Pour une perspective comportementale dans les méta-modèles de processus

Sana Damak Mallouli, Saïd Assar, Carine Souveyet

Pour une perspective comportementale dans les méta-modèles de processus

Sana Damak Mallouli*—Saïd Assar— Carine Souveyet ***

** Université Paris 1 Panthéon Sorbonne*
Centre de Recherche en Informatique
90, rue de Tolbiac, F- 75013 Paris cedex
{sana.mallouli, carine.souveyet }@univ-paris1.fr

*** Institut Télécom, Ecole de Management*
Département Systèmes d'Information
9, rue Ch. Fourier, F-91011 Evry Cedex
said.assar@it-sudparis.eu

RÉSUMÉ. *Concevoir des modèles exécutables est l'une des préoccupations majeures de l'ingénierie dirigée par les modèles (IDM). Pour atteindre cet objectif, il est nécessaire de spécifier formellement les propriétés dynamiques d'un modèle selon un méta-modèle, afin de faciliter son exécution par un outil méta-CASE (Ingénierie logicielle assistée par ordinateur). L'objectif de ce papier est de montrer que la spécification des propriétés dynamiques d'un modèle permettant de dériver son outil d'exécution doit couvrir la perspective «comportement», en plus des perspectives «données» et «traitement». C'est pour cette raison qu'une approche de méta-modélisation basé sur un modèle orienté événement offrant ainsi la vision systémique qui permet de gérer l'interaction entre l'outil d'exécution du méta-modèle et de son environnement est proposée. La pertinence et l'applicabilité de la proposition sont illustrées à travers deux exemples: le méta-modèle de workflow et le méta-modèle de la carte.*

ABSTRACT. *Designing executable models is one of the major concerns of model-driven engineering (MDE). To reach this aim, it is necessary to formally specify dynamic properties of a model according to a meta-model, in order to be directly executed by a meta-CASE tool (Computer Aided Software Engineering). Moreover, for some kind of models, it is so important to define a causal relationship in the meta-model specification that offers the possibility to manage the interaction between the execution tool and its environment. For this reason, we propose a meta-modeling approach based on an event model that covers the three-level description: "data", "process" and "behavior". We applied our proposition in the cases of Workflow and Map meta- models to demonstrate its relevance.*

MOTS-CLÉS. *Méta-modélisation, perspective comportementale, exécutabilité des modèles, approche événementielle, vision systémique, outils méta-CASE.*

KEYWORDS: *Meta-modeling, behavioral perspective, model executability, event-based approach, causal relationship, meta-CASE tools.*

1. Introduction

Un méta-modèle est une définition formelle d'un modèle qui aide à le comprendre et qui facilite le raisonnement sur sa structure, sa sémantique et son usage. La méta-modélisation, qui est l'activité de construire des méta-modèles, est très utilisée dans le domaine de l'ingénierie des systèmes d'information et particulièrement dans l'ingénierie des modèles et des méthodes (Rolland, 2005), (Rolland, 2007). Dans l'ingénierie des méthodes, c'est un outil conceptuel indispensable pour définir et raisonner sur de nouveaux modèles. Dans l'ingénierie des outils, les méta-modèles sont une définition formelle nécessaire pour concevoir et construire les outils pour éditer, vérifier, transformer et éventuellement exécuter des instances de ces modèles.

Généralement, la pratique de la méta-modélisation consiste à spécifier des méta-modèles contemplatifs qui reflètent la structure statique des modèles, c.à.d. les concepts et les liens entre ces concepts (Jeusfeld *et al.*, 2009). Selon le formalisme utilisé, un tel méta-modèle peut être complété par la spécification de certaines contraintes. En utilisant UML par exemple pour construire un (méta-) diagramme de classe, il est possible de compléter cette spécification avec des contraintes exprimées avec le langage OCL. Cette vision orientée structure s'apparente à la perspective « données » du cadre de référence des modèles d'ingénierie des SI *(Olle et al. 1989)* (fig. 1). Cependant, on constate que les perspectives « Processus» et « Comportement » sont souvent absentes des pratiques de méta-modélisation dans le domaine de l'ingénierie des SI. Dans la suite du papier, pour éviter la confusion avec les méta-modèles ciblés par ce travail, la perspective « Processus » est nommée perspective « traitement ».

