à une transition ayant « confirmed-self-managment » égale à « false ». Le coût cumulé estimé de cette activité est calculé grâce aux valeurs des attributs « estimated-execution-duration » et « estimated-coast » du « step » sur lequel l'activité fait référence.
- Activité sans coût : une activité ayant une sortie « AcitivityInputOutPut » faisant référence à une transition ayant « confirmed-self-managment » égale à « true ». le cout cumulé de cette activité sera égal à zéro grâce à l'automatisation de cette activité (vu la non implication d'utilisateur dans son exécution et le temps négligé de cette exécution).

De la même façon des temps d'exécution, nous allons calculer les coûts réels ainsi que les coûts fictifs estimés des processus de maintenance pour les futures prévisions budgétaires.

Par conséquent, le coût cumulé estimé d'un processus de maintenance est :

$$Ce(p) = \text{cout directe } (p) + \text{cout indirect } (p) = \sum (Cact(\ p.Act_i\ )) + \ (k * (TexP(p)))$$

Sachant que Cact ($p.Act_i$) prèsente le coût d'une activité *i* dans un processus *p*. De plus, la constante k représente le coût d'arrêt de production par unité de temps. En d'autres termes, coût de la perte à cause de l'arrêt de la production.

Ainsi, le coût annuel estimé est noté par :

$$CAe(A) = NI(p) * Ce(p)$$

[49] http://chohmann.free.fr/

NI(p) présente le nombre d'instances du processus « p » durant l'année A.

Le cout réel approximatif des processus de maintenance (que les processus faisant référence à des « process pattern » de type « maintenance type ») durant une année CMr (Coût Manitenance Réel) est égal à :

$$CMR(A)= \sum NI_j * Ce(p_j)$$

Dans cette équation, NI_j présente le nombre d'instances du processus p_j durant l'année A.

Ainsi, le gain annuel de maintenance que l'entreprise peut obtenir présente la différence entre le coût estimé et le coût réel durant une année. Ce gain est présenté par l'équation suivante :

$$Gain\ (A)= Ce(A) - CMR(A)$$

Cet indicateur est utile pour évaluer les performances économiques de la maintenance ainsi que pour ajuster les futurs de m dudgets aintenance.

6.3- Simulation par étude de cas

Dans cette section, nous présentons une étude de cas de l'entreprise fictive TEM (Technlogies, Equipmente and Maintenance), relative à la gestion de la maintenance via la plateforme de s-maintenance. Nous évaluons l'impact de cette gestion sur les performances économiques et techniques de cette entreprise.

TEM est spécialisée dans la fabrication de machines destinées au décolletage de pièces à partir de barres ou de lopins. L'entreprise propose aujourd'hui une large palette de produits. Les gammes principales se composent de tours automatiques mono-broches à poupée mobile, des tours multibroches et de tours multibroches à cames. En effet, en complément de ses produits industriels, TEM fournit aussi à ses clients des services de maintenance avec les machines qu'elle vend. Pour assurer ses services de maintenance, TEM utilise la plateforme de s-maintenance Tem@web.

Le nombre de machine géré par la plateforme Tem@web s'élève à plus de 2000 machines dispersées un peu partout dans le monde. Nous étudierons un échantillon de 1000 machines de la même gamme produites par cette entreprise.

Dans ce contexte, nous posons un ensemble d'hypothèses permettant de simuler sur une période donnée, des données reflétant l'activité de la plateforme de s-maintenance Tem@web dans l'entreprise TEM (voir tableau 5-7).

Hypothèses sur le nombre de processus instanciés :

Le nombre de processus instanciés dans la plateforme dépend du nombre de défaillances enregistrées de ces machines qui est en lien direct avec les événements déclencheurs.

Afin de mettre en place des hypothèses sur les nombre de défaillances, nous avons utilisé la courbe en baignoire pour simuler un nombre de défaillances sur chaque période du cycle de fonctionnement de chaque équipement à savoir la jeunesse, la maturité et la vieillesse (voir Figure 5-24).

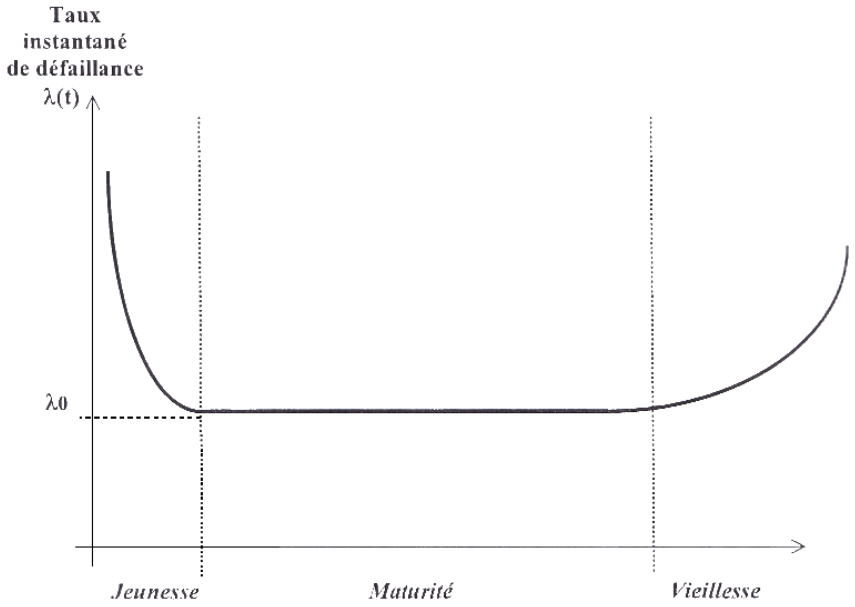


Figure 5-24 Courbe en baignoire

Selon la décomposition de la courbe en baignoire, sur les 1000 machines gérées par la plateforme Tem@web pour une période de 10 ans, nous prenons comme hypothèses : le nombre d'événements déclencheurs par mois s'élève à

- 20% du nombre de machine sur la phase de jeunesse qui dure une année, soit de 2400 par ans.

- 10% du nombre de machine sur la phase de maturité qui dure cinq années, soit de 1200 par ans.

- 35% du nombre de machine sur la phase de vieillesse qui dure quatre années, soit de 4200 par ans.

Chaque événement déclencheur induit l'instanciation d'un processus. Le nombre de processus de maintenance instanciés par an est égal au nombre d'événements instanciés.

Hypothèses sur les processus et leurs coûts

- Il faut mentionner que les processus instanciés ne sont pas tous du même type (ne font pas référence au même « process pattern» et plus précisément au même « maintenance type »). Les coûts de chaque pattern doivent être calculés indépendamment les uns des autres.

- Pour simplifier la simulation, nous allons considérer un coût moyen direct d'un processus (CMdP). Ce coût est calculé par la sommation des coûts estimés de toutes les instances du concept « maintenance type » divisé par le nombre d'instance de ce concept.

$$CMdP = \sum Ce(\ MaintenanceType_i\ ) / Maintenance\_Type.Nombre\_Instances$$

Hypothèses sur les activités :

- Le coût d'un processus dépend des coûts des activités composant ce processus. Afin de simplifier les calculs, nous faisons le calcule du coût moyen des activités dans un processus (CMA).

$$CMA\ (p) = (\sum (Coût(\ p.Actvité_i\ ))) / p.nombre\_Activité$$

- Soulignons que le nombre d'activités dans un processus varie d'un processus à un autre et dépend principalement du nombre de « step » dans un « process pattern ». Pour cela nous considérons une moyenne du nombre d'activités par processus. Cette moyenne consiste à diviser la somme des « step » appartenant à chaque instance de « maintenance type » par le nombre de ces instances.

$$NAP= \sum (nombre\_step.MT_i) / Maintenance\_Type.Nombre\_Instances$$

Hypothèses sur l'apprentissage :

- En ce qui concerne l'apprentissage, nous soulignons que toutes les activités ne peuvent pas être considérées par les services d'auto-apprentissage et d'autogestion. Il existe des activités qui nécessitent des intervenants humains pour des opérations manuelles (par exemple remplacement d'un composant). A cet effet, les activités prises en compte par les deux services sont généralement les activités de prise de décision qui nécessitent des compétences et connaissances de raisonnement.

- D'autre part, nous émettons l'hypothèse que 60% des activités d'un processus est susceptible d'être automatisé.

- De même l'estimation du taux d'apprentissage sur les activités pendant la période de gestion de 10 ans des 1000 machines. Cette période sera décrite à partir de la courbe en baignoire. Le taux d'apprentissage varie d'une phase à l'autre. La Figure 5-25 présente le taux d'apprentissage sur une période de 10 ans, qui varie suivant le nombre de processus instanciés le long de chaque phase. Ainsi, nous pouvons le constater sur la courbe présentée sur la Figure 5-25, l'existence de périodes où l'apprentissage est égal à 0. Ce qui signifie que le processus d'auto-apprentissage pendant cette période n'a inféré aucune règle. Ceci peut être considéré comme une situation qui n'a rien d'inquiétant. En fait, cela se produit quand le système se stabilise et atteint un niveau de maturité suite aux apprentissages effectués pendant les périodes précédentes.

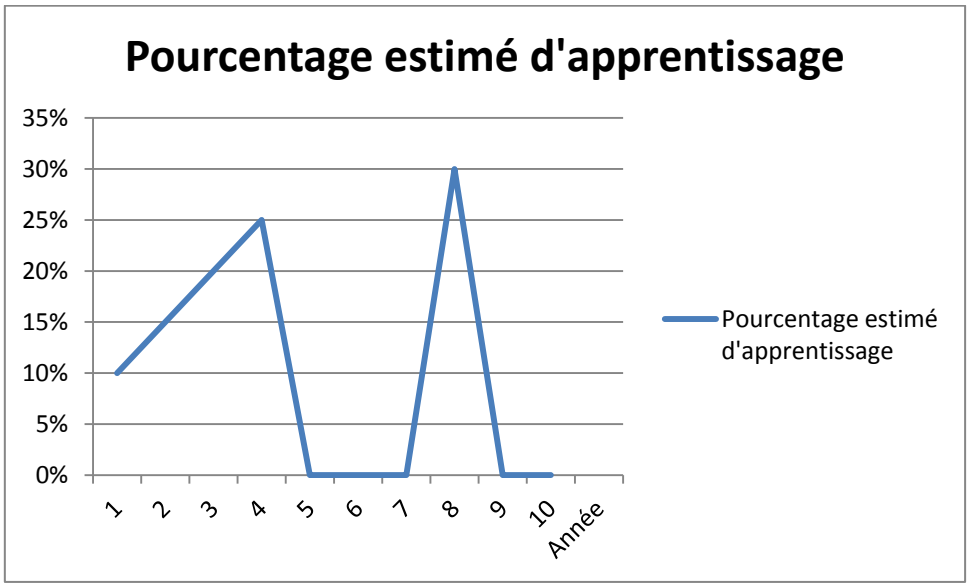


Figure 5-25 Hypothèses d'apprentissage sur une période de 10 ans

Le taux d'apprentissage d'une année se répercutera sur l'année suivante, ce qui explique que le temps réel d'excution lors de la première anné est identique au temps d'exécution estimé.

Simulation des calculs

Avant d'effectuer les calculs de simulation résumés dans le tableau 5-7, nous présentons un ensemble d'hypothèses de départ sur les coûts à utiliser :

- Nombre moyen d'activités dans un processus (NAP) = 10 activités
- Coût moyen d'une activité CMA (p) = 10 Unités de Coût (UC)
- Temps d'exécution estimé d'une activité = 5 Unités de Temps (UT)
- Coût moyen d'un processus = 100 Unités de Coût (UC)
- Temps moyen estimé d'un processus = 50 Unités de Temps (UT)
- Le coût indirect d'un processus = coût d'arrêt de production = (k * (TexP(p), dans notre cas k=1 Unité de Coût).

Il est à noter, que les coûts réels sont calculés à travers les fonctionnalités d'auto-X de la plateforme qui tiennent compte de la présence des règles de décision extraites et enregistrés dans la base de connaissances. Ces règles ont pour objectif la simplification des fontionnalités d'auto-X.

Les coûts estimés ne tiennent pas compte de ces fonctionnalités mais ils sont estimés par rapport à toutes les activités des processus préalablement définis.

Tableau 5-7 Données des performances de maintenance simulées pour TEM sur 10 ans

		1ère année	2ème année	3ème année	4ème année	5ème année	6ème année	7ème année	8ème année	9ème année	10ème année
1	Nombre de processus instanciés	2400	1200	1200	1200	1200	1200	4200	4200	4200	4200
2	Taux d'apprentissage	10%	15%	20%	25%	0	0	0	30%	0	0
3	Nombre moyen d'activité par processus	10	9.4	8.59	7.7	6.8	6.8	6.8	6.8	6	6
4	Temps d'exécution estimé (Tee) (en UT)	120000	60000	60000	60000	60000	60000	210000	210000	210000	210000
5	Temps d'exécution réel (en UT)	120000	56400	51540	46200	40800	40800	142800	142800	126000	126000
6	Gain en temps d'exécution (en UT)	0	3600	8460	13800	19200	19200	67200	67200	84000	84000
7	Coût estimé d'un processus (en UC)	100	100	100	100	100	100	100	100	100	100
8	Coût estimé des processus sans auto-X (en K UC)	360 K	180 K	180 K	180 K	180 K	180 K	630 K	630 K	630 K	630 K
9	cout réel approximatif avec auto-X (en K UC)	360 K	169K	155K	139K	122 K	122 K	429 K	429 K	378 K	378 K
10	Gain/année (en UC)	0	11 K	25 K	41 K	58 K	58 K	201 K	201 K	252 K	252 K
11	Cout cumulé estimé (en K UC)	360 K	540 K	720 K	900 K	1080 K	1260 K	1890 K	2520 K	3150 K	3780 K
12	Cout réel cumulé (en K UC)	360 K	529 K	684 K	822 K	945 K	1067 K	1496 K	1924 K	2302 K	2680 K
13	Gain cumulé (en K UC)	0	11 K	36 K	77 K	135 K	193 K	394 K	595 K	847 K	1099 K

Synthèse et interprétation

Les résultats de cette simulation montrent que le gain pourrait atteindre les 30% des coûts estimés sans la prise en considération des fonctionnalités auto-X de la plateforme. Ce gain possible est nettement identifiable au niveau des coûts de maintenance. Par ailleurs, nous constatons que le gain en disponibilité pourrait aussi être très élevé grâce au gain en temps. Ainsi, nous représentons la variation des temps et des coûts estimés et réels respectivement aux Figures 5-26 (A et B).

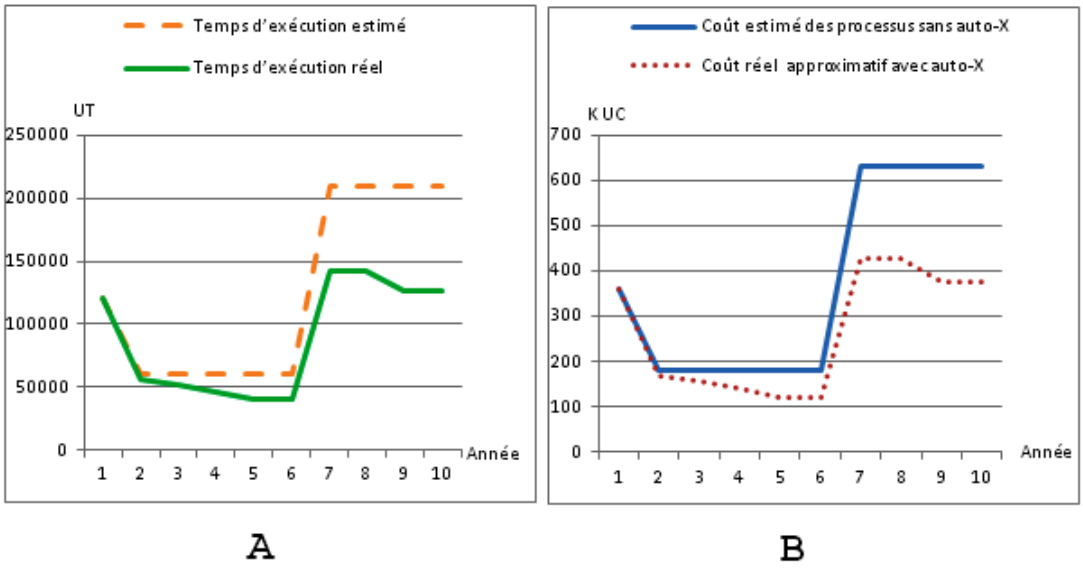


Figure 5-26 Variation des coûts et des temps estimés et réels sur 10 ans

Nous remarquons qu'il y a une baisse importante du coût et du temps à partir de la sixième année. Ce qui signifie que la plateforme de s-maintenance permet de rentabiliser les coûts de maintenance ; c'est-à-dire plus la plateforme apprend plus on obtient une réduction de coûts.

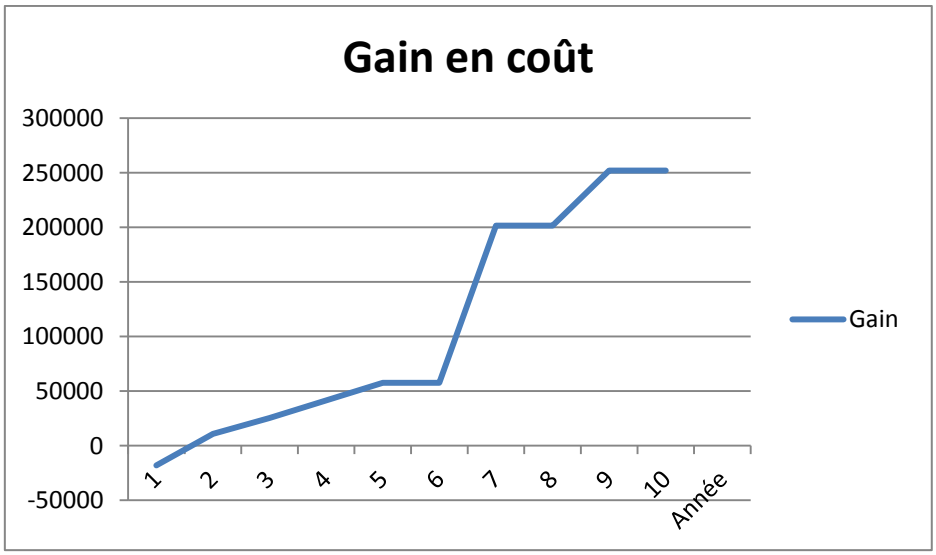


Figure 5-27 Evolution des gains sur 10 ans

En outre, nous remarquons que le gain en temps ou en coût dépend du nombre de processus instanciés, en l'occurrence les phases de jeunesse et de vieillissement des équipements (le nombre de défaillance). De même, le gain (réduction des coûts) évoluerait dans le temps en passant d'une année à une autre (voir Figure 5-27).

D'autre part, dans la pratique, nous devrions tenir compte de la fiabilité du système de surveillance et du dispositif de commutation, ainsi que des applications intégrées (par exemple, diagnostic, pronostic).

7. Conclusion

La dynamique des connaissances des services de la plateforme de s-maintenance prennent appui sur des fonctionnalités d'auto-X (auto-traçabilité, auto-apprentissage et autogestion). La mise en place de ses fonctionnalités s'est faite à l'aide de l'ingénierie des traces qui permet la réutilisation, l'analyse et l'exploitation des interactions faites dans un système.

Partant d'un système à base de traces (SBT) défini pour les environnements d'apprentissages interactifs initialement composé de trois modules interdépendants de base à savoir le système de collecte, le système de transformation et le système de visualisation, nous avons proposé un système à base de traces adapté à la s-maintenance.

Ce SBT prend appui sur l'ontologie du domaine de maintenance (IMAMO) et est composé de deux principaux systèmes gérés par les composants de la plateforme : un système d'auto-traçabilité composé d'un système de collecte et d'un système de transformation ; et un système d'auto-apprentissage.

Le système de collecte qui est géré par le composant coordinateur, enregistre dans des fichiers logs les interactions faites via la plateforme que ce soit entre la plateforme et les utilisateurs ou les interactions entre les composants de la plateforme (activités exécutées).

Le système de transformation géré par le composant *GTM (gestionnaire de traces modélisées)* fait un mapping entre les concepts de l'ontologie (la vue portant sur les processus) et le contenu des fichiers logs des interactions (traces non structurées). Ce mapping a pour objectif d'extraire des traces modélisées afin d'alimenter la base de connaissances et faciliter la tache de leurs interprétation.

Le troisième système de ce SBT est le système d'auto-apprentissage géré par le composant raisonneur. Ce système assure la fonctionnalité d'auto-apprentissage par l'analyse des traces modélisées et l'extraction de nouvelles règles concernant l'exécution et le déroulement des activités des processus gérés par la plateforme. Ce système prend appui sur une méthode à base de vue inspirée des tables de transitions d'état et un algorithme de classifications. Ainsi, l'ajout de ces règles apprises dans la base de connaissances au fur et mesure de l'exécution de la plateforme assure par conséquent l'évolution dynamique souhaitée.

Quant à l'exploitation des connaissances générées par ce SBT, nous avons illustré sur un cas d'exploitation la fonctionnalité d'autogestion du processus de maintenance, et sur un deuxième cas d'exploitation un service de diagnostic dynamique.

D’autre part, l’impact de la plateforme sur les performances de la maintenance a été évalué sur une simulation dédiée au cas d’utilisation d’une entreprise fictive TEM. Cette simulation a portée sur la gestion de 1000 machines par la plateforme de s-maintenance durant une période de 10 ans. La simulation basée sur un ensemble d’indicateurs de coûts et de temps définis, a mis en évidence une réduction en temps de résolution de problème et en coûts de maintenance. En effet, grâce à l’exploitation des fonctionnalités auto-X dans la plateforme, l’entreprise obtiendrait un gain qui pourrait atteindre 30% des coûts estimés ainsi qu’un gain de disponibilité conséquent grâce à la réduction des temps de traitement de la maintenance. De plus, nous avons pu remarquer que le coût pourrait diminuer au cours du temps, grâce à l’expérience de la plateforme et des règles de connaissances extraites.

Conclusion générale

1. Conclusion

La fonction de maintenance tient une position stratégique dans l'organisation de l'entreprise et répond à des besoins permettant de maîtriser techniquement et économiquement le maintien en condition opérationnel des équipements industriels. Vu l'importance grandissante qu'a prise cette fonction, un accent particulier a été mis sur le développement de systèmes informatiques d'aide à la gestion de la maintenance industrielle. Toutefois, les services, les fonctionnalités et les indicateurs fournis par ces systèmes d'aide ne s'adaptent pas à l'évolution des besoins des utilisateurs et nécessitent une modification voire une mutation en un autre type de système informatique d'aide. Ce qui nous a amené à proposer un nouveau concept de système de maintenance le concept de s-maintenance, concrétisé par une plateforme informatique intelligente orientée connaissance répondant aux besoins évolutifs des utilisateurs.

Pour réaliser cet objectif nous avons organisé notre travail en deux parties, une partie concernant la spécification de cette génération de système et une deuxième dédiée à l'élaboration de celui-ci.

En ce qui concerne la spécification, nous avons tout d'abord étudié les enjeux auxquels doit faire face les systèmes informatiques de maintenance de demain en prenant appui sur les projets de référence dans le domaine comme TATEM et les projets de IMS-Center. Les principales caractéristiques de ces systèmes sont le partage la réutilisation des connaissances, la standardisation, le traitement intelligent et les fonctionnements autonomes.

Ainsi, nous avons été amenés à étudier l'évolution des systèmes informatiques de maintenance et nous avons recensé cinq générations de maintenance dont la quatrième est la e-maintenance présentant les systèmes actuels les plus performants et une cinquième génération que nous avons proposé dans ce travail la s-maintenance.

Afin de situer notre nouvelle génération, par rapport à la e-maintenance, nous avons été amenés à faire un état de l'art sur le concept de e-maintenance. Force de constater que la plupart des travaux dans le domaine ne différencient pas le concept, la plateforme et l'utilisation des technologies.

Par conséquent, nous avons proposé deux définitions, une pour le concept de e-maintenance et l'autre pour le modèle de plateforme associée. Puis nous avons donc défini le concept de s-maintenance comme la réalisation de la maintenance basée sur le partage et la réutilisation des connaissances expertes du domaine en fournissant des services adaptatifs et autonomes. Par contre une plateforme de s-maintenance formalise ces connaissance et les partage entre les différentes applications intégrées dans le système, ce qui garantit une interopérabilité technique et sémantique tout en assurant des fonctionnalités auto-X (auto-traçabilité, auto-apprentissage et autogestion) et fournissant des services à la demande.

Une étude sur les plateformes de e-maintenance existantes par rapport aux caractéristiques spécifiques de la s-maintenance, a souligné que ces plateformes ne pouvaient être reprises pour élaborer une plateforme de s-

maintenance. Cette étude a permis d'une part de confirmer les différences existante entre e-maintenance et s-maintenance et d'autre part, d'identifier deux plateformes (IIEMED et POA) basées sur la philosophie de la séparation des modules logiciels (des services pour l'un et des agents pour l'autre) interopérables et exploitant les connaissances. Ces deux plateformes ont pu confirmer nos choix lors de l'élaboration de l'architecture de la plateforme de s-maintenance.

En ce qui concerne le développement de cette architecture, nous avons adopté une architecture à base de composants (ABC) pour le faible couplage entre la structure du système et le niveau d'abstraction élevé, ainsi que lors de la conception de la composition en différents composants fonctionnels et logiques indépendants, fortement interopérables, et extensibles.

Par conséquent, chaque composant dans l'architecture de cette plateforme a été choisi pour répondre à une ou plusieurs des caractéristiques spécifiées. Grâce à une mise en relation (*mapping* composant/fonctionnalités), les composants clés de la plateforme que nous avons identifiés sont : le coordinateur pour la gestion de la collaboration ; La base de connaissance et le gestionnaire de base de connaissances pour le partage et la réutilisation des connaissances ; Le raisonneur en lien direct avec la base de connaissances assurant la fonctionnalité d'auto-apprentissage ; Le gestionnaire de processus pour orchestrer les processus gérés par la plateforme et assurer la fonctionnalité d'autogestion ; Le gestionnaire de traces modélisées assurant la fonctionnalité d'auto-traçabilité ; Le générateur de services pour la gestion des services à la demande répondant aux nouveaux besoins des utilisateurs ; Finalement, le médiateur sémantique adoptant une approche de médiation basée sur l'ontologie du domaine de maintenance permettant d' assurer l'interopérabilité sémantique. Ainsi, les implémentations faites et l'étude de la fiabilité de cette plateforme nous ont permis d'identifier les composants critiques de celle-ci et de proposer une stratégie de duplication afin de réduire les risques de crashs.

En dépit de l'absence d'une ontologie opérationnelle couvrant tous les aspects de la maintenance, nous avons développé une ontologie de domaine de maintenance.

Parmi les méthodologies de création d'ontologies, nous avons choisi la méthodologie METHONTOLOGY, et le langage de description PowerLoom pour l'implémentation de l'ontologie que nous avons appelé IMAMO (Industrial MAintenance Management Ontology.

Ainsi, la création de IMAMO a pris appui sur les modèles standards comme MIMOSA-CRIS et les ontologies du domaine existantes comme SMAC-Model et SOM. Les différentes approches d'évaluation que nous avons appliqué sur IMAMO nous ont permis de conclure que notre ontologie présente un manque au niveau des attributs des concepts, mais qu'elle couvre tous les aspects de la maintenance et peut être exploitée pour garantir l'interopérabilité sémantique entre application et d'être exploitée pour la réutilisation des connaissances.

Toujours, dans le volet élaboration de la plateforme de s-maintenance, nous avons développé un système à base de traces permettant d'assurer les fonctionnalités d'auto-traçabilité et d'auto-apprentissage dans la plateforme. Ces fonctionnalités mettent en évidence la dynamicité des connaissances de la plateforme par la traçabilité et l'analyse des activités effectuées au sein de celle-ci. L'analyse de ces traces nous a permis de fournir un retour d'expérience (sous forme de règles de connaissances) soit pour l'utilisateur de la plateforme soit pour les

applications qu'elle intègre. Par ailleurs cette dynamicité de connaissances a été exploitée au niveau des services fournis par la plateforme ainsi qu'au niveau des fonctionnalités d'autogestion des processus de maintenance.

Finalement, après avoir spécifié et élaboré cette plateforme orientée connaissance, nous avons évalué l'impact possible de celle-ci sur les performances de la gestion de maintenance par la plateforme, sur une simulation dédiée au cas d'utilisation d'une entreprise fictive TEM. Cette évaluation nous a permis de remarquer une possibilité de gain important en temps et en coûts qui pourrait atteindre 30% des coûts estimés ainsi qu'un gain de disponibilité non négligeable grâce à la réduction des temps de traitement de la maintenance dû à la prise en compte du retour d'expérience concrétisé par l'extraction des règles de connaissances sur les activités dans la plateforme.

2. Limites de notre contribution

Nous avons proposé une définition du contexte de s-maintenance qui mettait des contraintes sur la e-maintenance, pour permettre de définir dans une plateforme de s-maintenance des nouvelles fonctionnalités donnant des éléments de réponse aux besoins des *end-users*.

Nous avons proposé une première version de plateforme de s-maintenance, en développant une plateforme orientée connaissances. Ceci en tenant compte des travaux actuels en recherche mais qui ne demande qu'à évoluer ne serait ce que la formalisation des connaissances et de leur évolution pour mieux appréhender leur dynamicité.

Nous avons proposé une ontologie qui demande à être validée par son utilisation effective dans la plateforme par les *end-users* et dont les attributs doivent être étayés.

Nous avons montré la faisabilité de ce type de plateforme, mais cela nécessite une évaluation à grandeur nature pour une validation réelle. Une première plateforme de s-maintenance existe mais ne contient qu'une partie de l'ontologie définie, et demande le développement de certains composants, d'un SBT et des différentes fonctionnalités qui dans ce travail ont été développé en dehors de la plateforme.

D'autre part, en ce qui concerne l'auto-apprentissage, pour être effective ceci demande une période de fonctionnement conséquente de la plateforme permettant une collecte suffisante de l'information. De plus, il définit au plus près les seuils d'apprentissage (la quantité d'information à partir de laquelle l'analyse doit être lancée). Sachant que la collecte d'information dépend du nombre de machines à gérer par cette plateforme, et cet apprentissage n'est utile que par un grand parc de machine ou la réutilisation de l'expérience est fortement nécessaire. Ainsi, la configuration des seuils de validation de l'apprentissage prend appui sur l'expérience de l'expert humain ce qui nous oblige dans un premier temps de les définir empiriquement.

3. Perspectives

A court terme, nous envisageons de développer un médiateur sémantique pour garantir l'interopérabilité sémantique la plateforme. Ceci nécessite d'abord la validation et l'élaboration de différents tests de l'algorithme d'alignement d'ontologie associé à ce médiateur à grandeur nature.

En ce qui concerne l'ontologie, va être implantée entièrement dans la plateforme et être validée au cours du temps en grandeur nature. Ceci peut être par une évolution collaborative avec d'autres acteurs du domaine. De plus, nous envisageons enrichir l'ontologie par l'ajout d'attributs vu sa pauvreté à ce niveau. Nous envisageons aussi l'ajout d'un ensemble d'instances comme par exemple au niveau des concepts de modèle comme *« Maintenance strategy »* qui a *« Conditional maintenance »* comme instance.

D'autre part, afin de facilité l'interopérabilité sémantique entre IMAMO et d'autres ontologies, nous envisageons faire un *mapping* effectif suivant l'approche *bottom-up* avec une ontologie de haut niveau (Dolce, WordNet, SUMO).

A moyen terme, nous envisageons exploiter les différentes vues de l'ontologie à l'aide d'application de maintenance orientée connaissance. Nous désirons élaborer une combinaison des instances de vues structurelles et fonctionnelles concernant un équipement ou un modèle d'équipement pour générer un modèle de comportement *« behavior model »* de l'équipement. Ce modèle pourra ainsi être exploité en faveur de l'application du diagnostic basé sur un raisonnement qualitatif. Un autre exemple, l'exploitation de la vue mémorielle permettra de générer une carte mémoire sur le cycle de vie d'un équipement.

Par ailleurs, la validation de cette plateforme et le concept de s-maintenance nécessite une vérification à grande échelle. Cependant, ceci nécessite un temps de fonctionnement conséquent pour alimenter la base de connaissances, développer les fonctionnalités auto-X et tester ainsi son impact sur le maintien en état opérationnel de l'équipement.

A long terme, la plateforme de s-maintenance pourra évoluer vers une plateforme *SaaS (Software as a Service)* en la développant en une architecture basée sur le *cloud computing* assurant un accès simple rapide et sécurisé. La combinaison des connaissances des différents clients sous cette plateforme *SaaS* rendra plus efficace les fonctionnalités d'auto-traçabilité et d'auto-apprentissage et par conséquent fournir des connaissances plus dynamiques et plus pertinentes.

Bibliographie

Afim. (2011). *Guide national de la maintenance*. Consulté le septembre 2011, sur Association française des ingénieurs et responsables de maintenance: http://www.afim.asso.fr/publications/guide/guide.asp

Akerkar, R., & Srinivas, S. P. (2009). *"Knowledge-based systems".* Sudbury, MA, USA: Jones & Bartlett Publishers.

Ali, A., Chen, Z., Lee, J., & Koc, M. (2002). Web-enabled device-to-business platform for distributed and dynamic decision making systems. *Proceedings of MIM'2002—fifth international conference in managing innovative manufacturing*, (pp. 157-169). Milwaukee, USA.

Altmannshoffer, R. (2006). *Industrielles FM Der Facility Manager*. Récupéré sur http://www.facility-manager.de/

Arnaiz, A., Iung, B., Jantunen, E., Levrat, E., & Gilabert, E. (2007). Dynaweb, a Web Platform for Flexible Provision of E-Maintenance Services. *Proceedings of the second World Congress on Engineering Asset Management (WCEAM).* Harrogate, UK.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (2007). *The Description Logic Handbook: Theory, Implementation and Applications, .* Cambridge University Press.

Bachimont, B. (2004). Pourquoi n'y a-t-il pas d'expérience en ingénierie des connaissances ? *Actes de la conférence « Ingénierie des connaissances (IC2004) », p. 55-64.* Lyon.

Baldwin, R. C. (2004). *How do you spell e-maintenance? .* Récupéré sur www.mt-online

Bangemann, T., Rebeuf, X., Reboul, D., Schulze, A., Szymanski, J., Thomesse, J., et al. (2006). PROTEUS - Creating Distributed Maintenance Systems through an Integration Platform. *Computers in Industry* , 539-551.

Bell, M. (2008). *Service-Oriented Modeling: Service Analysis, Design, and Architecture.* Wiley & Sons.

Bellahsene, Z., Bonifati, A., & Rahm, E. (20111). *Schema Matching and Mapping .* Springer.

Berardi, D., Calvanese, D., & De Giacomo, G. (2005). Reasoning on UML class diagrams. *Artificial Intelligence, 168* , 70–118.

Bertino, E. (1992). A view mechanism for object-oriented databases. *International Conference on Extending Database Technologies (EDBT'92).* Vienna: Lecture Notes in Computer Science, Springer-Verlag.

Bézivin, J. (2000). De la programmation par objets à la modélisation par ontologie. *Journal of Ingénierie de connaissances* .

Bittner, T., Dinnelly, M., & Winter, S. (2006). Ontology and Semantic Interopera-bility. Dans Z. a. Prosperi, *Large-scale 3D Data Integration: Challenges and Opportunities* (pp. 139-160). Taylor&Francis, A CRC press book.

Bouckaert, R. (2010). *WekaManual 3-6-4.*

Bousbia, N. (2011). *Analyse des traces de navigation des apprenants dans un environnement de formation dans une perspective de détection automatique des styles d'apprentissage.* Thèse de Doctorat Université de Pierre et Marie Curie.

Brank, J., Grobelnik, M., & Mladenić, D. (2005). A survey of ontology evaluation techniques. *Proceedings of Conference on Data Mining and Data Warehouses .*

Breen, M. (2005). Experience of using a lightweight formal specification method for a commercial embedded system product line. *Requirements Engineering Journal 10 (2)* .

Campos, J. (2009). Development in the application of ICT in condition monitoring and maintenance. *Computers in Industry 60 (1)* , 1-22.

Candell, O., Karim, R., & So derholm, P. (2009). eMaintenance—Information logistics for maintenance support. *Robotics and Computer-Integrated Manufacturing (25)* , 937–944.

Cao, X., & Jiang, P. (2008). Development of SOA Based Equipments Maintenance Decision Support System. *Intelligence Robotics and Applications* , 576-582.

Chalupsky, H., MacGregor, R., & Russ, T. (2010). *PowerLOOM manual Powerful knowledge representation and reasoning with delivery in Common-Lisp, Java, and C++ Version: 1.48 16.* University of Southern California. .

Champin, P., Prie, Y., & Mille, A. (2004). Musette : a framework for knowledge capture from experience. *Extraction et Gestion des Connaissances EGC'04.* Clermont Ferrand.

Charlet, J., Bachimont, B., & Troncy, R. (2004). Ontologies pour le Web sémantique. *Revue I3, numéro Hors Série «Web sémantique»* .

Charlet, J., Bachimont, B., Bouaud, J., & Zweigenbaum, P. (1996). Ontologie et réutilisabilité : expérience et discussion. Dans *Acquisition et ingénierie des connaissances : tendances actuelles* (pp. 69–87). Cepaduès-éditions.

Chen, D., Doumeingts, G., & Vernadat, F. (2008). Architectures for enterprise integration and interoperability: Past, present and future. *Journal of Computers in Industry 59* , 647–659.