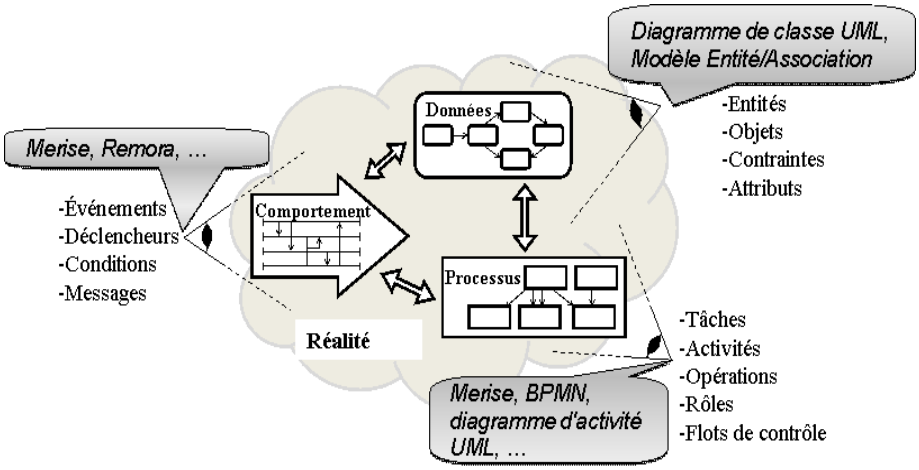

Figure 1. *Les trois perspectives de la modélisation (Olle et al. 1989)*

Selon la nature du modèle représenté, cette absence prive l'ingénieur de méthode et le concepteur d'outils d'une connaissance importante sur les modèles qu'il est en train de manipuler. Dans le cas d'un modèle de processus, les perspectives « traitement» et « comportement» du méta-modèle correspondant renseigneraient sur sa sémantique exécutable. Si l'on considère par exemple le méta-modèle suivant qui représente – en utilisant la notation UML – un extrait du méta-modèle SPEM (fig.2), on constate que ce méta-modèle décrit la dimension structurelle de ce modèle de processus (les concepts et les relations entre eux). La manière avec laquelle ces éléments interagissent lors de l'exécution n'est pas explicitement exprimée dans le méta-modèle.

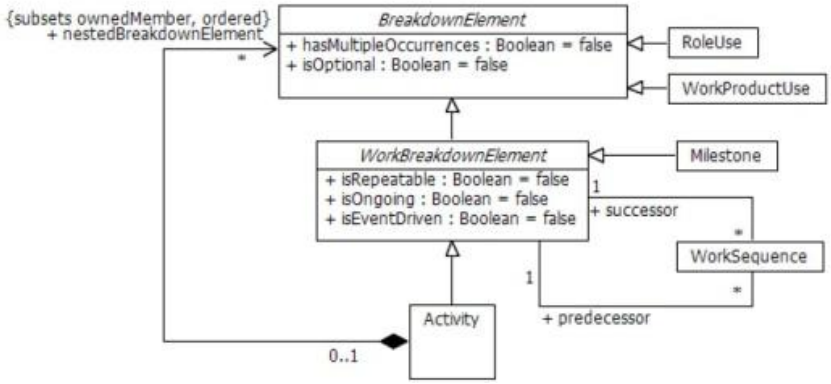

Figure 2. *Extrait du méta-modèle de SPEM (SPEM 2008, p.54)*

L'objectif de cet article est de présenter et de discuter la problématique de la prise en compte des perspectives « traitement » et « comportement » dans la spécification des méta-modèles de processus. Nous nous intéressons particulièrement aux modèles de processus en ingénierie des SI ayant une nature interactive. En effet, ces modèles (tel que BPMN, Workflow, etc.) ont été conçus pour représenter des systèmes organisationnels faisant intervenir des agents externes au système. Pour exprimer cette interaction et toute la sémantique exécutable qui l'entoure, les méta-modèles qui les décrivent doivent prendre en compte non seulement la dimension structure mais aussi la dimension dynamique.