Choquet, C., & Iksal, S. (2007). Modeling Tracks for the Model Driven Reengineering of a TEL System. *JILR, Journal of Interactive earning Research , Special Issue : "Usage Analysis in Learning Systems: Existing Approaches and Scientific Issues", Vol. 18, No. 2.* , 161-184.

Cocheteux, P., Voisin, A., Levrat, E., & Iung, B. (2007). Formalisation du pronostic à base d'une approche processus. *Proceedings de la 3ème Colloque Internationnal Francophone Performance et Nouvelles Technologies en Maintenance (PENTOM).* Mons : Belgique.

Collins English Dictionary. (2011). Collaboration. Complete & Unabridged 10th Edition.. Dictionary.com: HarperCollins.

Corcho, O., Fernández, M., Gómez-Pérez, A., & López-Cima, A. (2005). Building legal ontologies with methontology and webode. Dans L. a. Web, *R. Benjamins, P. Casanovas, J. Breuker, and A. Gangemi* (pp. 142–157). Berlin: Springer-Verlag.

Corcho, O., Fernandez-Lopez, M., & Gomez-Perez, A. (2003). Methodologies, tools and languages for building ontologies: Where is their meeting point? *Data & Knowledge Engineering, Vol. 46* , 41-64.

Cram, D., Jouvin, D., & Mille, A. (2007). Visualisation interactive de traces et réflexivité : application à l'EIAH collaboratif synchrone eMédiathèque. *Journale des Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation. Numéro spécial "Analyses des traces d'utilisation dans les EIAH"* .

Cranefield, S. (2001). Networked Knowledge Representation and Exchange using UML and RDF. *Journal of Digital Information, Vol 1, No 8* .

Cranefield, S., & Purvis, M. (2000). Extending agent messaging to enable OO information exchange. *Proceedings of the 5th European Meeting on Cybernetics and Systems Research.*

Crespo Marquez, A., & Gupta, J. (2006). Contemporary maintenance management: process, framework and supporting pillars. *Omega;34(3)* , 313–326.

Crespo Marquez, A., & Iung, B. (2008). A review of e-maintenance capabilities and challenges. *Journal of Systemics, Cybernetics and Informatics* , 62-66.

EL AOUFIR, H., & BOUAMI, D. (2005). Les coûts directs de la maintenance :De la comptabilité analytique vers la gestion par les activités. *Conference of Integrated Design and Production.* Casablanca, Morocco.

Ermine, J. L. (2000). Challenges and approaches for knowledge management. *Proceedings of the Conference on Principles and Practice of Knowledge Discovery in Databases.*

Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., & Trojahn, C. (2011). Ontology Alignment Evaluation Initiative: six years of experience . *Journal on Data Semantics* .

Fauvet, M., & S. Baina, S. (2001). Evaluation coopérative de requêtes sur des données semi-structurées distribuées. *Proceedings of Information Systems Engineering.*

Fernandez-Lopez, M., & Corcho, O. (2004). *Ontological Engineering.* Berlin: Springer-Verlag.

Fernández-López, M., Gómez-Pérez, A., & Juristo, N. (1997). Methontology: from ontological art towards ontological engineering. *Proceedings of Symposium on Ontological Engineering of AAAI* .

Firestone, J. (1999). Knowledge Base Management Systems and The Knowledge Warehouse: A "Strawman". *KMCI/AIIM KM ANSI/ISO Standards Committee Meeting.* Executive Information Systems.

Främling, K., & Nyman, J. (2008). Information architecture for intelligent products in the internet of things. *Proceedings of 20th NOFOMA logistic conference.* Helsinki: Finland.

Frankovic, B., & Budinska, I. (2006). The role of ontology in building of knowledge systems for industrial applications. *Proceedings of the 4th Slovakian - Hungarian Joint Symposium on Applied Machine Intelligence*, (pp. 15-25).

FÜRST, F. (2002). *L'ingénerie ontologique.* Institut de Recherche en Informatique de Nantes.

Ganek, A. G., & Corbi, T. A. (2003). The dawning of the autonomic computing era. *IBM Systems Journal, vol. 42(1)* , 5-18.

Garlan, D., & Schmerl, B. (2002). Model-based Adaptation for Self-Healing Systems. *Proceedings of the first workshop on Self-healing systems.*

GE-Aviation. (2007, june 19). *TATEM Project Highlights Aircraft Maintenance Technologies*. Récupéré sur Sensors Mag : http://www.sensorsmag.com/sensors-mag/news/tatem-project-highlights-aircraft-maintenance-technologies-2601

Georgeon, O., Mille, A., & Bellet, T. (2006). Musette-Abstract : un outil et une méthodologie pour analyser une activité humaine médiée par un artefact technique complexe. *IC, Ingéniérie des Connaissances.* Nantes.

Gómez-Pérez, A. F. (1996). Towards a Method to Conceptualize Domain Ontologies . *Proceedings of the Workshop on Ontological Engineering. ECAI'96. (pp. 41-52).* Budapest. Hungary.

Gómez-Pérez, A. (1999). Ontological Engineering: A state of the art. *Expert Update, 2(3)* , 33-43.

Granitzer, M., Sabol, V., Weng Onn, K., Lukose, D., & Tochtermann, K. (2010). Ontology Alignment: A Survey with Focus on Visually Supported Semi-Automatic Techniques. *Future Internet* .

Grundstein, M. (2002). *Management des connaissances de l'entreprise: problématique, axe de progrès, orientations.* MG Conseil.

Guarino, N. (1998). Formal Ontology and Information Systems. *Proceedings of FOIS'98: Formal Ontology and Information Systems* (pp. 3-15). Trento: Italy: IOS Press.

Guarino, N. (1997). Understanding, building and using ontologies. *International J. Human-Computer Studies, 46* , 293-310 .

Guarino, N., & Welty, C. (2002). Evaluating Ontological Decisions with OntoClean. *Communications of the ACM. 45(2)* , 61-65.

Gzara, L. (2000). *LES PATTERNS POUR L'INGENIERIE DES SYSTEMES D'INFORMATION PRODUIT.* Thèse de doctorat à l'Institut National Polytechnique de Grenoble.

Halevy, A., & Madhavan, J. (2003). Composing mappings among data sources. *Proceedings of the conference on very large databases*, (pp. 572-583). Berlin, Germany.

Han, T., & Yang, B.-S. (2006). Development of an e-maintenance system integrating advanced techniques. *Computers in Industry 57* , 569–580.

Hayes, B. (2008). Cloud Computing. *Communications of the ACM 50(7)* , 9-10.

Heiler, S. (1995). Semantic Interoperability. *ACM Computing Surveys (CSUR)* .

Hilbert, D. M., & Redmiles, D. F. (2000). Extracting usability information from user interface events . *ACM Computing Surveys, Vol. 32, No. 4.* , 384-421.

Holmberg, K., Helle, A., & Halme, J. (2005). Prognostics for industrial machinery availability. *Proceedings of POHTO'05, International seminar on maintenance, condition monitoring and diagnostics.* Oulu, Finland.

Horn, P. (2001). Autonomic computing: IBM's perspective on the state of information technology. *IBM Corporation 15* .

Hung, M., Chen, K., Ho, R., & Cheng, F. (2003). Development of an e-diagnostics / maintenance framework for semiconductor factories with security considerations. *Advanced Engineering Informatics* , 165-178.

IEEE Computer Dictionary. (1990). *Interoperability.* Compilation of IEEE Standard Computer Glossaries.

IEEE-610.12. (1990). Interoperability. Dans *"IEEE Standard Glossary of Software Engineering Terminology".* Software Engineering, IEEE.

Ikeda, M., Seta, K., & Mizoguchi, R. (1997). Task Ontology Makes It Easier To Use AuthoringTools. *Proceedings of IJCAI-97*, (pp. 342-347). Nagoya, Japan.

Isabel, F., & Xiao, C. H. (2009). Ontology Driven Data Integration in Heterogeneous Networks. *Complex Systems in Knowledge-based Environments: Theory, Models and Applications* , 75-98.

Iung, B., Levrat, E., Crespo Marquez, A., & Erbe, H. (2009). Conceptual framework for e-Maintenance: Illustration by e-Maintenance technologies and platforms. *Annual Reviews in Control 33(2)* , 220-229.

Jae Yang, S., Nieh, J., Selsky, M., & Tiwari, N. (2002). *The Performance of Remote Display Mechanisms for Thin-Client Computing*. Consulté le 2011, sur USENIX : http://www.usenix.org/event/usenix02/full_papers/yang/yang_html/

Jardine, A. K., Daming, L., & Banjevic, D. (2006). review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing* , 1483-1510.

Jones, S., Wilikens, M., Morris, P., & Masera, M. (2000). Trust requirements in ebusiness: A conceptual framework for understanding the needs and concerns of different stakeholders. . *Communications of the ACM, 43, 12* , 81-87.

Kaffel, H. (2001). *La maintenance distribuée: concept, évaluation et mise en oeuvre.* Quebec: Thèse de doctorat, Université Laval,.

Kahn, J. (2003). Overview of MIMOSA and the Open System Architecture for Enterprise Application Integration. *Proceedings of the 8th Condition Monitoring and Diagnostic Engineering Management (COMADEM)*, (pp. 661-670). Växjö University, Sweden.

Karim, R. (2008). *A service-oriented approach to eMaintenance of complex technical systems.* Lulea, Sweden: Doctoral Thesis, Luleå University of Technology.

Karim, R., Kajko-Mattsson, M., & Söderholm, P. (2008). Exploiting SOA within eMaintenance. Liepzig, Germany: Proceedings of the 30th International Conference on Software Engineering (ICSE), the 2nd international workshop on Systems development in SOA environments.

Karray, M. H., Chebel-Morello, B., & Zerhouni, N. (2010). A contextual semantic mediator for a distributed cooperative maintenance platform. *Proceedings of 8th IEEE International Conference on Industrial Informatics (INDIN'10).* Osaka: Japan.

Karray, M. H., Chebel-Morello, B., & Zerhouni, N. (2009). Toward a maintenance semantic architecture. *Proceedings of the Fourth World Congress on Engineering Asset Management (WCEAM)* (pp. 98-111). Athens: Springer-Verlag London .

Karray, M. H., Morello-Chebel, B., & Zerhouni, N. (2010). Evolution of maintenance information systems: "towards s-maintenance". *workshop of Sustainable products and production, services and maintenance.* Zurich: IMS 2020.

Kayser, D. (1997). *La représentation des connaissances.* Hermès.

Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. *Computer 36(1)* , 41-50.

Kiritsis, D. (2004). Ubiquitous product lifecycle management using product embedded information devices. *Proceedings of IMS'2004: International conference on intelligent maintenance systems.* Arles, France.

Kitamura, Y., & Mizoguchi, R. (1999). An Ontological Analysis of Fault Process and Category of Faults. *Proceedings of Tenth International Workshop on Principles of Diagnosis, (pp. 118-128).*

Kitamura, Y., & Mizoguchi, R. (1998). Functional Ontology for Functional Understanding. . *International Workshop on Qualitative Reasoning (QR-98)* (pp. 77-87). Cape Cod, USA: AAAI Press.

Kitchenham, B., on Mayrhauser, A., Niessink, F., Schneidewind, N., Singer, J., Takada, S., et al. (1999). Towards an Ontology of Software Maintenance. *Journal of Software Maintenance: Research and Practice 11* .

Klein, M. (2004). *Change Management for Distributed Ontologies.* PhD thesis, Department of Computer Science, Vrije Universiteit Amsterdam.

Kobbacy, K., & Jeon, J. (2001). The development of a hybrid intelligent maintenance optimization system (HIMOS). *Journal of the operational research society, 52 (7)* , 762-778.

Koc, M., & Lee, J. (2001). A system framework for next-generation e-maintenance system. *Proceedings of the second international symposium on environmentally conscious design and inverse manufacturing.* Tokyo, Japan.

Koc, M., Ni, J., Lee, J., & Bandyopadhyay, P. (2003). Introduction of e-manufacturing. *Proceedings of the 31st North American manufacturing research conference (NAMRC).* Hamilton, Canada.

Kohavi, R., & Quinlan, J. R. (2002). Decision-tree discovery. In Will Klosgen and Jan M. Zytkow, *Handbook of Data Mining and Knowledge Discovery* (pp. 267-276). Oxford University Press.

Komatsoulis, G. A., Warzel, D. B., Hartel, F. W., Shanbhag, K., Chilukuri, R., Fragoso, G., et al. (2008). caCORE version 3: Implementation of a model driven, service-oriented architecture for semantic interoperability. *Journal of biomedical informatics* .

Kramer, I. (2003). *Proteus - Modélisation terminologique,.* INRIA.

Kusper, J. (2007). SMMART - System for Mobile Maintenance Accessible in Real Time. *Proceedings of the 3rd European Workshop on RFID Systems and Technologies (RFID SysTech).* Duisburg: Germany .

Labib, A. W. (2006). Next generation maintenance systems: towards the design of a self-maintenance machine. *Proceedings of the IEEE International Conference on Industrial Informatics*, (pp. 213 - 217). Singapore .

Laflaquière, J., Prié, Y., & Mille, A. (2008). Ingénierie des traces numériques d'interaction comme inscriptions de connaissances. *Actes du 19es Journées Francophones d'Ingénierie des Connaissances (IC 2008),.* Nancy .

Lebold, M., & Thurston, M. (2001). Open Standards for Condition-Based Maintenance and Prognostic Systems. *Proceedings of the 5th Annual Maintenance and Reliability Conference (MARCON).* Gatlinburg, USA.

Lee, J., & Ni, J. (2004). Infotronics-based intelligent maintenance system and its impacts to closed-loop product life cycle systems. *Proceedings of IMS'04:International conference on intelligent maintenance.* Arles, France.

Lee, J., & Wang, H. (2008). New Technologies for Maintenance. Dans K. A. Kobbacy, & D. N. Murthy, *Complex System Maintenance Handbook* (pp. 49-78). Springer Series in Reliability Engineering.

Lee, J., Liao, L., Lapira, E., Ni, J., & Li, L. (2009). Informatics Platform for Designing and Deploying e-Manufacturing Systems. *Collaborative Design and Planning for Digital Manufacturing* , 1-35.

Leondes, C. T. (2000). *Knowledge – Based systems: Techniques and Applications.* Academic Press.

Levrat, E., & Iung, B. (2007). TELMA: A full e-maintenance platform. *Proceedings of the Second World Congress on Engineering Asset Management (WCEAM).* Harrogate.

Levrat, E., Iung, B., & Crespo-Marquez, A. (2008). e-Maintenance: review and conceptual framework. *Production Planning & Control 19 (4)* , 1-22.

Levy, A. Y., Rajaraman, A., & Ordille, J. J. (1996). Querying Heterogeneous Information Sources Using Source Descriptions. *In Intl. Conference on Very Large Data Bases (VLDB)*, (pp. 251-262).

Liyanage, J., & Kumar, U. (2003). Towards a value-based view on operations and maintenance performance management. *Journal of Quality in Maintenance Engineering,Vol. 9* , 333–350 .

Lowe, G., & Shirinzadeh, B. (2005). A Knowledge-base Self-Learning System. *Proceeding (483) ACIT - Automation, Control, and Applications.*

Maedche, A., & Staab, S. (2000). Semi-automatic engineering of ontologies from texts . *Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering (SEKE 2000)*, (pp. 231-239). USA.

Mao, M. (2008). *Ontology mapping: Towards semantic interoperability in distributed and heterogenous environments.* PhD Thesis University of Pittsburgh.

March, S., Hevner, A., & Ram, S. (2000). Research commentary: An agenda for information technology research in heterogeneous and distributed environment. *Information System Research 11, 4* , 327–341.

Matsokis, A., & Kiritsis, D. (2009). An Ontology-based Approach for Product Lifecycle Management. *Computers in Industry. Special Issue: Semantic Web Computing in Industry* .

Matsokis, A., Karray, M., Morello-Chebel, B., & Kiritsis, D. (2010). An Ontology-based Model for providing Semantic Maintenance. *Proceedings of the 1st IFAC workshop on Advanced Maintenance Engineering, Services and Technology (A-MEST'10).*

Matta, N., Ermine, J. L., Aubertin, G., & Trivin, J. Y. (2001). Knowledge Capitalization with a knowledge engineering approach: the MASK method. *In proceedings of IJCAI'2001 workshop on Knowledge Management and Organizational Memory.*

Mellor, S., Scott, K., Uhl, A., & Weise, D. (2002). "Model-Driven Architecture" in Advances in Object-Oriented Information Systems. *Lecture Notes in Computer Science vol. 2426* , 233–239.

Menzies, T. (2000). *Handbook of Software Engineering and Knowledge Engineering.* World Scientific Publishing Company.

Mille, A. (2006). From Case-Based Reasoning to Trace-Based Reasoning. *Annual Reviews in Control* , 223-232.

Mitchell, J., Bond, T., Bever, K., & Manning, N. (1998). MIMOSA – Four Years Later. *Sound and Vibration* , 12-21.

Mizogouchi, R., & Bourdeau, J. (2004). Le rôle de l'ingénierie ontologique dans le domaine des EIAH. *Revue STICEF, Volume 11* .

Mizoguchi, R. (1998). A Step Towards Ontological Engineering. *Proceedings of the 12th National Conference on AI of JSAI.*

Mizoguchi, R. (2004). Tutorial on ontological engineering. . *New Generation Computing 22(2)* , 198-220.

Moore, W. J., & Starr, A. G. (2006). An intelligent maintenance system for continuous cost-based prioritisation of maintenance activities. *Computer in Industrie [special issue on e-maintenance];57(6):* , 595–606.

Moradi, M., & Vallespir, B. (2010). Enterprise modelling and knowledge management: toward a unified enterprise knowledge modelling. *Revue ISDM °36 spécial Gestion des connaissances* .

Mrissa, M., Benslimane, D., Ghedira, C., & Maamar, Z. (2005). Towards a Semantic- and Context-based Approach for Composing Web Services. *International Journal of Web and Grid Services (IJWGS), Vol. 1(3/4):* , 268-286.

Muller, A. (2005). *Contribution à la maintenance prévisionnelle des systèmes de production par la formalisation d'un processus de pronostic.* Thèse de doctorat, Université Henri Poincaré,Nancy.

Muller, A., Marquez, A. C., & Iung, B. (2008). On the concept of e-maintenance: Review and current research. *Journal of Reliability Engineering and System Safety* , 1165–1187.

Nilsson, J. (1998). *Introduction to Machine Learning: An Early Draft of a Proposed Textbook.* http://robotics.stanford.edu/people/nilsson/mlbook.html.

Obrst, L. (2003). Ontologies for Semantically Interoperable Systems. *Proceedings of the twelfth international conference on Information and knowledge management* (pp. 366–369). New York, NY, USA: ACM Press.

Obrst, L., Werner, C., Inderjeet, M., Steve, R., & Smith, B. (2007). The Evaluation of Ontologies: Toward Improved Semantic Interoperability. Dans J. O. Christopher, Baker, & K.-H. Cheung, *Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences.* Springer.