Pour illustrer notre propos, nous allons travailler sur les spécifications de deux modèles de processus: le modèle de Workflow (WfMC, 1995) et le modèle de carte Map (Rolland, 2007). Le premier est un modèle de processus orienté activité très utilisé dans l'industrie, le second est un des modèles de processus intentionnels assez connu dans les milieux académiques et qui a été utilisé dans différents travaux de recherche. Cet article est organisé en 7 sections. La section 2 est un état de l'art sur l'expression de l'exécutabilité dans la méta-modélisation. La section 3 rappelle très brièvement la démarche de méta-modélisation et les formalismes utilisés. La section 4 est la première illustration de notre approche sur le modèle de Workflow. La section 5 est la seconde illustration de l'approche sur le modèle intentionnel de carte.

La section 6 discute les avantages, les inconvénients et les enseignements que l'on peut tirer de l'approche proposée. Enfin, la section 7 est une conclusion qui nous permettra notamment de situer ce travail dans notre projet de recherche en cours concernant le développement d'un environnement outillé pour supporter le formalisme intentionnel de la carte.

2. Etat de l'art

C'est dans le domaine du génie logiciel que la problématique de l'expression de l'exécutabilité dans les méta-modèles a été étudiée en profondeur, grâce notamment à l'essor de l'approche IDM. En effet, étant donné que l'IDM repose fondamentalement sur l'usage extensif de modèles dans toutes les phases de développement d'un logiciel, la question s'est vite posée de comment exécuter un modèle et de comment exprimer la sémantique exécutable de celui-ci. Dans (Breton *et al.,* 2001), une première réflexion sur le lien entre méta-modèle et l'expression de l'exécutabilité des modèles a été faite sur les réseaux de Petri. Les auteurs complètent le méta-modèle statique qui décrit les structures fixes dans un modèle (arcs et transitions dans un réseau de Petri), par un méta-modèle dynamique qui décrit les structures de données nécessaires pendant l'exécution d'une instance de ce modèle (les marquages et les mouvements de jeton). Les auteurs reconnaissent cependant que cet ajout n'est pas suffisant pour exprimer toute l'exécutabilité, car le formalisme utilisé (diagramme de classe UML) n'a lui-même pas de sémantique exécutable, et ils appellent ainsi à la création d'un UML exécutable. Et c'est probablement suite à ces réflexions préliminaires sur l'expression de l'exécutabilité que le langage Kermeta a été développé (Muller *et al.*, 2005). Kermeta apporte un complément aux méta-diagrammes UML sous la forme d'une méta-spécification directement opérationnelle. Combiné avec le principe des méta-classes dynamiques introduit précédemment (Breton *et al.,* 2001), cette approche permet de construire une méta-spécification complète pour un modèle. Le langage Kermeta a pour le moment été utilisé pour construire une description complète et exécutable d'un modèle simple, celui de la machine à états finis (Kermeta, 2011), et expérimenté dans le contexte des systèmes logiciels embarqués pour spécifier les méta-modèles d'UML 2.0 en Kermeta (Koudir *et al.,* 2007).

D'autres travaux dans la communauté génie logiciel et IDM se sont penchés sur les modèles de processus d'ingénierie (appelés *modèles de procédés logiciels*), étant donnée l'importance de décrire, contrôler, et automatiser les procédures avec lesquels les systèmes logiciels sont construits. Un travail important dans ce registre est celui autour d'UML4SPM, qui définit un langage de modélisation de processus d'ingénierie qui est basé sur UML et proche du modèle SPEM de l'OMG (Bendraou *et al.,* 2005) (Bendraou *et al.,* 2007b). Plusieurs expérimentations sont faites pour spécifier la sémantique et exprimer l'exécutabilité de ce langage : d'abord, en utilisant le langage d'exécution de processus métier BPEL (Bendraou *et al.,* 2007a), et ensuite en utilisant le langage de méta-spécification Kermeta (Bendraou *et al.*,

2008). Pour ces deux approches, la problématique de l'interaction avec l'extérieur (l'utilisateur ou d'autres systèmes) est soulignée comme celle qui distingue les deux approches. Malgré plusieurs avantages (notamment l'existence de systèmes techniques fiables pour l'exécution de modèles BPEL), l'usage de BPEL n'est pas jugé satisfaisant à cause de l'absence de concepts pour prendre en compte l'interaction avec l'utilisateur. D'autres recherches, plus orientées vers l'étude comparatives d'approches, complètent ces travaux autour de l'exécutabilité d'un modèle de processus (Combemale *et al.*, 2006a ; Combemale *et al.*, 2006b).