Onori, M., Semere, D., & Lindberg, B. (2011). Evolvable systems: an approach to self-X production. *International Journal of Computer Integrated Manufacturing* , 506-516.

Osborn, S., Sandhu, R., & Munawer, Q. (2000). (2000). "Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security (TISSEC)* , 85–106.

Park, J., & Ram, S. (2004). Information System Interoperability: What Lies Beneath? *ACM Transactions on Information Systems, vol. 22, n°4* .

Pbworks. (2007). *Principles of System Architecture*. Consulté le 2011, sur http://www.pbworks.com: http://sysarch.pbworks.com

Pinhanez, C. (2008). Service Systems as Customer-Intensive Systems and its Implications for Service Science and Engineering. *Proceedings of the 41st Hawaii International Conference on System Sciences.*

Pinto, H. S., Tempich, C., & Staab, S. (2004). Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engingeering of ontologies. *In Proceedings of the 16th European Conference on Arti¯cial Intelligence (ECAI).*

Pollock, J. T. (2001). The Big Issue: Interoperability vs. Integration. *eAI Journal* .

Poole, D., Mackworth, A., & Goebel, R. (1998). *Computational Intelligence: A Logical Approach.* Oxford University Press.

PowerLoom. (2011). Consulté le 2011, sur http://www.isi.edu: http://www.isi.edu/isd/LOOM/PowerLoom/

PROMISE. (2008). *SOM FP6 project. www.promise.no.*

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* San Francisco: Morgan Kaufmann.

Quinlan, J. R. (1990). Induction of Decision Trees. In J. Shavlik, & T. Dietterich, *Readings in Machine Learning* (pp. 57-69). San Francisco: Morgan Kaufmann.

Rahm, E., & Bernstein, P. (2001). A survey of approaches to automatic schema matching,. *The International Journal on Very Large Data Bases, vol. 10, n° 4,* , 334-350.

Rasovska, I. (2006). *Contribution à une méthodologie de capitalisation des connaissances basée sur le raisonnement à partir de cas : Application au diagnostic dans une plateforme d'e-maintenance.* Phd Thesis Université de Franche-Comté.

Rasovska, I., Chebel -Morello, B., & Zerhouni, N. (2005). Process of s-maintenance: decision support system for maintenance intervention. *Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation.* Catania, Italy.

Rasovska, I., Chebel–Morello, B., & Zerhouni, N. (2008). A mix method of knowledge capitalization in maintenance. *Journal of Intelligent Manufacturing Volume 19, Number 3* , 347-359 .

Rasovska, I., Chebel-Morello, M., & Zerhouni, N. (2004). A conceptual model of maintenance process in unified modeling language. *Proceedings of 11 th IFAC Symposium on Information Control Problems in Manufacturing (INCOM).*

Rasovska, I., Morello-Chebel, B., & Zerhouni, N. (2007). Classification des différentes architectures en maintenance . *Proceedings du 7ème Congrès international de génie industriel.* Trois-Rivières, Canada.

Rebeuf, X., Blanc, N., Charpillet, F., Cheve, D., Dutech, A., Lang, C., et al. (2004). Proteus, des web services pour les systèmes de maintenance . *Proceedings de NOTERE'04: Nouvelles Technologies de la Réparation*, (pp. 163-178). Saidia: Maroc.

Retour, D., Bouche, M., & Plauchu, V. (1990). Où va la maintenance industrielle. *Problèmes Économiques, No. 2.159* , 7-13.

Rezgui, Y., Boddy, S., & Wetherill, C. G. (2009). Past, present and future of information and knowledge sharing in theconstruction industry: Towards semantic service-based e-construction? *Journal of Computer-Aided Design* .

Ribeiro, L., Barata, J., & Silvério, N. (2008). A High Level E-Maintenance Architecture to Support on-site Teams. *Enterprise and Work Innovation Studies, Journal of IET Research Center 4(4)* , 129-138.

Rossi, F., Lechevallier, Y., & El Golli, A. (2005). Visualisation de la perception d'un site Web par ses utilisateurs. *Proceedings de la 5ème journées d'Extraction et Gestion des Connaissances (EGC).* (pp. 563-574). Paris: Suzanne Pinson and Nicoles Vincent editors.

Ruh, W. A., Maginnis, F. X., & Brown, J. W. (2001). *Enterprise Application Integration: A Wiley Tech Brief.* John Wiley & Sons.

Ruiz, F., Vizcaino, A., Piattini, M., & García, F. (2004). An Ontology for the management of software maintenance projects. *International Journal of Software Engineering and Knowledge* .

Saint-Voirin, D. (2006). *Contibution à la modélisation et l'analyse des systèmes coopératifs: Application à la e-maintenance.* Thèse de doctorat, Université de Franche-Comté.

Seguy, A. (2008). *Décision collaborative dans les systèmes distribués - Application à la e-maintenance.* Thèse de doctorat INP Toulouse.

Settouti, L., Prié, Y., Mille, A., & Marty, J. (2006). Système à base de traces pour l'apprentissage humain . *TICE, Colloque International en "Technologie de l'Information et de la Communication dans l'Enseignement Supérieur et l'Entreprise".* Toulouse.

Settouti, L.-S., Prié, Y., Marty, J.-C., & Mille, A. (2007). *Vers des Systèmes à Base de Traces modélisées pour les EIAH.* Lyon: rapport de recherche, LIRIS-RR-2007-016.

Shedrof, N. (1999). Information interaction design: a unified field theory of design. *Information Design (Jacobson, R., ed.) MIT Press* , 267–292.

Shenoy, D., & Bhadury, B. (1998). *Maintenance resources management:Adapting MRP.* London: Taylor & Francis.

SMAC. (2009). *http://smac.univ-fcomte.fr*. Récupéré sur http://smac.univ-fcomte.fr

Sofia Pinto, H., Gomez-Perez, A., & Martins, J. (1999). Some Issues on Ontology Integration. *Proceedings of IJCAI99's Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends.*

Spadoni, M. (2004). Système d'information centré sur le modèle CIMOSA dans un contexte d'entreprise étendue. *Journal Européen des Systèmes Automatisés (JESA), Volume 38, n°5* , 497-525.

Stanley, S. (2011). *MTBF, MTTR, MTTF & FIT Explanation of Terms.* IMC Networks.

Stuart, A. (1996). *Knowledge Management.* Récupéré sur CIO Magazine: http://www.cio.com/cio

Tartir, S., Arpinar, I. B., Moore, M., Sheth, A. P., & Aleman-Meza, B. (2005). OntoQA: Metric-based Ontology quality analysis . *Proceedings of IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources.*

Tatir, S., & Budak Arpinar, I. (2007). Ontology Evaluation and Ranking using OntoQA. *Proceedings Int. Conf. on Semantic Computing (ICSC)* .

Thurston, M. G. (2001). An Open Standard for Web-Based Condition-Based. *Proceedings of AUTOTESTCON*, (pp. 401- 415). USA, Valley Forge.

Tsang, A. H. (1995). Condition-Based Maintenance: Tools and Decision Making. *Journal of Quality in Maintenance Engineering Vol. 1, No. 3* , 3-17.

Tsang, A. (2002). Strategic dimensions of maintenance management. *Journal of Quality inMaintenance Engineering 8(1)* , 7-39.

Ucar, M., & Qiu, R. G. (2005). E-maintenance in support of E-automated manufacturing systems. *Journal of the Chinese Institute of Industrial Engineers 22(1)* , 1-10.

Uschold, M., & Gruninger, M. (1996). Ontologies: Principles, Methods and Applications. *Engineering Review 11 No. 2* , 93-113.

Uschold, M., & King, M. (1995). Towards a Methodology for Building Ontologies. *Workshop on Basic Ontological Issues in Knowledge Sharing* .

Van Der Straeten, R., Simmonds, J., Mens, T., & Jonckers, V. (2003). Using description logic to maintain consistency between UML models. *Lecture Notes in Computer Science vol. 2863* , 326–340.

Wang, K. (2003). Intelligent condition monitoring and diagnosis system: a computational intelligent approach. . *Frontiers in Artificial Intelligence and Applications* , 93-132.

Warwick, K., Ekwue, A. O., & Aggarwal, R. (1997). *Artificial Intelligence Techniques in Power Systems.* Power & Energy, Publishing & Inspec.

Watson, M. (2008). *Practical Artificial ntelligence Programming With Java.*

Weiss, G., Zeller, M., & Eilers, D. (2011). Towards Automotive Embedded Systems with Self-X Properties (chapter 21). Dans M. Chiaberge, *New Trends and Developments in Automotive System Engineering.* InTech.

Welty, C., & Andersen, W. (2005). Towards OntoClean 2.0: a framework for rigidity. *Journal of Applied Ontology 1(1)* , 107-116.

Wiederhold, G. (1991). Mediators in the architecture of future information systems. *IEEE Computers.25, 3* , 38–49.

Wiederhold, G., & Genesereth, M. R. (1997). The conceptual basis for mediation services. . *IEEE Expert 12, 5,* , 38–47.

Willems, J. (2008). *From Having to Using: Information Logistics Experience Centre is Born.* (Working Paper). NyenrodeResearch and Innovation Institute, Nyenrode University, .

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems.* John Wiley & Sons Ltd.

worldofcomputing. (2011). *Machine-learning overview*. From intelligence.worldofcomputing.net: //intelligence.worldofcomputing.net/machine-learning/machine-learning-overview.html

Xu, C., Xu, Z., Xiao, P., Zhou, Z., Liu, S., & Jiang, Z. (1995). A self-learning system and its application in fault diagnosis. *and Measurement Technology Conference, IMTC/95. Proceedings of "IEEE Integrating Intelligent Instrumentation and Control".*

Yang, Q., & Zhang, Y. (2006). Semantic interoperability in building design: Methods and tools. *Journal of Computer-Aided Design 38* , 1099–1112.

Yildiz, B. (2006). *Ontology Evolution and Versioning.* Technical Report, TU Vienna, .

Yu, R., Iung, B., & Panetto, H. (2003). A multi-agents based E-maintenance system with case-based reasoning decision support. *Engineering Applications of Artificial Intelligence (16)* , 321–333.

Zephir, O. (2007). Supply chain improvement - assessing readiness for change trough collaboration evaluation. *Proceedings of the Ninth International Conference on Enterprise Information Systems* , (pp. 609-614). Funchal, Portugal.

Zhang, W. (2007). Knowledge Management for E-Maintenance of Industrial Automation Systems. *Studies in Computational Intelligence (SCI)* , 139-155.

Zhang, W., Halang, W., & Diedrich, C. (2003). An agent-based platform for service integration in E-maintenance. *Proceedings of ICIT: IEEE international conference on industrial technology, vol. 1*, (pp. 426– 433). Maribor, Slovenia.

Les Annexes

Annexe 1 : Type de maintenance

Nous présentons dans les paragraphes qui suivent les définitions de chaque type de maintenance (voir Figure 0-1) déjà présentées dans la thèse de Kaffel (Kaffel, 2001).

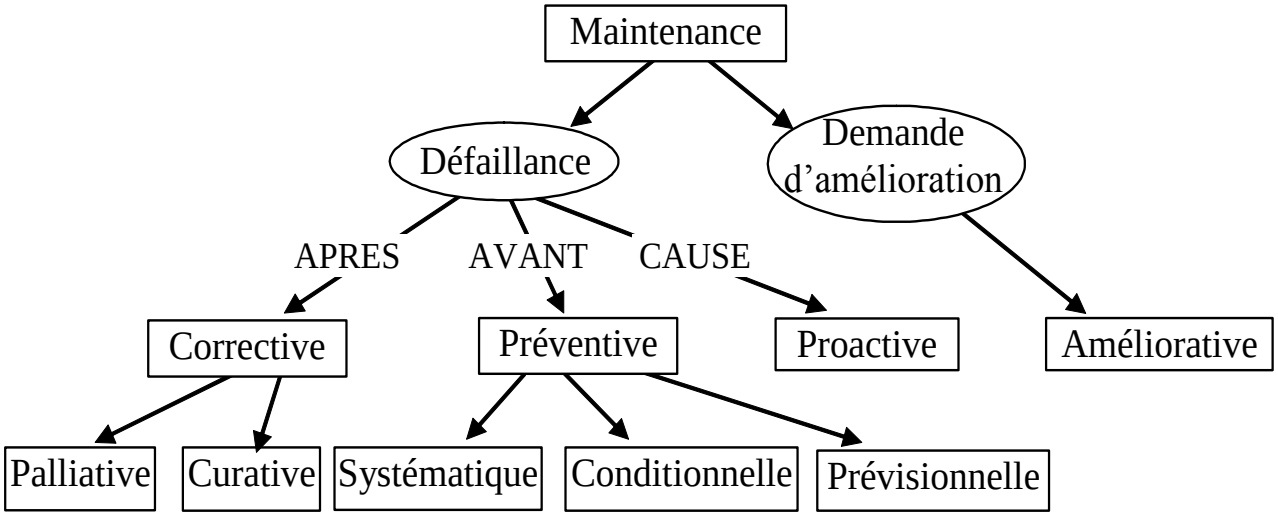


Figure 0-1 Différent types de maintenance

La maintenance corrective est définie comme une maintenance effectuée après défaillance (AFNOR X 60-010). Elle est caractérisée par son caractère aléatoire et requiert des ressources humaines compétentes et des ressources matérielles (pièces de rechange et outillage) disponibles sur place. La maintenance corrective débouche sur deux types d'intervention. Le premier type est à caractère provisoire, ce qui caractérise la maintenance palliative. Le deuxième type est à caractère définitif, ce qui caractérise la maintenance curative.

La maintenance préventive est définie quant à elle comme une maintenance effectuée dans l'intention de réduire la probabilité de défaillance d'un bien ou d'un service rendu. Les activités correspondantes sont déclenchées selon un échéancier établi à partir d'un nombre prédéterminé d'unités d'usage (maintenance systématique) ou de critères prédéterminés significatifs de l'état de dégradation du bien ou du service (maintenance conditionnelle).

La maintenance préventive systématique est une maintenance effectuée selon un échéancier établi selon le temps ou le nombre d'unités d'usage (AFNOR). La périodicité des remplacements est déterminée selon deux méthodes : la première est de type bloc et la seconde, de type âge. La politique de remplacement de type âge suggère de remplacer l'équipement à la panne ou après T unités de temps de bon fonctionnement. La politique de type bloc suggère de remplacer l'équipement après une période prédéterminée de temps T,2T, etc. indépendamment de l'âge et de l'état du composant.

La maintenance préventive conditionnelle est une maintenance subordonnée à un type d'événement prédéterminé (AFNOR). Divers outils comme l'analyse de la vibration et l'analyse d'huile, permettent de détecter les signes d'usure ou de dégradation de l'équipement. Ceci s'effectue en mesurant, à chaque inspection, la valeur d'un paramètre de contrôle tel que l'amplitude de déplacement, de vitesse ou d'accélération des

vibrations, le degré d'acidité, ou la teneur de particule solide dans l'huile. L'action ne se déclenche que lorsque le paramètre de contrôle dépasse un seuil déterminé empiriquement, fixé par le constructeur ou par les normes de santé et de sécurité au travail.

La maintenance prédictive (ou prévisionnelle) est une maintenance préventive subordonnée à l'analyse de l'évolution surveillée de paramètres significatifs de la dégradation du bien, permettant de retarder et de planifier les interventions.

La maintenance proactive est une stratégie de maintenance qui pour la satbilisation de la fiabilité des éuipement. Elle consiste à identifier et à supprimer la cause d'origine des défaillances et à empêcher la réapparition de ces dernières.[Jim C. Fitch, Proactive maintenance. The cost-reduction strategy for the 90s, in Diesel & Gas Worldwide, June 1992, p. 48]

Le self-maintenance est définie comme la capacité d'une machine d'ajuster sa propre fonctionnalité en fonction de son état de santé est une partie intégrante d'un paradigme d'auto-maintenance. Le self-maintenance nécessite à la fois l'intelligence fonctionnelle en fournissant les informations sur l'état de fonctionnement en ligne ainsi que l'intelligence de la santé qui consiste à évaluer l'état de santé actuel et le taux de dégradation, et prédit la probabilité de défaillance [http://www.automation.com/resources-tools/application-stories/machine-monitoring-control/smart-sensor-and-prognostics-for-self-maintenance-machines].

Annexe 2 : Définitions : Données, Information, Connaissance, Coopération et Collaboration

Selon la définition du dictionnaire, la collaboration est le faite de travailler ensemble pour atteindre un objectif [Collins English Dictionary - Complete & Unabridged 10th Edition.. Dictionary.com: HarperCollins].

Selon (Seguy, 2008) la coopération conduit à «participer à un effort commun » et que la collaboration est synonyme de « travailler ensemble pour une tâche commune ».

Un ensemble d'acteurs est dit en coopération s'ils sont impliqués dans une activité commune en assumant une partie individuelle de cette activité. Il s'agit simplement d'une participation à une activité collective. Ce dernier est segmenté en différentes activités, réparties entre les acteurs qui coopèrent. Chaque acteur garde son organisation, ses ressources et ses objectifs (Seguy, 2008).En revanche, un ensemble d'acteurs est dit en collaboration s'ils travaillent ensemble sur une production commune avec un objectif commun même si tout le monde porte une partie de l'activité du groupe. Les acteurs partagent leurs ressources, information, matières, etc. Ce partage peut être réalisé dans un espace commun pour que toutes les ressources du groupe soient disponibles pour tous les acteurs. Ces derniers partagent également les résultats (positifs ou négatifs) avec le concept de partage des risques en cas de défaillance du groupe (Seguy, 2008).

Nous prèsentons les definitions de données, information, connaissances et compréhension (liée à compétence) [Ackoff, R. L., "From Data to Wisdom", Journal of Applies Systems Analysis, Volume 16, 1989 p 3-9.] et [Gene Bellinger, Durval Castro, Anthony Mills, Data, Information, Knowledge, and Wisdom 2004, http://www.systems-thinking.org/dikw/dikw.htm]

Données : les données sont brutes. Elle existe tout simplement et n'a aucune signification au-delà de son existence (en soi). Elle peut exister dans n'importe quelle forme, utilisable ou non. Elle n'a pas de sens en soi. Dans le jargon informatique, un tableur commence généralement par la détention de données.

Information : les informations sont des données qui ont été données un sens par voie de connexion relationnelle. Ce "sens" peut être utile, mais ce n'est pas réellement le cas. Dans le jargon informatique, une base de données relationnelle rend de l'information à partir des données stockées.

La connaissance : la connaissance est la collection appropriée de l'information afin qu'elle soit être utile. La connaissance est un processus déterministe. Quand quelqu'un "mémorise" les informations. Cette connaissance a un sens utile pour eux, mais il ne prévoit pas, en soi, une telle intégration en déduirait de nouvelles connaissances. Dans le jargon informatique, la plupart des applications que nous utilisons (modélisation, simulation, etc) manipule un certain type de connaissances stockées.

La compréhension et la compétence: compréhension est un processus d'interpolation et probabilistes. C'est une compétence cognitive et analytique. Il est le processus par lequel nous pouvons acquérir les connaissances et de synthétiser de nouvelles connaissances à partir des connaissances précédemment acquises. La différence entre la compréhension et la connaissance est la différence entre «apprentissage» et «mémoriser». Les gens qui ont la compréhension peuvent entreprendre des actions utiles, car ils sont capables de synthétiser de nouvelles

connaissances. Dans le jargon informatique, des systèmes d'intelligence artificielle possèdent la compréhension dans le sens où ils sont capables de synthétiser de nouvelles connaissances à partir des informations et des connaissances précédemment stockées.

Annexe 3 : Architectures des systèmes de maintenance

- **La relation d'autonomie** représente un régime sous lequel un système dispose du pouvoir maximal de gestion et est indépendant de tous les autres systèmes et éléments. Il n'y a ni échange ni communication entre ce système et les autres et il doit être auto-suffisant au niveau des informations nécessaires.

- **La relation de communication** est une liaison entre deux ou plusieurs systèmes qui permettent des transferts ou des échanges. Les informations transmises lors de la communication ne se limitent plus aux caractères alphanumériques et comprennent également des images, du son et les séquences vidéo. En contexte, le terme communication est souvent employé comme synonyme de télécommunication.

- **La relation de coopération** représente un travail coopératif qui est accompli par une division du travail dans laquelle chaque acteur est responsable d'une partie de la résolution du problème. Dans notre contexte, il s'agit surtout de la coopération technologique et industrielle donc un accord de coopération conclu entre des systèmes indépendants qui s'engagent à réaliser des projets communs de production des services de maintenance.

- **La relation de collaboration** représente un partenariat stratégique en vue d'atteindre à l'excellence en combinant des compétences, des fournisseurs ou des produits divers. La collaboration implique un engagement mutuel des acteurs dans un effort coordonné pour résoudre ensemble le problème mettant en commun des ressources, des informations et des compétences en vue de mieux adapter les organisations à leur environnement.

- **Le système de maintenance** comprend un seul système informatique présent sur le site de production et utilisé sur le site de maintenance. Ce système est autonome sans échange de données avec d'autres systèmes. En parallèle avec la classification des entreprises, cela correspond à l'entreprise traditionnelle, donc nous parlons d'une architecture traditionnelle d'un système d'information.

- **Le système de télémaintenance** est constitué d'au moins deux systèmes informatiques un émetteur et un récepteur de données et d'informations qui échangent à distance. Selon la définition d'AFNOR la télémaintenance est « la maintenance d'un bien exécutée sans accès physique du personnel au bien ». Nous parlons d'une architecture distribuée, basée sur la notion de distance qui permet de transférer les données par une radio, une ligne téléphonique ou par l'intermédiaire d'un réseau local.

- Avec l'extension d'Internet, les systèmes de télémaintenance émergent vers le concept de **e-maintenance.** Le système d'e-maintenance sera implémenté sur une plateforme distribuée coopérative intégrant différents systèmes et applications de maintenance. Cette plateforme doit prendre appui sur le réseau mondial d'Internet (d'où le terme e-maintenance) et la technologie web permet d'échanger, de partager et de distribuer des données et des informations et de créer ensemble des connaissances. Ici le concept de la maintenance intelligente peut être exploité et les stratégies de maintenance proactives et coopératives sont mises en place.

- Enfin, nous proposons une architecture destinée à améliorer la performance de l'architecture d'e-maintenance au niveau de la communication et de l'échange des données entre les systèmes et qui permet de tenir compte de la sémantique des données traitées dans les applications - la **s-maintenance** (où « s » signifie sémantique) (Rasovska & al, 2005). Nous décrirons à la section 5 ce concept qui prend appui sur le web sémantique.

Annexe ONTOL

Vue des événements

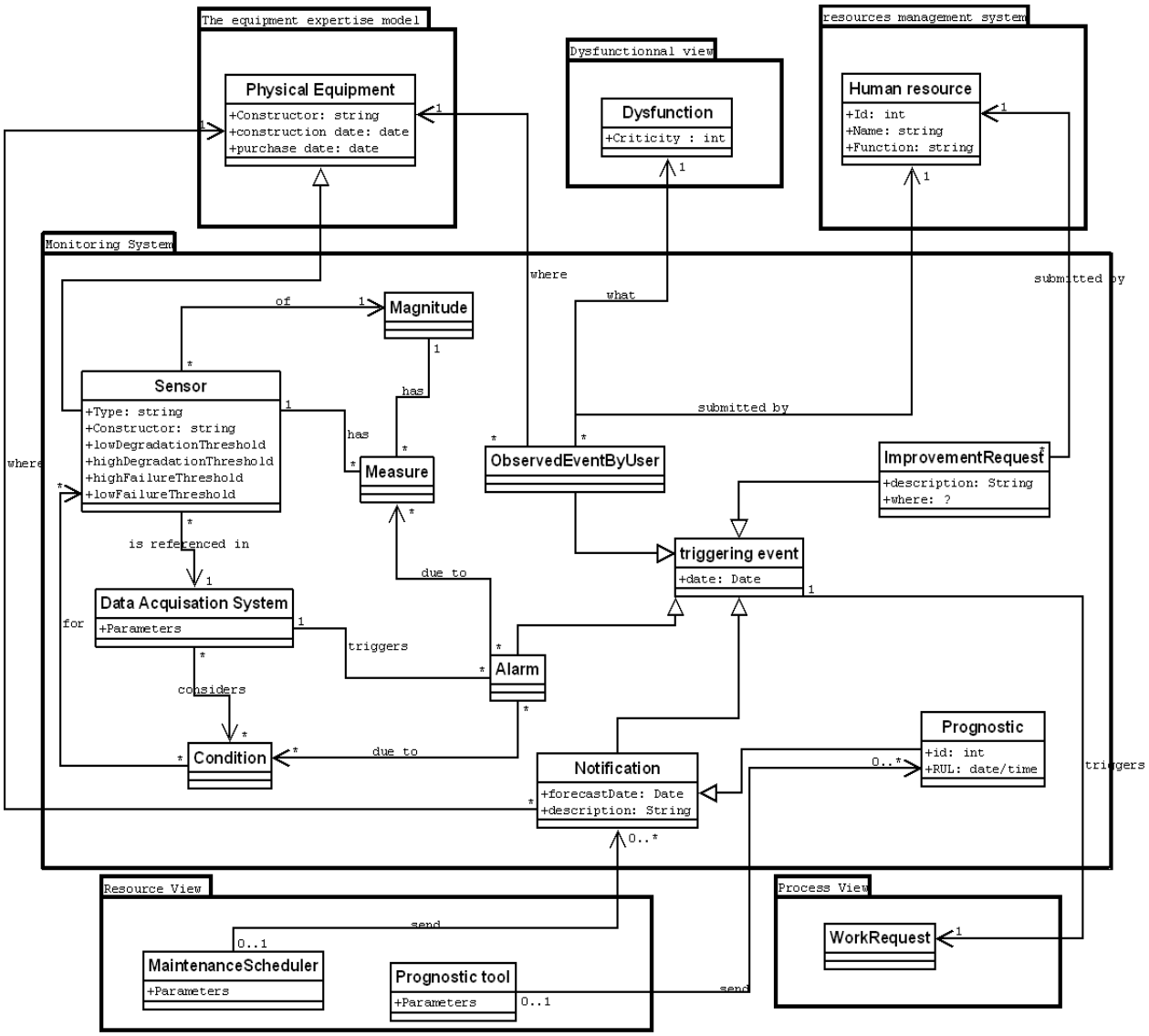


Figure 0-2 Vue événementielle dans IMAMO

Tableau 0-1 Dictionnaire de données de la vue événementielle

Concept	Termes anglais	Termes français	Synonymes anglais	Description
measure;	Measure;	Mesure;	Measurement ;	Nombre, mesure ou la quantité capturée par un capteur.
magnitude;	Magnitude;	Grandeur;		Une grandeur d'une taille ou d'une quantité. Elle présente la propriété d'une mesure relative.
Data	Data	Système		Un système logiciel (abrégé par

acquisition system;	acquisition system;	d'acquisition de données;		l'acronyme DAS ou DAQ) typiquement converti les signaux analogiques généralement récupérés à partir des capteurs en valeurs numériques pour le traitement.
condition;	Condition;	Condition;		Exigence environnemental ou fonctionnel définie pour superviser (tâches de surveillance) un équipement physique spécifique ou un lieu (site, par exemple) par l'utilisation de capteurs et de système d'acquisition de données.
triggering event;	Triggering event;	Evénement; Evénement déclencheur;	Event;	Quelque chose qui arrive à un équipement physique et un temps donnés qui déclenche un processus spécifique de maintenance qui est le processus de demande de travail.
alarm;	Alarm;	Alarme;		Type d'événement déclencheur lancé à partir d'un système d'acquisition de données indiquant qu'il y a une mesure à partir d'un capteur à violer certaines conditions sur un équipement ou environnement spécifique.
improvement request;	Improvement request;	Demande d'amélioration;		Un événement déclencheur spécifique ou général demandant une amélioration d'un équipement physique. Sachant que l'amélioration est la combinaison de toutes les mesures techniques, administratives et de gestion, destinée à améliorer la fiabilité d'un équipement physique, sans changer sa fonction requise.
event observed by user;	Event observed by user;	Evénement signalé par l'utilisateur; Evénement observé par l'utilisateur;		Type d'événement déclencheur d'un dysfonctionnement sur un équipement physique observé par l'utilisateur qui est une ressource humaine.
notification;	Notification;	Notification; Avis;		Type d'événement déclencheur informant sur les événements futurs pour la maintenance systématique planifiée ou le RUL du pronostic.
prognostic;	Prognostic;	Pronostic;		Type de notification composée par l'état de santé à un moment futur et la vie utile restante (RUL) d'un équipement physique. C'est la sortie de l'outil de pronostic.
Prognostic tool;	Prognostic tool;	Outil de pronostic;		La description du concept sera présentée dans la vue informationnelle.
Maintenance scheduler;	Maintenance scheduler;	Planificateur de maintenance, Ordonnanceur de maintenance;		La description du concept sera présentée dans la vue informationnelle.

Vue fonctionnelle et dysfonctionnelle

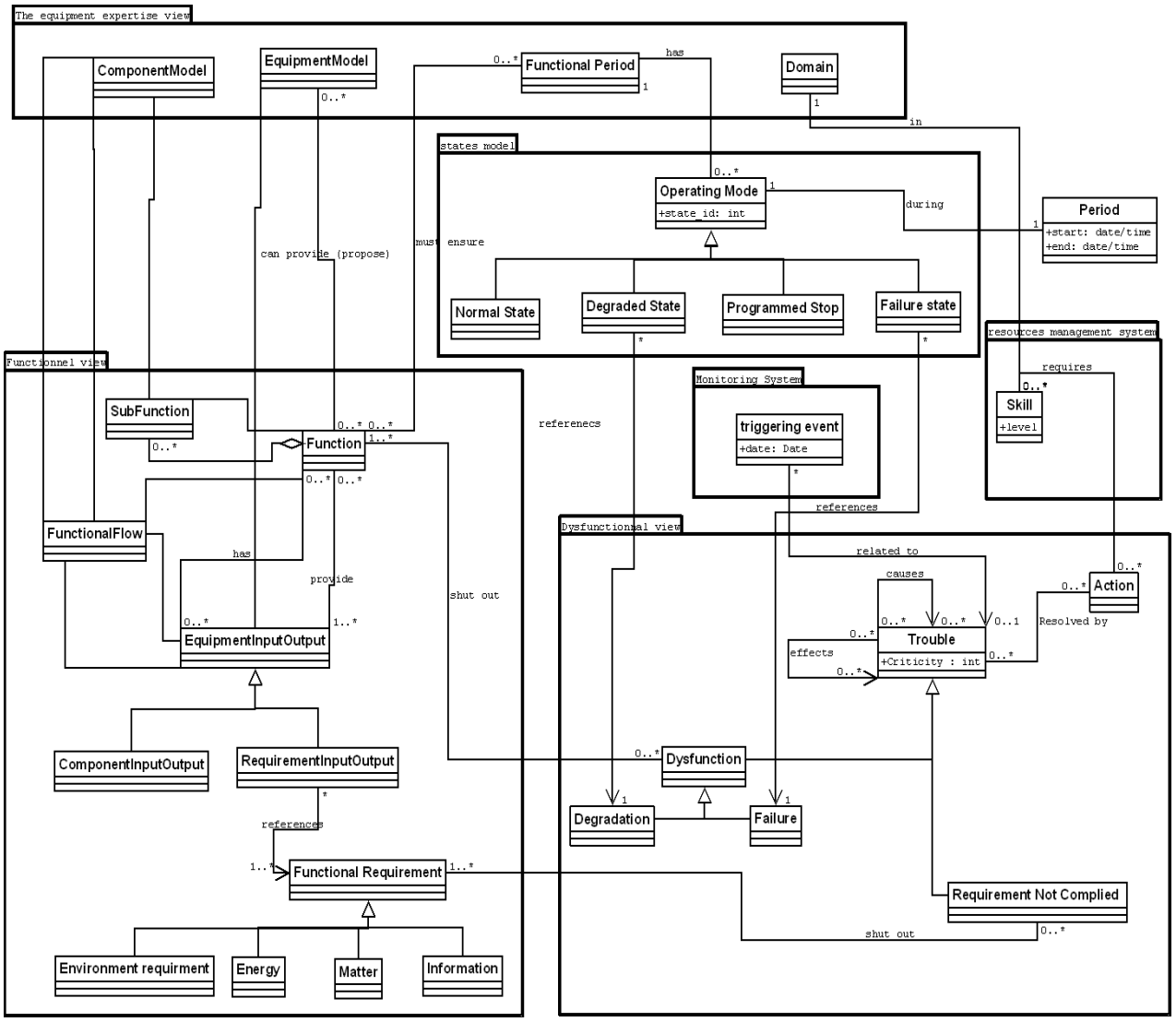


Figure 0-3 Vue fonctionnelle et dysfonctionnelle dans IMAMO

Tableau 0-2 Dictionnaire de données de la vue fonctionnelle et dysfonctionnelle

Concept	Termes anglais	Termes français	Synonymes anglais	Description
function;	Function	Fonction	Feature;	Fonction ou combinaison de fonctions d'un équipement physique qui sont jugées nécessaires pour fournir un service donné.
Sub-function;	Sub-function	Sous-fonction		Fonction élémentaire dans une combinaison de fonctions.
functional requirement;	Functional requirement	Besoin fonctionnel; Exigence fonctionnelle;	Functional need; Essential resource;	Besoin requis par un équipement physique pour effectuer une fonction particulière. Il fait référence à l'entrée ou à la sortie d'un équipement physique (par exemple l'imprimante a besoin du papier, de l'énergie électrique et de l'encre d'impression

				pour effectuer la fonction d'impression).
Environment requirement;	Environment requirement;	Besoin environnemental;	Environment need;	Type de besoin fonctionnel présentant les besoins environnementaux (température, humidité, etc) de l'équipement physique pour effectuer ses fonctions.
matter;	Matter;	Matière première;		Type de besoin fonctionnel présentant les matières premières (papier, huile, etc) nécessaires en entrée pour un équipement physique afin d'effectuer une fonction ou présentant le produit matériel fourni comme sortie de l'équipement physique en exerçant une fonction.
information;	Information;	Information; Commande;		Type de besoin fonctionnel présentant une information (par exemple une valeur, des commandes PLC, etc) nécessaire en entrée par un équipement physique à exercer une fonction ou l'information fournie en tant que sortie de l'équipement physique, tout en exerçant une fonction.
energy;	Energy;	Énergie;		Type de besoin fonctionnel présentant une quantité thermodynamique équivalent à la capacité d'un système physique (par exemple, électrique, hydraulique, etc.) nécessaires en entrée par un équipement physique pour exercer une fonction ou l'énergie fournie en sortie de l'équipement physique en exerçant une fonction.
functional flow;	Functional flow;	Flux fonctionnel;		Flux relationnel entre deux composants défini par un flux d'entrée / sortie.
equipment input;	Equipment input;	Entrée d'un équipement;		Entrée nécessaire par un équipement physique pour exercer une fonction. Il permet la gestion des flux fonctionnels.
equipment output;	Equipement output;	Sortie d'un équipement;		Sortie assurée par un équipement physique en exerçant une fonction. Il permet la gestion des flux fonctionnels.
operating mode;	Operating mode;	Mode opérationnel; Mode fonctionnel;	Availability performance;	Aptitude d'un équipement physique pour être en état d'accomplir une fonction requise dans des conditions données à un instant de temps donné ou pendant un intervalle de temps donné, en supposant que les ressources externes nécessaires sont fournis.
normal state;	Normal state;	État normal; État fonctionnel;	Operating state	Un mode opérationnel spécifique. Etat dans lequel un équipement physique fourni ses fonctions requises.
degraded state;	Degraded state;	État dégradé;		Un mode opérationnel spécifique. Etat d'un équipement physique par laquelle cet équipement physique continue à exercer une fonction à des limites acceptables, mais qui sont inférieures aux valeurs spécifiées ou continues pour effectuer seulement une partie de ses fonctions requises.
programmed stop;	Programmed stop;	Arrêt programmé;	Stand by state ;	Un mode opérationnel spécifique. Etat de non-fonctionnement instantané pendant un temps nécessaire déjà programmé.
failure state;	Failure state;	État de panne;	Fault ;	Un mode opérationnel spécifique. État d'un équipement physique caractérisé par l'incapacité d'accomplir une fonction requise, à l'exclusion de l'incapacité pendant la maintenance préventive ou d'autres actions

				prévues comme l'arrêt programmé, ou en raison du manque de besoins fonctionnels.
trouble;	Trouble;	Trouble;		Source de difficulté, problème.
dysfunction;	Dysfunction;	Dysfonctionnement,		Trouble particulier présentant une anatomie de la fonction, cela signifie que le matériel physique n'est pas en mesure d'accomplir une fonction requise.
cause;	Cause;	Cause;	Origin;	Relation entre troubles. C'est la raison conduisant à un trouble. Les raisons peuvent être le résultat d'un ou plusieurs des éléments suivants: défaut de conception, défaut de fabrication, l'échec d'installation, utilisation incorrecte, une mauvaise manipulation, et trouble de maintenance connexes.
effect;	Effect;	Effet;		Relation entre troubles. Conséquence qui suit et causée par certains effets précédents.
degradation;	Degradation;	Dégradation;		Type spécifique de trouble. Un processus irréversible dans une ou plusieurs caractéristiques d'un équipement soit avec le temps, l'utilisation ou une cause extérieure.
failure;	Failure;	Panne;	Fault;	Type spécifique de trouble. Cessation de l'aptitude d'un équipement à accomplir une fonction requise.
action;	Action;	Action; Acte;		Action physique prise pour récupérer et éliminer un trouble.
skill;	Skill;	Compétence; Capacité;	Competence; Capacity;	Capacité acquise par un effort délibéré, systématique et soutenue en douceur et adaptative pour mener des activités complexes ou des tâches impliquant des idées, des choses (compétences techniques), et / ou des personnes (compétences interpersonnelles).
requirement not complied;	Requirement not complied;	Besoin fonctionnel non fourni; Exigence fonctionnel non respectée;		Trouble particulier (spécifique) causé par un non-respect des besoins (exigences) fonctionnelles nécessaires pour assurer la fonction de l'équipement physique