Dans le domaine de l'ingénierie des SI et celui de l'ingénierie des méthodes en particulier, peu de travaux à notre connaissance se sont penchés sur la question de l'expression explicite de l'exécutabilité dans un méta-modèle de processus. La définition d'une méthode étant la combinaison d'un méta-modèle de produit et d'un méta-modèle de processus, la spécification des produits est historiquement la plus ancienne (Harmsen *et al.*, 1996). Concernant le processus, c'est l'approche par assemblage de composants méthodologiques qui est la plus utilisée. Dans (Brinkkemper *et al.*, 2001), le langage MEL[1], un langage formel pour la spécification des méthodes, est proposé. En dehors de la spécification structurelle des composants, l'aspect processus est décrit dans MEL sous forme d'opérateurs formels dont la sémantique est garantie par la notation mathématique sous-jacente. Cette approche par assemblage de composants méthodologique reste actuellement prédominante (Henderson-Sellers *et al.*, 2010), on s'interroge cependant sur les modèles utilisés pour formaliser l'approche, pour spécifier le contenu d'un composant méthodologique, et pour exprimer le processus d'assemblage (Seidita *et al.*, 2007). Enfin, une nouvelle perspective de recherche dans ce sens est celle de définir les composants méthodologiques comme étant des services (Rolland, 2009). Un composant méthodologique est directement exécutable étant donné que c'est un service qui s'écrit avec les standards du SOA, tandis que la composition des services est exprimée informellement à un haut niveau d'abstraction à l'aide de carte intentionnelle Map.

Pour terminer ce tour d'horizon, il faut mentionner les formalismes et langages de méta-modélisation proposés les outils et environnement de type meta-CASE. MetaEdit est un outil connu et populaire (Kelly *et al.*, 1996)., il permet de spécifier un méta-modèle statique avec le formalisme GOPRR[2], et de générer immédiatement un éditeur graphique pour le modèle (MetaCase, 2011). Il est destiné à la création et l'outillage de nouveaux langages spécifique aux domaines (Kelly *et al.*, 2007). Pour ce qui est des perspectives « traitement » et « comportement », elles sont reléguées à la phase de génération de code où un langage de script (appelé *Merle*) permet de parcourir et manipuler les instances du modèle et de générer des instructions dans n'importe quel langage cible (HTML, XML, C++, Java, etc.). La sémantique exécutable s'exprime ainsi par la transformation des structures statiques et non productives du modèle en un ensemble d'instructions (ou de composants) logiciels

[1] MEL : Method Engineering Language
[2] GOPRR : Graph, Object, Property, Relationship, Role

dans un autre langage ou un autre modèle dont l'exécutabilité est connue et supportée par une plate-forme cible. Alors que la définition du méta-modèle se fait d'une manière déclarative avec une interface graphique, l'exécutabilité s'exprime d'une manière opérationnelle avec une interface de type programmation. C'est le principal inconvénient de MetaEdit.

Un autre formalisme de méta-modélisation supporté par un meta-CASE est l'environnement ConceptBase (Jarke *et al.*, 2010) construit autour du modèle et langage Telos (Mylopoulos *et al.*, 1990). Construit avec le langage de logique déclarative Datalog, ConceptBase est un environnement de méta-modélisation extrêmement puissant qui permet de spécifier un nombre quelconque de niveaux d'abstractions et d'exprimer des contraintes et des requêtes sur plusieurs de ces niveaux (et non pas sur un seul niveau comme dans OCL). La saisie d'une spécification se fait textuellement, mais le contenu du référentiel s'affiche graphiquement à la demande de l'utilisateur. Pour ce qui est des perspectives « traitement » et « comportement », ConceptBase a introduit dans ses versions les plus récentes, des règles de la forme Evénement-Condition-Action (ECA) pour exprimer la dynamique d'un méta-modèle. Une illustration est proposée dans (Jeusfeld *et al.*, 2009) avec les règles d'exécution d'un réseau de Petri.

3. A propos des deux exemples

Le but de cet article est de montrer l'importance de l'expression des perspectives « traitement » et « comportement » en l'appliquant dans la méta-modélisation de deux modèles de processus. Nous voulons mettre en évidence à quel point une telle description permettrait d'améliorer la formalisation de la sémantique opérationnelle du modèle et de faciliter par la suite son exécution. Nos exemples sont le modèle de Workflow (WfMC, 1995), et celui des cartes intentionnelles Map (Rolland, 2007).

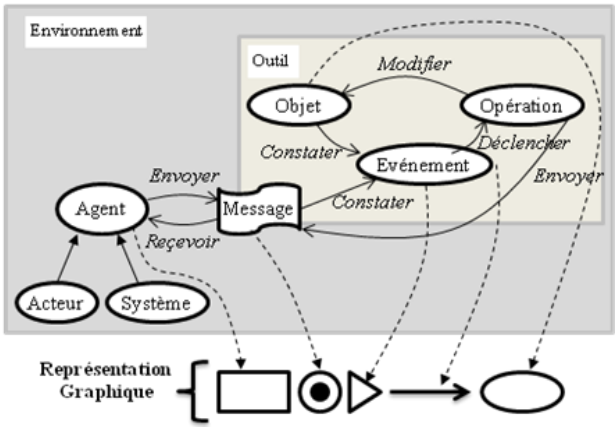

Figure 3. *Représentation graphique de la perspective "comportement"*

Pour mettre en évidence la complémentarité des perspectives de méta-modélisation, nous allons combiner l'usage de 2 techniques. Pour chacun des modèles étudiés, nous appliquons dans un premier temps une méta-modélisation statique à l'aide d'un diagramme de classe UML qui sera complétée d'une part avec des méta-structures nécessaires pour mettre en œuvre l'exécution, et d'autre part, avec la spécification exécutable des méthodes (dans le sens orienté objet) rattachées aux méta-classes. Dans un second temps, à cette spécification est ajouté un schéma Remora (Rolland *et al.*, 1988) qui décrit la dynamique d'interaction et d'exécution du méta-modèle dans la perspective « comportement ».

Remora représente la base d'une méthode événementielle pour la modélisation systémique. On peut noter que Merise/2, ou les méthodes à objets comme OMT ou UML ou O*, reconnaissent implicitement la pertinence de cette orientation et s'en ont inspiré. Les principaux atouts du formalisme Remora sont la simplicité (peu de concepts), l'extensibilité, la représentation graphique de modèles (comme représenté dans la figure 3) et la description formelle. En effet, ce formalisme a été implémenté par l'outil Rubis (Lingat, 1989), ce qui montre qu'il a une sémantique claire et exécutable, ce qui est un critère nécessaire pour la spécification de modèles exécutables.

Le concept d'événement est caractérisé par son nom, son type (interne ou externe), un prédicat exprimant quand il se produit et un corps décrivant un flux d'opérations exécutées lorsque l'événement se déclenche. Un agent est décrit par son nom, son type (humain ou système), un ensemble de messages entrants et un ensemble de messages sortants. La relation « constater » est définie à partir d'un message d'un agent ou une classe d'objet à un événement. La relation « déclencher » concerne un événement à une ou plusieurs opérations encapsulées dans des classes d'objets ou agents.

4. Premier exemple: Le modèle de processus Workflow

Un workflow est défini comme la représentation sous forme de flux d'opérations à réaliser par des agents (WF_Participant) pour accomplir un ensemble de tâches ou d'activités (WF_Activité) regroupées en un même processus métier (WF_Processus). La figure 4 montre la description correspondante du méta-modèle de Workflow. Il s'agit d'un diagramme de classe UML qui décrit les concepts structurels du modèle et auquel a été ajouté d'une part les éléments de méta-information nécessaires à l'exécution (tel que l'attribut « Act_State » dans la classe WF_Activité), et d'autre part, des méthodes qui peuvent être spécifié avec un méta-langage tel que Kermeta. La description de la dynamique du processus avec Kermeta consiste à enchaîner des opérations et à appeler d'autres.