Vue des informations

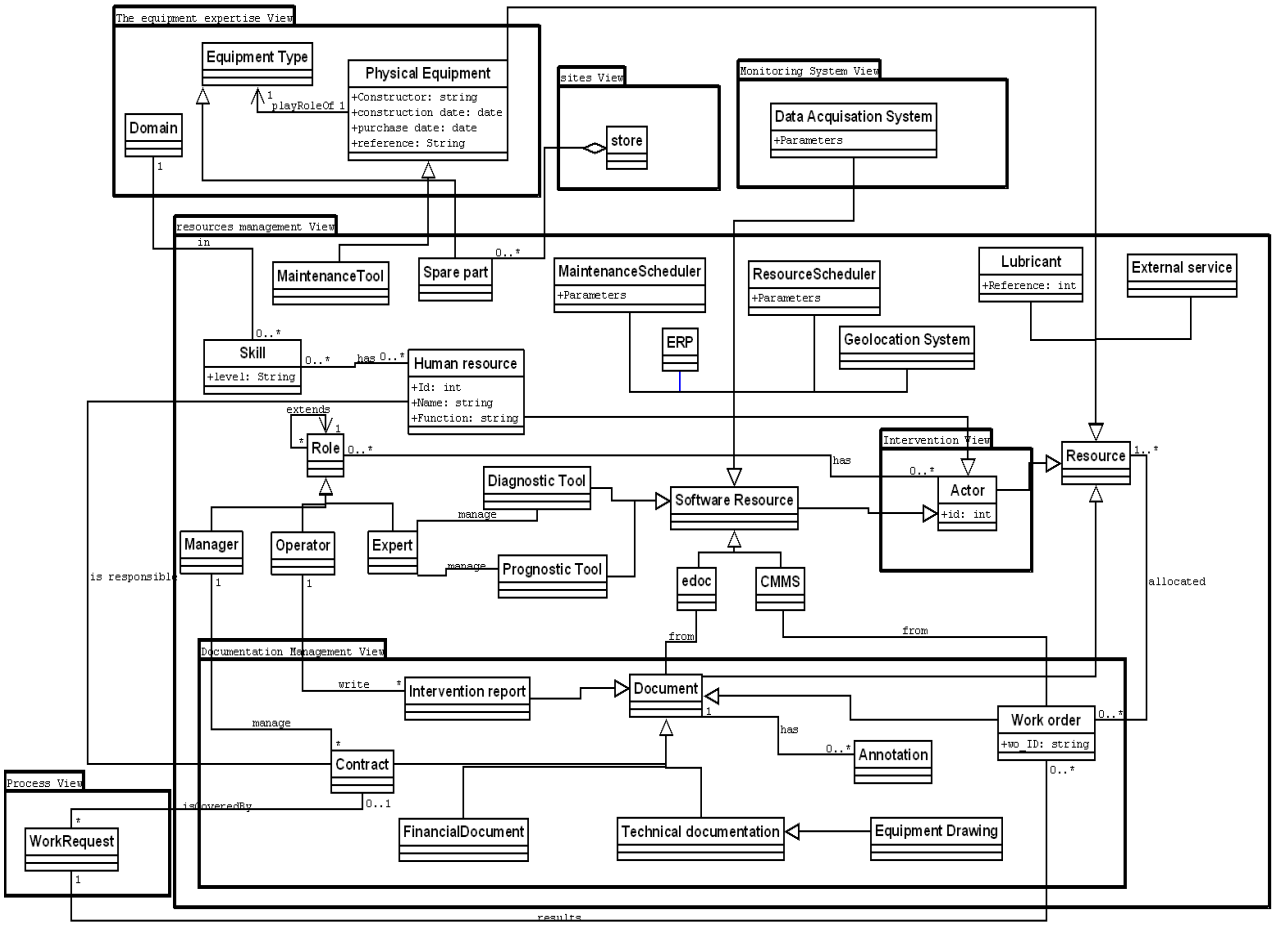


Figure 0-4 Vue informationnelle dans IMAMO

Tableau 0-3 Dictionnaire de données de la vue informationnelle

Concept	Termes anglais	Termes français	Synonymes anglais	Description
Resource;	Resource;	Ressource;	Maintenance support	Ressources et services de gestion nécessaires pour effectuer l'entretien.
actor;	Actor;	Acteur;		Personne ou système informatique qui interagit dans le processus de maintenance.
role;	Role;	Role;		Prescrits ou comportements prévus associés à une position particulière ou de statut dans le processus de maintenance.
manager;	Manager;	Manager; Gestionnaire; Administrateur; Responsible;		Rôle joué par un acteur dans le processus de maintenance. Il est le responsable de la gestion des tâches dans le processus de maintenance.
Operator;	Operator;	Opérateur; Technicien; Ouvrier; réparateur;	Technician;	Rôle joué par un acteur dans le processus de maintenance. Il est le responsable des tâches de maintenance et surtout le processus d'intervention et les actions de réparation.
expert;	Expert;	Expert; Ingénieur de		Rôle joué par un acteur dans le processus de maintenance. Il est le responsable de

		maintenance;		maintenance des tâches de surveillance, de diagnostic, des analyses pronostiques, et en particulier d'identifier les causes dysfonctionnelles et des actions de réparation à faire.
human resource;	Human resource;	Ressource humaine;		Acteurs humains qui contribuent à toutes les actions techniques, administratives et de gestion dans le processus de maintenance.
document;	Document;	Document;		Elément contenant quelques informations concernant le domaine de la maintenance. Il est contrôlé et identifié par un numéro dans un système de gestion de documents.
contract;	Contract ;	Contrat ;	Deal ; Agreement ; Engagement ;	Type spécifique de document définissant une entente ou accord de service entre un prestataire et un propriétaire ou un exploiteur équipements physiques pour maintenir un ensemble d'équipements physiques. Le contrat peut être réalisée sur un modèle d'équipements (par exemple pour tous les moteurs), sur une ou plusieurs zones (par exemple le site Peugeot de Belfort) ou sur un ensemble d'équipements de production (par exemple station robotisée 1).
financial document;	Financial document;	Document financier;		Type spécifique de document contenant des informations financières sur l'ensemble des actions techniques, administratives et managériales.
technical documentation ;	Technical documentation;	Document technique;		Type spécifique de document contenant des informations techniques sur les équipements physiques ou des ressources logicielles (comme le manuel utilisateur, SADT, etc.)
equipment draw ;	Equipment draw;	Plan de l'équipement;		Type de documentation technique. Il contient les informations sur la conception de l'équipement physique.
work order ;	Work order;	Ordre de travail; Ordre de mission; Ordre d'intervention;	job order; job ticket; work ticket;	Un ordre édité par un acteur pour planifier une intervention sur une demande de travail. Il est considéré comme un type spécifique de document contenant des informations sur les ressources allouées à l'intervention. Il est reçu par les acteurs désignés pour assurer les activités composant l'intervention.
work request;	Work request;	Demande de travail; Demande d'intervention;	Intervention request ;	Un type spécifique de document édité par un acteur lorsqu'un événement déclencheur est détecté. Chaque demande de travail est couverte par un contrat de maintenance. L'édition d'un document de demande de travail lance le processus de demande de travail.
software resource;	Software resource;	Ressource logiciel;		Acteur logiciel contribuant à toutes les actions techniques, administratives et de gestion dans le processus de maintenance.
Maintenance scheduler;	Maintenance scheduler;	Planificateur de maintenance, Ordonanceur de maintenance;		Type de ressource logicielle permettant la planification, l'attribution de temps considérable, et le degré élevé de coordination entre différents départements. Il est généralement initié par un ordre de travail.
resource	resource	Ordonanceur de		Type de ressource logicielle permettant aux

scheduler;	scheduler;	ressource;		acteurs de la maintenance de réserver tout type de ressources tels que les pièces de rechange, des ressources humaines, des outils de maintenance, voitures de société, et d'autres. Il permet la vérification de la disponibilité des ressources et réservant en ligne des ressources.
geo-location system;	Geo-location system ;	Système de géolocalisation ;		Type de ressource logicielle (type GPS, balises ...) qui permet d'identifier l'emplacement exact ou approximatif (zone par exemple) d'un équipement de production (même les équipements mobiles).
e-doc system;	e-doc system;	Système de gestion électronique de document;		Type de ressource logicielle appelé système de gestion de documents qui est un système informatique (ou ensemble de programmes informatiques) utilisé pour suivre et stocker les documents électroniques et / ou des images de documents papier.
CMMS;	CMMS;	GMAO;		Type de ressource logicielle. Système informatisé de gestion de la maintenance (GMAO), également connu comme Enterprise Asset Management et Système informatisé de gestion de la maintenance (SIGM). Le but de la GMAO est de simplifier la planification et les fonctions administratives de la maintenance, les achats et la gestion des stocks.
ERP;	ERP;	ERP;		Type de ressource logicielle. L'acronyme de « Enterprise Resource Planning » qui est un système informatique intégré utilisé pour gérer les ressources internes et externes, y compris les immobilisations corporelles, les ressources financières, des matériaux et des ressources humaines.
Prognostic tool;	Prognostic tool;	Outil de pronostic;		Type de ressource logicielle. Outil logiciel ou un système permettant de prédire et estimer le reste du temps à la défaillance et le risque de l'existence ultérieure d'un ou plusieurs modes de défaillance d'un niveau de confiance qui est une valeur indiquant le degré de certitude que le pronostic est correct.
Diagnostic tool;	Diagnostic tool;	Outil de diagnostic;		Type de ressource logicielle utilisé pour identifier les problèmes de reconnaissance, la localisation des troubles, les caractéristiques d'un dysfonctionnement particulier et les causes. Dans certains cas, il peut fournir ou recommander des actions à faire pour remédier le trouble.
Diagnostic;	Diagnostic;	Diagnostic;		Résultat fourni par un outil de diagnostic. Il est principalement le tuple (uplet) composé par le (trouble, la localisation, la cause, les actions).
external service;	External service;	Service externe;		Considéré comme un type de ressources ou de services fournis par un organisme externe à l'entreprise.
Lubricant;	Lubricant;	Lubrifiant; Consommables;	Consumable;	Substances consommables comme la graisse ou d'huile utilisée pour réduire la friction entre les composants à entretenir le

				matériel physique. C'est un type de ressource.

Vue des interventions

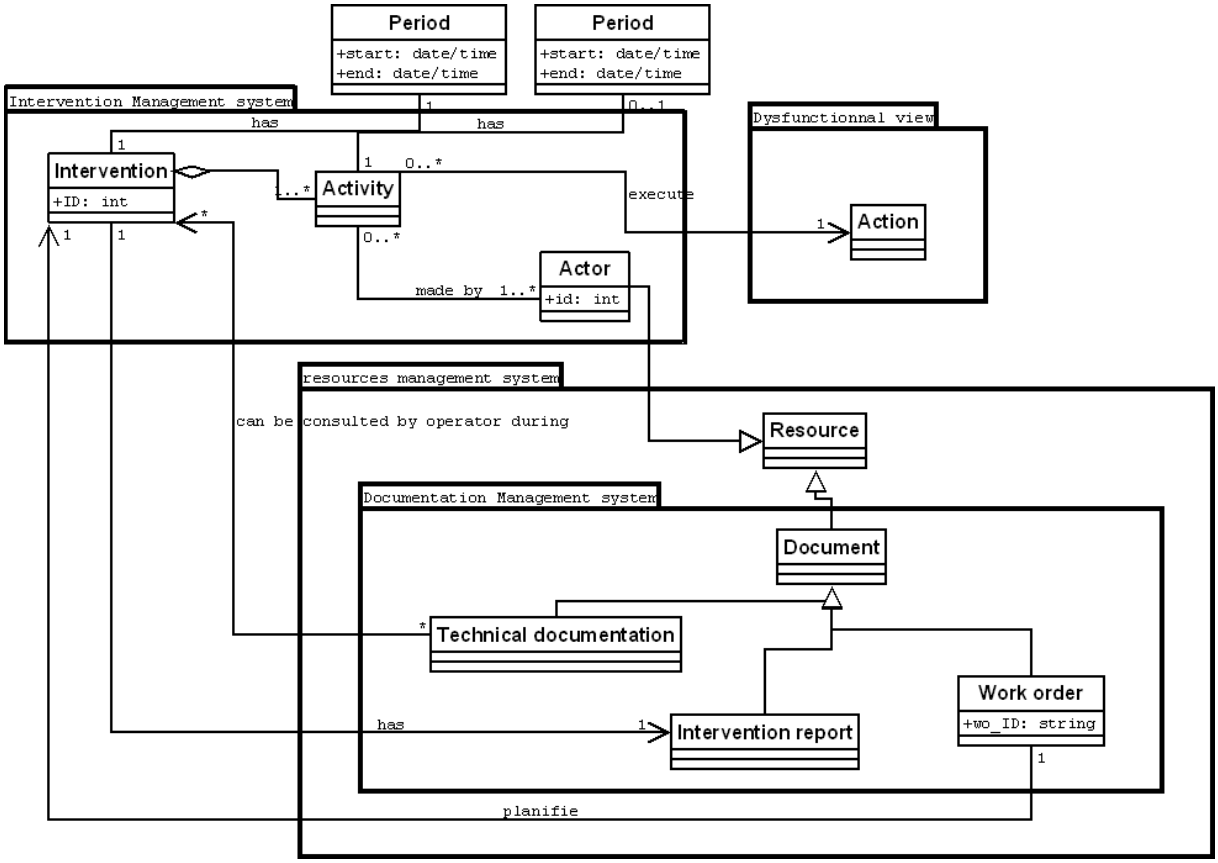


Figure 0-5 Vue intervention dans IMAMO

Tableau 0-4 Dictionnaire de données de la vue intervention

Concept	Termes anglais	Termes français	Synonymes anglais	Description
intervention;	Intervention;	Intervention;		Type spécifique de processus concernant la partie principale technique de maintenance qui est au cœur de la maintenance.
activity;	Activity;	Activité;		Unité d'organisation pour effectuer une action spécifique assurée par un acteur.
intervention report;	Intervention report;	Rapport d'intervention;		Type spécifique de document édité contenant des informations sur l'intervention comme les observations et les remarques faites par les acteurs. Ce type de document est exploité pour le processus de retour d'expérience.

Vue des stratégies

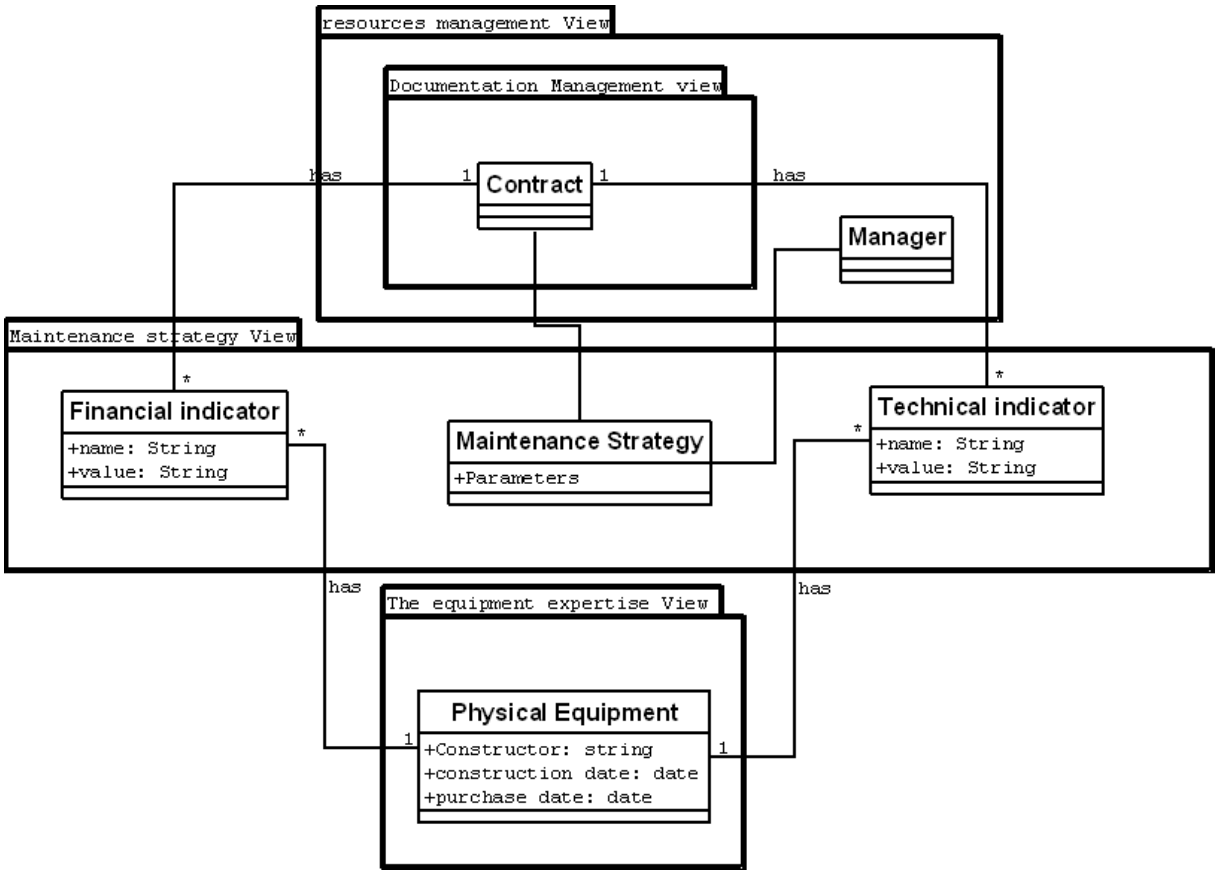


Figure 0-6 Vue de stratégie dans IMAMO

Tableau 0-5 Dictionnaire de données de la vue de stratégie

Concept	Termes anglais	Termes français	Synonymes anglais	Description
Maintenance strategy;	Maintenance strategy;	Stratégie de maintenance;		Un plan à long terme, couvrant tous les aspects de la gestion de la maintenance qui définit l'orientation de cette gestion, et contient des plans d'action de l'entreprise pour parvenir à un état futur souhaité pour la fonction de maintenance en respectant le type de maintenance.
technical indicator;	Technical indicator;	Indicateur technique;		Valeur calculée à partir de différents facteurs techniques de la maintenance sur un équipement physique maintenu sous un contrat. Il présente une évaluation technique spécifique du contrat.
financial indicator;	Financial indicator;	Indicateur financier		Valeur calculée à partir de différents facteurs financiers et de gestion de la maintenance sur un équipement physique maintenu sous un contrat. Il présente une évaluation spécifique financière du contrat.

Vue des processus

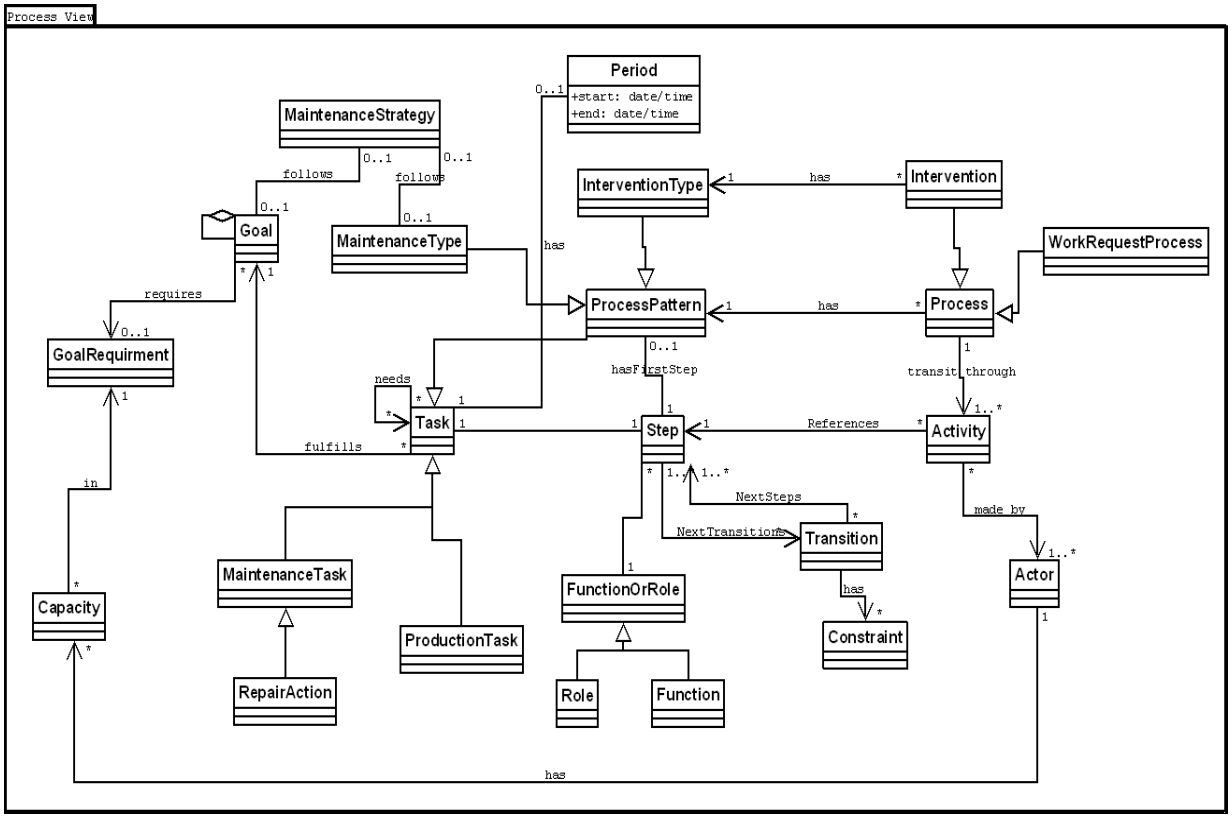


Figure 0-7 Vue des processus dans IMAMO

Tableau 0-6 Dictionnaire de données de la vue des processus

Concept	Termes anglais	Termes français	Synonymes anglais	Description
goal;	Goal;	But; Objectif;	Purpose ; Aim ;	Objectif à atteindre. Elle est suivie par la stratégie de maintenance et rempli par un ensemble de tâches (par exemple la disponibilité de 80%).
goal requirement;	goal requirement;	Exigence de l'objectif; Besoin de l'objectif;		Besoin nécessaire pour atteindre un but (par exemple reproduire les pièces de rechange des équipements physiques critique).
process;	Process;	Processus;		Séquence d'activités interdépendantes et liées, qui, à chaque étape, consomme une ou plusieurs ressources (employé à temps, l'énergie, des machines, de l'argent) pour convertir les entrées (données, documents, pièces, etc.) à des sorties, tout en respectant un modèle de processus (le modèle de processus présente le modèle général du processus). Ces sorties de transition servent comme entrée de transition pour l'étape suivante jusqu'à ce qu'un but soit atteint connu.
process pattern;	Process pattern;	Modèle de processus; Pattern de processus;		Motif qui décrit une série d'actions. Un modèle est une description d'une solution générale à un problème commun ou une question à partir de laquelle une solution détaillée à un problème spécifique peut être

				déterminée.
maintenance type;	Maintenance type;	Type de maintenance;		Type spécifique de modèle de processus concernant la méthode pour effectuer la maintenance et l'orchestration des processus utilisés en vue d'atteindre les objectifs de la stratégie de maintenance. Il y a 12 types possibles de maintenance qui sont: maintenance préventive, maintenance planifiée, maintenance prédéterminée, maintenance basée sur les conditions, maintenance prédictive, maintenance corrective, maintenance à distance, maintenance différée, maintenance immédiate, maintenance en ligne, maintenance sur site et maintenance de l'opérateur. (la traduction en anglais est la suivante : Preventive maintenance, Scheduled maintenance, Predetermined maintenance, Condition based maintenance, Predictive maintenance, Corrective maintenance, Remote maintenance, Deferred maintenance, Immediate maintenance, On line maintenance, On-site maintenance and Operator maintenance.)
maintenance task;	Maintenance task;	Tâche de maintenance;		Type de tâche spécifique concernant une partie des travaux de maintenance (par exemple : réparation, remplacement, inspection, lubrification, etc.)
intervention type;	Intervention type;	Type d'intervention;		Type spécifique de modèle de processus. Il présente le modèle générique d'une intervention.
task;	Task;	Tâche;		Partie de travail effectuée par un acteur dans le cadre de ses fonctions. La tâche est évaluée en regardant son résultat en termes d'exhaustivité, d'exactitude, de tolérance, de clarté, d'erreur ou de quantité.
repair action;	Repair action;	Action de reparation;		Type spécifique de tâches de maintenance définie comme une action physique prises pour rétablir la fonction requise d'un équipement physique défectueux.
production task;	Production task;	Tâche de production;		Type de tâche spécifique qui se termine par un produit discret ou un résultat qui est observable et mesurable.
constraint;	Constraint;	Contrainte;		Condition restrictive pour contrôler les transitions entre les étapes.
activity;	Activity;	Activité;		Unité d'organisation à exécuter une action spécifique. Une activité est l'exécution d'une tâche que ce soit une activité physique ou l'exécution de code. Il présente l'activité exercée par l'acteur dans le monde réel.
step;	Step;	Étape;	Work step ;	Manœuvre effectuée dans le cadre de progression vers l'état d'avancement d'un processus. Elle est référencée par une activité.
work request process;	Work request process;	Processus de demande d'intervention;		Type spécifique de processus lancé automatiquement ou par un acteur lors de la réception d'une demande de travail. Il permet de gérer la demande de travail jusqu'à la résolution du problème d'origine de l'événement déclencheur et la fin du processus d'intervention, y compris l'édition du rapport d'intervention.

Vue de milieu de vie

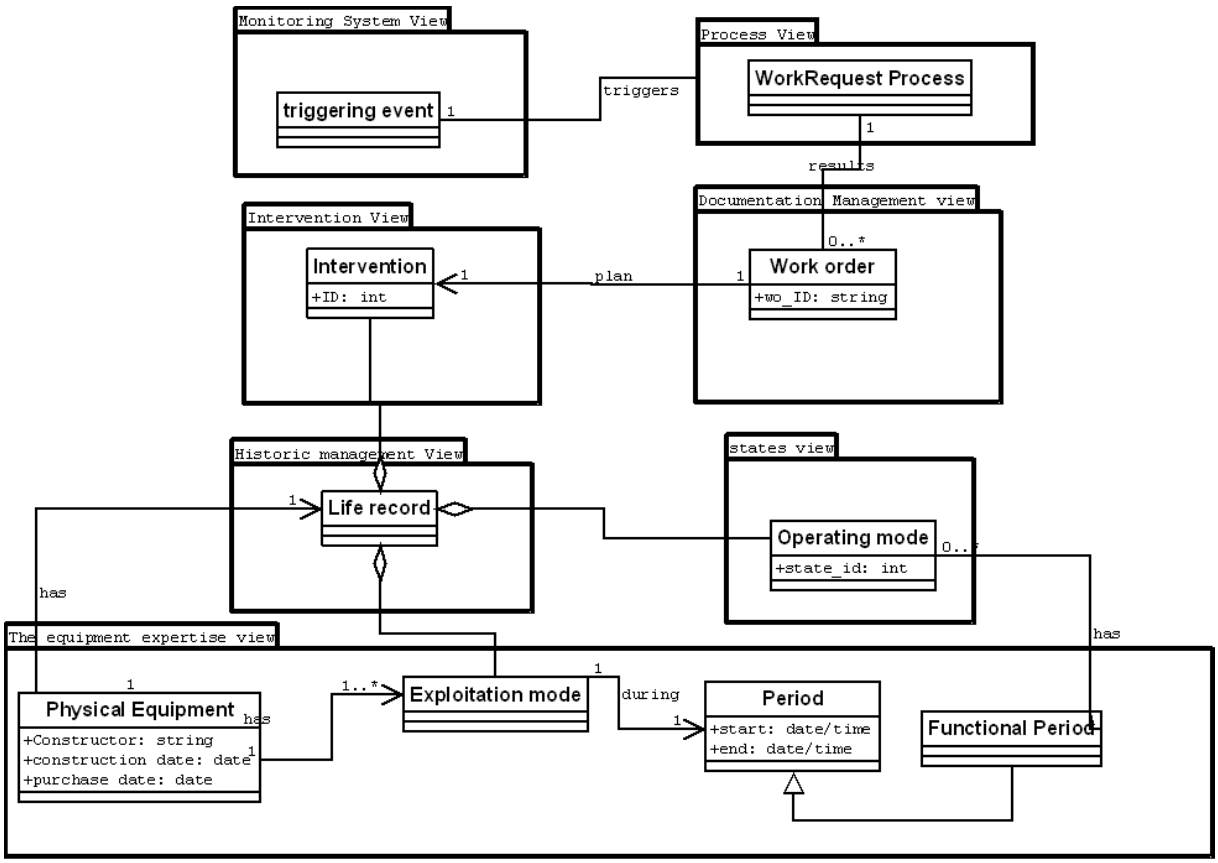


Figure 0-8 Vue de milieu de vie dans IMAMO

Tableau 0-7 Dictionnaire de données de la vue de milieu de vie

Concept	Termes anglais	Termes français	Synonymes anglais	Description
Life record;	Life record;	Mémoire de vie; Enregistrement de l'historique de vie; Carte mémoire de cycle de vie ;	Life-Cycle-Phase;	D'une approche PLM, la mémoire de vie est ajoutée comme un concept contenant les éléments d'information sur le milieu de la phase de vie d'un équipement physique (par exemple durant de l'exploitation, de l'achat au démontage).