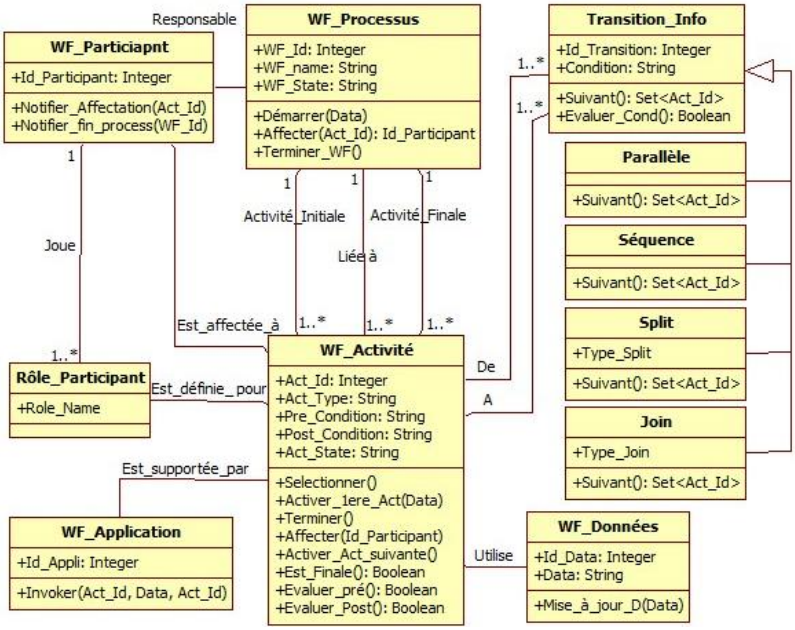

Figure 4. *Les perspectives données et traitement du méta-modèle de Workflow*

La figure 5 représente le méta-modèle de Workflow d'un point de vue comportement. Il s'agit d'une représentation simple qui met en évidence l'ensemble des événements externes et internes possibles lors de l'exécution d'un processus de Workflow ainsi que l'ensemble des opérations qu'ils déclenchent. Il existe deux types d'acteur dans l'outil d'exécution d'un Workflow: WF_Participant et WF_Application. Il est nécessaire de concevoir les interactions entre l'outil d'exécution du workflow et les acteurs. Cette représentation montre que WF_participant est vu comme une entité externe au système et non pas juste comme une classe d'objet.

Le processus de démarrage commence par l'événement extérieur EV1 qui est constatée par l'arrivée du message M1 de la part du participant responsable au déclenchement de cette première opération. Pour illustrer le fait que l'opération «Démarrer» appelle l'opération « Activer_1ere_Act » de l'activité initiale, nous introduisons une flèche en pointillés de l'opération « Démarrer» de la classe d'objet « WF_Activité ».

Dans ce qui suit est une description détaillée de quelques événements.

EV1 : Le démarrage du <WF_Processus> consiste à :

- activer la première <WF_Activité> du <WF_Processus>,
- mettre l'état de cette < WF_Activité > à « activée »,
- affecter cette < WF_Activité > à un <participant>,
- insérer cette < WF_Activité > dans la liste affectées au <WF_Participant>.

EV2 : Lorsque <WF_Activité> passe de « Activée » à « En cour d'exécution »:

- affecter <WF_Activité> à l'agent concerné par son exécution
- notifier l'agent < WF_Participant > par cette affectation

EV3 : Lorsqu'un < WF_Participant > sélectionne une < WF_Activité > à partir de sa liste (pour l'exécuter), le système répond en :

- mettant l'état de l'< WF_Activité > à « en cours d'exécution »,
- mettant à jour la liste des activités affectées au < WF_Participant >,
- mettant à jour la liste des activités en cours d'exécution
- invoquant <WF_Application> à exécuter avec les données (M3).