Annexe OPL

```
;;; -*- Mode: Lisp; Package: STELLA; Syntax: COMMON-LISP; Base: 10 -*-
(CL:IN-PACKAGE "STELLA")
(DEFMODULE "/PL-KERNEL-KB/PL-USER/ONTOLOGIE-MAINTENANCE"
  :DOCUMENTATION "Module for Maintenance"
  :INCLUDES ("PL-USER"))
(IN-MODULE "/PL-KERNEL-KB/PL-USER/ONTOLOGIE-MAINTENANCE")
(IN-DIALECT :KIF)
(DEFCONCEPT COMPONENT)
(DEFCONCEPT FONCTIONAL-COMPONENT (?C COMPONENT))
(DEFCONCEPT ADDITIONAL-COMPONENT (?C COMPONENT))
(DEFCONCEPT PHYSICAL-EQUIPMENT ((?A ASSET)))
(DEFRELATION PHYSICAL-EQUIPMENT-ID ((?C PHYSICAL-EQUIPMENT) (?ID STRING)))
(DEFRELATION PHYSICAL-EQUIPMENT-TYPE ((?C PHYSICAL-EQUIPMENT) (?TYPE STRING)))
(DEFRELATION PHYSICAL-EQUIPMENT-CONSTRUCTOR ((?C PHYSICAL-EQUIPMENT) (?CONSTRUCTOR
STRING)))
(DEFRELATION PHYSICAL-EQUIPMENT-ACTUAL-LOCATION ((?C PHYSICAL-EQUIPMENT) (?ACTUAL-
LOCATION STRING)))
(DEFRELATION    PHYSICAL-EQUIPMENT-HABITUAL-LOCATION    ((?C    PHYSICAL-EQUIPMENT)
(?HABITUAL-LOCATION STRING)))
(DEFRELATION    PHYSICAL-EQUIPMENT-CONSTRUCTION-DATE    ((?C    PHYSICAL-EQUIPMENT)
(?CONSTRUCTION-DATE STRING)))
(DEFRELATION PHYSICAL-EQUIPMENT-PURSHASE-DATE ((?C PHYSICAL-EQUIPMENT) (?PURSHASE-
DATE STRING)))
(DEFRELATION  PHYSICAL-EQUIPMENT-FUNCTIONNING-START-DATE  ((?C  PHYSICAL-EQUIPMENT)
(?FUNCTIONNING-START-DATE STRING)))
(DEFFUNCTION FUNCTIONNAL-DEGREE ((?C PHYSICAL-EQUIPMENT))
  :-> (?N INTEGER))
(DEFCONCEPT EQUIPMENT-MODEL-GROUP)
(DEFCONCEPT STATE)
(DEFCONCEPT NORMAL-STATE (?C STATE))
(DEFCONCEPT DEGRADED-STATE (?C STATE))
(DEFCONCEPT FAILURE-STATE (?C STATE))
(DEFCONCEPT PROGRAMMED-STOP (?C STATE))
(DEFRELATION STATE-ID ((?C STATE) (?ID INTEGER)))
(DEFRELATION STATE-STATE-START-DATE ((?C STATE) (?STATE-START-DATE STRING)))
(DEFRELATION STATE-STATE-END-DATE ((?C STATE) (?STATE-END-DATE STRING)))
(DEFRELATION STATE-STATE-START-TIME ((?C STATE) (?STATE-START-TIME STRING)))
(DEFRELATION STATE-STATE-END-TIME ((?C STATE) (?STATE-END-TIME STRING)))
(DEFCONCEPT FUNCTIONNAL-EQUIPMENT-MODEL)
(DEFCONCEPT MAIN-FUNCTION (?C FUNCTIONNAL-EQUIPMENT-MODEL))
(DEFCONCEPT SECOND-FUNCTION (?C FUNCTIONNAL-EQUIPMENT-MODEL))
(DEFCONCEPT FAILURE-EQUIPMENT-MODEL)
(DEFCONCEPT SYMPTOM)
(DEFCONCEPT ORIGIN)
```

```
(DEFCONCEPT ACTION)
(DEFCONCEPT FAILURE)
(DEFCONCEPT CHARACTERISTICS)
(DEFRELATION FAILURE-CHARACTER ((?C FAILURE) (?CHARACTER STRING)))
(DEFRELATION FAILURE-EQUIPMENT ((?C FAILURE) (?EQUIPMENT STRING)))
(DEFRELATION CHARACTERISTICS-CRITICITY ((?C CHARACTERISTICS) (?CRITICITY INTEGER)))
(DEFRELATION CHARACTERISTICS-FREQUENCY ((?C CHARACTERISTICS) (?FREQUENCY INTEGER)))
(DEFRELATION CHARACTERISTICS-NONDETECTION ((?C CHARACTERISTICS) (?NONDETECTION
INTEGER)))
(DEFRELATION CHARACTERISTICS-GRAVITY ((?C CHARACTERISTICS) (?GRAVITY INTEGER)))
(DEFCONCEPT ACTIVITY)
(DEFCONCEPT MAINTENANCE (?C ACTIVITY))
(DEFCONCEPT CORRECTIVE-MAINTENANCE (?C MAINTENANCE))
(DEFCONCEPT PREDECTIVE-MAINTENANCE (?C MAINTENANCE))
(DEFCONCEPT PREVENTIVE-MAINTENANCE (?C MAINTENANCE))
(DEFCONCEPT DIAGNOSTIC)
(DEFCONCEPT REPARATION)
(DEFCONCEPT MONITORING)
(DEFCONCEPT PROGNOSTIC)
(DEFCONCEPT UPKEEP)
(DEFCONCEPT ACTOR)
(DEFCONCEPT INTERVENTION (?C ACTIVITY))
(DEFRELATION ACTOR-ID ((?C ACTOR) (?ID INTEGER)))
(DEFRELATION INTERVENTION-ID ((?C INTERVENTION) (?ID INTEGER)))
(DEFRELATION EQUIPMENT-HAS-TOP-MODEL ((?E PHYSICAL-EQUIPMENT) (?MG EQUIPMENT-MODEL-
GROUP)))
(DEFRELATION EQUIPMENT-MODEL-GROUP-INHERITS ((?MG1 EQUIPMENT-MODEL-GROUP) (?MG2
EQUIPMENT-MODEL-GROUP)))
(DEFRELATION EQUIPMENT-COMPONENT-COMPOSED ((?E PHYSICAL-EQUIPMENT) (?C COMPONENT)))
(DEFCONCEPT CRITICAL-COMPONENT ((?P PHYSICAL-EQUIPMENT))
  :<<=>> (AND (PHYSICAL-EQUIPMENT ?P) (> (FUNCTIONNAL-DEGREE ?P) 6)))
(DEFRELATION EQUIPMENT-COMPOSED ((?E PHYSICAL-EQUIPMENT) (?COM PHYSICAL-
EQUIPMENT)))
(DEFCONCEPT WORK-REQUEST)
(DEFCONCEPT SENSOR)
(DEFCONCEPT MEASURE)
(DEFCONCEPT DATA-ACQUISITION-SYSTEM)
(DEFRELATION DATA-ACQUISITION-SYSTEM-CATCH-MEASURE ((?DAS DATA-ACQUISITION-SYSTEM)
(?M MEASURE)))
(DEFCONCEPT TRIGGERING-EVENT)
(DEFRELATION MEASURE-TRIGGER-EVENT ((?M MEASURE) (?T TRIGGERING-EVENT)))
(DEFCONCEPT WORK-ORDER)
(DEFRELATION WORK-ORDER-WO-START-DATE ((?C WORK-ORDER) (?WO-START-DATE STRING)))
(DEFRELATION WORK-ORDER-WO-START-TIME ((?C WORK-ORDER) (?WO-START-TIME STRING)))
(DEFRELATION WORK-ORDER-WO-END-DATE ((?C WORK-ORDER) (?WO-END-DATE STRING)))
(DEFRELATION WORK-ORDER-WO-END-TIME ((?C WORK-ORDER) (?WO-END-TIME STRING)))
```

```
(DEFCONCEPT LIFE-HISTORY)
(DEFCONCEPT MAINTENANCE-STRATEGY-MODEL)
(DEFCONCEPT FINANCIAL-INDICATOR (?C MAINTENANCE-STRATEGY-MODEL))
(DEFRELATION FINANCIAL-INDICATOR-LABOUR-COST ((?C FINANCIAL-INDICATOR) (?LABOUR-
COST INTEGER)))
(DEFRELATION FINANCIAL-INDICATOR-SPARE-PART-COST ((?C FINANCIAL-INDICATOR) (?SPARE-
PART-COST INTEGER)))
(DEFRELATION FINANCIAL-INDICATOR-NON-PRODUCTION-COST ((?C FINANCIAL-INDICATOR)
(?NON-PRODUCTION-COST INTEGER)))
(DEFRELATION FINANCIAL-INDICATOR-EQUIPMENT-PRICE ((?C FINANCIAL-INDICATOR)
(?EQUIPMENT-PRICE INTEGER)))
(DEFCONCEPT TECHNICAL-INDICATOR (?C MAINTENANCE-STRATEGY-MODEL))
(DEFRELATION TECHNICAL-INDICATOR-MAINTENANCE-INDICATOR ((?C TECHNICAL-INDICATOR)
(?MAINTENANCE-INDICATOR STRING)))
(DEFRELATION TECHNICAL-INDICATOR-EQUIPEMENT-TYPE ((?C TECHNICAL-INDICATOR)
(?EQUIPEMENT-TYPE STRING)))
(DEFRELATION TECHNICAL-INDICATOR-PRODUCTION-MODE ((?C TECHNICAL-INDICATOR)
(?PRODUCTION-MODE INTEGER)))
(DEFRELATION TECHNICAL-INDICATOR-MAINTENANCE-TIME ((?C TECHNICAL-INDICATOR)
(?MAINTENANCE-TIME STRING)))
(DEFCONCEPT RESSOURCE)
(DEFCONCEPT HUMAN-RESSOURCE (?C RESSOURCE))
(DEFCONCEPT SOFTWARERESSOURCE (?C RESSOURCE))
(DEFRELATION HUMAN-RESSOURCE-ID ((?C HUMAN-RESSOURCE) (?ID INTEGER)))
(DEFRELATION HUMAN-RESSOURCE-NAME ((?C HUMAN-RESSOURCE) (?NAME STRING)))
(DEFRELATION HUMAN-RESSOURCE-FUNCTION ((?C HUMAN-RESSOURCE) (?FUNCTION STRING)))
(DEFCONCEPT ROLE)
(DEFCONCEPT MANAGER (?C HUMAN-RESSOURCE))
(DEFCONCEPT EXPERT (?C HUMAN-RESSOURCE))
(DEFCONCEPT OPERATOR (?C HUMAN-RESSOURCE))
(DEFCONCEPT MATERIAL-RESSOURCE (?C RESSOURCE))
(DEFRELATION MATERIAL-RESSOURCE-ID ((?C MATERIAL-RESSOURCE) (?ID STRING)))
(DEFRELATION MATERIAL-RESSOURCE-COST ((?C MATERIAL-RESSOURCE) (?COST INTEGER)))
(DEFRELATION MATERIAL-RESSOURCE-QUANTITY ((?C MATERIAL-RESSOURCE) (?QUANTITY
INTEGER)))
(DEFCONCEPT TOOL (?C MATERIAL-RESSOURCE))
(DEFRELATION TOOL-TYPE ((?C TOOL) (?TYPE STRING)))
(DEFRELATION TOOL-EMPLACEMENT ((?C TOOL) (?EMPLACEMENT STRING)))
(DEFRELATION TOOL-PROPERTY ((?C TOOL) (?PROPERTY INTEGER)))
(DEFCONCEPT SPARE-PART (?C MATERIAL-RESSOURCE))
(DEFRELATION SPARE-PART-REFERENCE ((?C SPARE-PART) (?REFERENCE INTEGER)))
(DEFCONCEPT CONSUMABLE (?C MATERIAL-RESSOURCE))
(DEFRELATION CONSUMABLE-REFERENCE ((?C CONSUMABLE) (?REFERENCE INTEGER)))
(DEFCONCEPT DOCUMENT (?C RESSOURCE))
(DEFCONCEPT INTERVENTIONREPORT (?C DOCUMENT))
(DEFCONCEPT OBSERVATION)
```

```
(DEFRELATION OBSERVATION-SYMPTOM ((?C OBSERVATION) (?SYMPTOM STRING)))
(DEFRELATION OBSERVATION-ORIGIN ((?C OBSERVATION) (?ORIGIN STRING)))
(DEFRELATION OBSERVATION-ACTION ((?C OBSERVATION) (?ACTION STRING)))
(DEFCONCEPT TECHNICAL-COMMENT)
(DEFCONCEPT CONTRACT)
(DEFCONCEPT TECHNICAL-DOCUMENTATION)
(DEFCONCEPT EQUIPMENT-DRAWING)
(DEFRELATION HAS-STATE ((?PE PHYSICAL-EQUIPMENT) (?S STATE)))
(DEFCONCEPT SITE)
(DEFRELATION SITE-EMPLACEMENT ((?C SITE) (?EMPLACEMENT STRING)))
(DEFCONCEPT STORE (?C SITE))
(DEFCONCEPT PRODUCTION-SITE (?C SITE))
(DEFCONCEPT MAINTENANCE-CENTER (?C SITE))
(DEFCONCEPT FIXED-SITE (?C PRODUCTION-SITE))
(DEFCONCEPT MODULE-SITE (?C PRODUCTION-SITE))
(DEFRELATION ACTIVITY-ENSURED-BY ((?I ACTIVITY) (?O ACTOR)))
(DEFRELATION HAS-WORK-ORDER ((?I INTERVENTION) (?WO WORK-ORDER)))
(DEFRELATION HAS-WORK-REQUEST ((?I INTERVENTION) (?WR WORK-REQUEST)))
(DEFRELATION WORK-ORDER-CREATED-BY ((?WO WORK-ORDER) (?E EXPERT)))
(DEFRELATION WORK-REQUEST-ORIGIN ((?WR WORK-REQUEST) (?T TRIGGERING-EVENT)))
(DEFRELATION RELITED-SENSOR ((?DAS DATA-ACQUISITION-SYSTEM) (?S SENSOR)))
(DEFRELATION EQUIPMENT-SCADA ((?DAS DATA-ACQUISITION-SYSTEM) (?PE PHYSICAL-
EQUIPMENT)))
(DEFRELATION HAS-FUNCTIONNAL-MODEL ((?PE PHYSICAL-EQUIPMENT) (?FEM FUNCTIONNAL-
EQUIPMENT-MODEL)))
(DEFRELATION HAS-FAILURE-MODEL ((?PE PHYSICAL-EQUIPMENT) (?FEM FAILURE-EQUIPMENT-
MODEL)))
(DEFRELATION WORK-REQUEST-SUITED-BY ((?WR WORK-REQUEST) (?E EXPERT)))
(DEFRELATION HAS-LIFE-HISTORY ((?PE PHYSICAL-EQUIPMENT) (?LH LIFE-HISTORY)))
(DEFRELATION CRITICAL-MEASURE ((?M MEASURE)))
(DEFCONCEPT ASSET ((?PE PHYSICAL-EQUIPMENT)))
(DEFCONCEPT NEEDS)
(DEFRELATION PHYSICAL-EQUIPMENT-HAS_NEEDS ((?PE PHYSICAL-EQUIPMENT) (?N NEEDS)))
(DEFRELATION FUNCTIONNAL-EQUIPMENT-MODEL-RELATED-NORMAL-STATE ((?FEM FUNCTIONNAL-
EQUIPMENT-MODEL) (?NS NORMAL-STATE)))
(DEFCONCEPT INPUT)
(DEFCONCEPT OUTPUT)
(DEFRELATION FUNCTIONNAL-EQUIPMENT-MODEL-HAS-INPUT ((?FEM FUNCTIONNAL-EQUIPMENT-
MODEL) (?IN INPUT)))
(DEFRELATION FUNCTIONNAL-EQUIPMENT-MODEL-HAS-OUTPUT ((?FEM FUNCTIONNAL-EQUIPMENT-
MODEL) (?OUT OUTPUT)))
(DEFRELATION FAILURE-EQUIPMENT-MODEL-RELATED-FAILURE-STATE ((?FEM FAILURE-
EQUIPMENT-MODEL) (?FS FAILURE-STATE)))
(DEFRELATION PHYSICAL-EQUIPMENT-SITUATED-IN-PRODUCTION-SITE ((?PE PHYSICAL-
EQUIPMENT) (?PS PRODUCTION-SITE)))
(DEFRELATION SITE-CONTAINS-MATERIAL-RESSOURCE ((?S SITE) (?MR MATERIAL-RESSOURCE)))
```

```
(DEFRELATION ACTOR-IS-HUMAN-RESSOURCE ((?A ACTOR) (?HR HUMAN-RESSOURCE)))
(DEFRELATION LIFE-HISTORY-IS-COMPOSED-BY-STATE ((?LH LIFE-HISTORY) (?S STATE)))
(DEFRELATION LIFE-HISTORY-IS-COMPOSED-BY-INTERVENTION ((?LH LIFE-HISTORY) (?I
INTERVENTION)))
(DEFRELATION LIFE-HISTORY-IS-COMPOSED-BY-MEASURE ((?LH LIFE-HISTORY) (?M MEASURE)))
(DEFRELATION MANAGER-HAS-ACCESS-TO-MAINTENANCE-STRATEGY-MODEL ((?M MANAGER) (?MSM
MAINTENANCE-STRATEGY-MODEL)))
(DEFRELATION WORK-ORDER-DEPENDING-TO-MAINTENANCE-STRATEGY-MODEL ((?WO WORK-ORDER)
(?MSM MAINTENANCE-STRATEGY-MODEL)))
(DEFRELATION MANAGER-MANAGE-CONTRACT ((?M MANAGER) (?C CONTRACT)))
(DEFCONCEPT COMPETENCY)
(DEFRELATION ROLE-DEFINE-COMPETENCY ((?R ROLE) (?C COMPETENCY)))
(DEFRELATION HUMAN-RESSOURCE-HAS-COMPETENCY ((?HR HUMAN-RESSOURCE) (?C
COMPETENCY)))
(DEFCONCEPT ROBOT (?C PHYSICAL-EQUIPMENT))
(DEFCONCEPT ACTIONNEUR (?C PHYSICAL-EQUIPMENT))
(DEFCONCEPT DETECTEUR (?C PHYSICAL-EQUIPMENT))
(DEFCONCEPT CONVOYEUR (?C PHYSICAL-EQUIPMENT))
(DEFCONCEPT CAMERA-DE-SERVEILLANCE (?C ADDITIONAL-COMPONENT))
(DEFCONCEPT ACTIONNEUR-PNEUMATIQUE (?C ACTIONNEUR))
(DEFCONCEPT ACTIONNEUR-ELECTRIQUE (?C ACTIONNEUR))
(DEFCONCEPT DETECTEUR-DE-PRESENCE (?C DETECTEUR))
(DEFCONCEPT DETECTEUR-MAGNETIQUE (?C DETECTEUR))
(DEFCONCEPT COURROIE (?C PHYSICAL-EQUIPMENT))
(DEFRELATION ADDITIONAL-COMPONENT-TRIGGER-EVENT ((?AC ADDITIONAL-COMPONENT) (?TE
TRIGGERING-EVENT)))
(ASSERT (ASSET SISTRE))
(ASSERT (PHYSICAL-EQUIPMENT TEST))
(ASSERT (PHYSICAL-EQUIPMENT-CONSTRUCTOR SISTRE "Bosch"))
(ASSERT (PHYSICAL-EQUIPMENT-CONSTRUCTOR TEST "Siemens"))
(ASSERT (EQUIPMENT-MODEL-GROUP PLATEFORME))
(ASSERT (EQUIPMENT-HAS-TOP-MODEL SISTRE PLATEFORME))
(ASSERT (EQUIPMENT-COMPOSED SISTRE ROBOT))
(ASSERT (EQUIPMENT-COMPOSED SISTRE S1))
(ASSERT (EQUIPMENT-COMPOSED SISTRE S2))
(ASSERT (EQUIPMENT-COMPOSED SISTRE S3))
(ASSERT (EQUIPMENT-COMPOSED SISTRE S4))
(ASSERT (EQUIPMENT-COMPOSED SISTRE S5))
(ASSERT (EQUIPMENT-COMPOSED SISTRE D1))
(ASSERT (EQUIPMENT-COMPOSED SISTRE D2))
(ASSERT (EQUIPMENT-COMPOSED SISTRE D3))
(ASSERT (EQUIPMENT-COMPOSED SISTRE D4))
(ASSERT (EQUIPMENT-COMPOSED SISTRE D5))
(ASSERT (EQUIPMENT-COMPOSED SISTRE D6))
(ASSERT (EQUIPMENT-COMPOSED SISTRE D7))
(ASSERT (EQUIPMENT-COMPOSED SISTRE D8))
```

```
(ASSERT (EQUIPMENT-COMPOSED SISTRE D9))
(ASSERT (EQUIPMENT-COMPOSED SISTRE CONVOYEUR))
(ASSERT (EQUIPMENT-COMPONENT-COMPOSED SISTRE CAMERA-DE-SERVEILLANCE))
(ASSERT (FONCTIONAL-COMPONENT ENTRETOISE))
(ASSERT (EQUIPMENT-COMPONENT-COMPOSED CONVOYEUR ENTRETOISE))
(ASSERT (EQUIPMENT-COMPOSED CONVOYEUR COURROIE))
(ASSERT (ROBOT ROBOT))
(ASSERT (ACTIONNEUR-PNEUMATIQUE POUSSEUR))
(ASSERT (ACTIONNEUR-PNEUMATIQUE TIREUR))
(ASSERT (ACTIONNEUR-PNEUMATIQUE INDEXEUR))
(ASSERT (ACTIONNEUR-ELECTRIQUE S1))
(ASSERT (ACTIONNEUR-ELECTRIQUE S2))
(ASSERT (ACTIONNEUR-ELECTRIQUE S3))
(ASSERT (ACTIONNEUR-ELECTRIQUE S4))
(ASSERT (ACTIONNEUR-ELECTRIQUE S5))
(ASSERT (PHYSICAL-EQUIPMENT ACTIONNEUR))
(ASSERT (PHYSICAL-EQUIPMENT DETECTEUR))
(ASSERT (PHYSICAL-EQUIPMENT CONVOYEUR))
(ASSERT (COMPONENT CAMERA-DE-SERVEILLANCE))
(ASSERT (DETECTEUR-DE-PRESENCE D1))
(ASSERT (DETECTEUR-DE-PRESENCE D2))
(ASSERT (DETECTEUR-DE-PRESENCE D3))
(ASSERT (DETECTEUR-DE-PRESENCE D4))
(ASSERT (DETECTEUR-DE-PRESENCE D5))
(ASSERT (DETECTEUR-DE-PRESENCE D6))
(ASSERT (DETECTEUR-DE-PRESENCE D7))
(ASSERT (DETECTEUR-DE-PRESENCE D8))
(ASSERT (DETECTEUR-DE-PRESENCE D9))
(ASSERT (DETECTEUR-MAGNETIQUE BAL0))
(ASSERT (DETECTEUR-MAGNETIQUE BAL1))
(ASSERT (COURROIE TAP-INT))
(ASSERT (COURROIE TAP-EXT))
(ASSERT (= (FUNCTIONNAL-DEGREE ROBOT) 4))
(ASSERT (TRIGGERING-EVENT PALETTE-DESORDRE))
(ASSERT    (ADDITIONAL-COMPONENT-TRIGGER-EVENT    CAMERA-DE-SERVEILLANCE    PALETTE-
DESORDRE))
(ASSERT (WORK-REQUEST WORK-REQUEST-1))
(ASSERT (WORK-REQUEST-ORIGIN WORK-REQUEST-1 PALETTE-DESORDRE))
(ASSERT (EXPERT PIERRE))
(ASSERT (WORK-REQUEST-SUITED-BY WORK-REQUEST-1 PIERRE))
(ASSERT (WORK-ORDER WORK-ORDER-1))
(ASSERT (WORK-ORDER-CREATED-BY WORK-ORDER-1 PIERRE))
(ASSERT (INTERVENTION INTERVENTION-1))
(ASSERT (HAS-WORK-REQUEST INTERVENTION-1 WORK-REQUEST-1))
(ASSERT (HAS-WORK-ORDER INTERVENTION-1 WORK-ORDER-1))
(ASSERT (OPERATOR PAUL))
```

```
(ASSERT (OPERATOR JACQUE))
(ASSERT (ACTOR ACTOR-1))
(ASSERT (ACTOR ACTOR-2))
(ASSERT (ACTOR-IS-HUMAN-RESSOURCE ACTOR-1 PAUL))
(ASSERT (ACTOR-IS-HUMAN-RESSOURCE ACTOR-2 JACQUE))
(ASSERT (DIAGNOSTIC DIAGNOSTIC-1))
(ASSERT (REPARATION REPARATION-1))
(ASSERT (ACTIVITY-ENSURED-BY DIAGNOSTIC-1 PAUL))
(ASSERT (ACTIVITY-ENSURED-BY REPARATION-1 JACQUE))
(ASSERT (DATA-ACQUISITION-SYSTEM SCADA-1))
(ASSERT (EQUIPMENT-SCADA SCADA-1 SISTRE))
(ASSERT (CRITICAL-MEASURE M-1))
(ASSERT (DATA-ACQUISITION-SYSTEM-CATCH-MEASURE SCADA-1 M-1))
(ASSERT (TRIGGERING-EVENT TEMPERATURE-ELEVEE))
(ASSERT (MEASURE-TRIGGER-EVENT M-1 TEMPERATURE-ELEVEE))
(ASSERT (WORK-REQUEST WORK-REQUEST-2))
(ASSERT (WORK-REQUEST-ORIGIN WORK-REQUEST-2 TEMPERATURE-ELEVEE))
(ASSERT (EXPERT JEAN))
(ASSERT (WORK-REQUEST-SUITED-BY WORK-REQUEST-2 JEAN))
(ASSERT (WORK-ORDER WORK-ORDER-2))
(ASSERT (WORK-ORDER-CREATED-BY WORK-ORDER-2 JEAN))
(ASSERT (INTERVENTION INTERVENTION-2))
(ASSERT (HAS-WORK-REQUEST INTERVENTION-2 WORK-REQUEST-2))
(ASSERT (HAS-WORK-ORDER INTERVENTION-2 WORK-ORDER-2))
(ASSERT (OPERATOR RENE))
(ASSERT (ACTOR ACTOR-3))
(ASSERT (ACTOR-IS-HUMAN-RESSOURCE ACTOR-3 RENE))
(ASSERT (PROGNOSTIC PROGNOSTIC-1))
(ASSERT (ACTIVITY-ENSURED-BY PROGNOSTIC-1 RENE))
```