EV4 : Lorsqu'un <WF_Application> termine l'exécution d'une activité, il faut :

- mettre à jour le produit <WF_Données>
- notifier <WF_Activité> par cette fin d'exécution via le message M4.

EV5 : Lorsque <WF_Activité> passe de « en cours d'exécution » à « terminée », le système aura un traitement différent suivant que < WF_Activité > est:

- la dernière du <WF_Processus>
- la dernière d'une <transition> de type <AND-Join>
- la première d'une <transition> de type <OR-Join>
- suivie d'une transition de type <Séquence> ou <AND-split>

EV6 : Lorsque <WF_Processus> passe de « en cours d'exécution » à « terminée », le système répond en informant le participant avec un message M5.

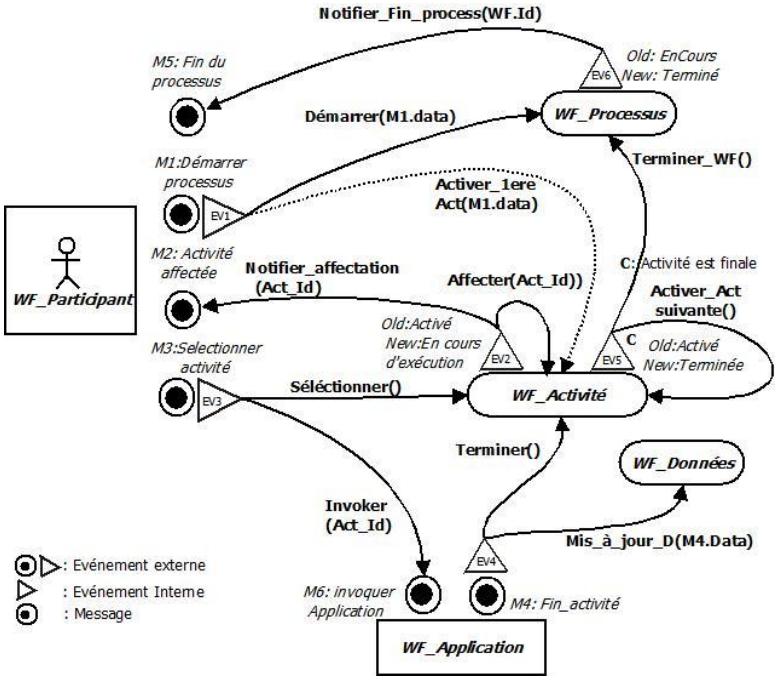

Figure 5. *La perspective « comportement » du méta-modèle de Workflow*

Même si la figure 5 représente un schéma incomplet de la description de la perspective comportementale du Map (par manque d'espace et pour des raisons de simplification), il est important à dire que le formalisme Rémora permet de donner une idée claire sur la dynamique du modèle sur l'échange de messages et les interactions possibles avec l'environnement au moment de l'exécution du modèle.

Comme le montrent les deux tableaux ci-dessous, les événements sont spécifiés par un prédicat. Le prédicat d'un événement extérieur contient une condition sur la validité du message tant disque le prédicat d'un événement interne correspond à un changement d'état d'un objet. Le changement d'état d'un objet peut être exprimé par deux états. L'ancien est adressé avec le mot-clé OLD et le nouvel état est accessible avec le mot-clé NEW.

Table 1. Prédicats des événements internes

EV2: Activation d'une activité : New.Act_State == "Activée"
EV5: Fin d'exécution d'une activité : New. Act _State == "Terminée"
EV6: Fin d'exécution d'un processus : New.WF_State =="Terminé"

Table 2. Prédicats des événements externes

EV1: Démarrer un Processus:(M1.WF_Id).WF_State == "créé"
EV3: Sélectionner une activité: (M3.Act_Id).Act_State == "Activée" and M3.Id_Participant == (M3.Act_Id).atteint
EV4: Fin d'une activité: (M4.Act_Id).Act_State== "Sélectionnée"

En résumé de cet exemple, la perspective « comportement » du méta-modèle de Workflow est importante pour la spécification de sa sémantique opérationnelle parce que les interactions des participants et les applications avec l'outil d'exécution de Workflow sont spécifiées avec une approche événementielle. Nous remarquons que certains concepts du méta-modèle de Workflow sont considérés comme des classes d'objets dans la perspective « donnée », et aussi comme des agents dans la perspective « comportement ». C'est là où l'on perçoit la pertinence de cette perspective qui donne une sémantique aux concepts. Par exemple le « WF_Participant » est vu à la fois en tant qu'informations (ou données) sur les acteurs, mais aussi en tant qu'entité externe au système.

5. Second exemple: le modèle de la carte Map

Le modèle de la carte, appelé aussi Map, est un modèle de processus orienté intention qui n'est pas directement exécutable. Un Map est composé d'un ordonnancement non figé d'intentions et de stratégies. Une stratégie est une manière de réaliser une intention. Elle peut être une action exécutée par une application externe qui aboutit à des transformations du produit. Une section est composée

d'une intention source, une intention cible et une stratégie. Les cartes ne contraignent pas l'utilisateur dans une démarche constituée d'étapes successives mais permettent au contraire, un grand degré de liberté dans l'ordonnancement de réalisation des intentions et dans le choix des techniques qu'il désire appliquer. C'est sans doute ce critère qui fait que jusqu'à maintenant l'exécution du modèle de la carte reste un défit à soulever.

Le diagramme de classes UML de la figure 6 décrit la structure statique du méta-modèle de la carte.

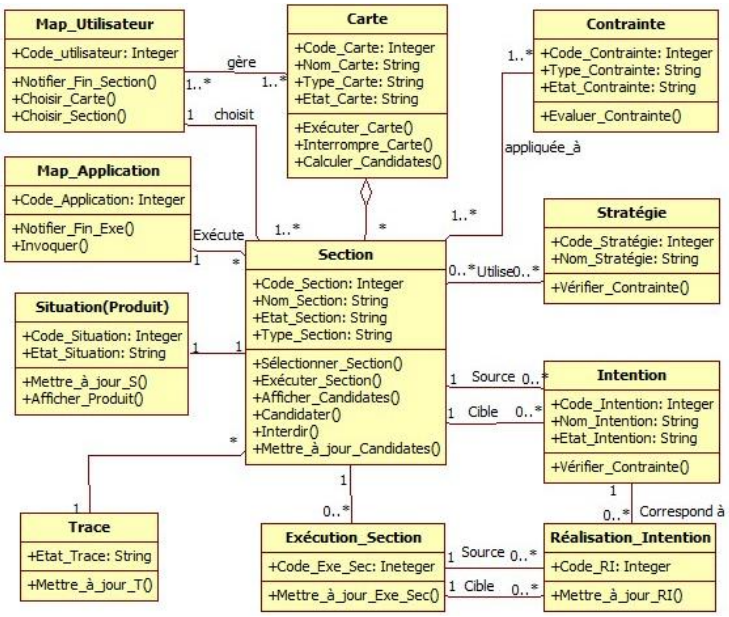

Figure 6. *Schéma statique du méta-modèle de la Carte*

L'intégration de la dynamique dans cette description se traduit dans cette étape par l'ajout d'un ensemble de classes et de nouveaux attributs. En effet, à part les classes de base du méta modèle de la carte qui sont : *Carte*, *Section*, *Intention*, *Stratégie* et *Contrainte*, nous avons ajouté les classes *Trace*, *Réalisation_Intention*, *Exécution_Section* et *Situation*. Nous avons aussi définit des attributs « état » dans plusieurs classes. Par exemple, Etat_Section prend les valeurs {Sélectionnée, Exécutée, Candidate, Interdite, En cours d'exécution}. Cette information indique sur l'évolution du statut de l'objet *Section* lors de l'exécution du processus Map et joue ainsi un rôle important pour raisonner sur l'avancement de cette exécution.

Les classes *Map_Utilisateur* et *Map_Application* correspondent à des agents externes qui interagissent avec le modèle de la carte, leurs présences dans le diagramme de classe permettent d'avoir des informations statiques sur les utilisateurs du système. La perspective « traitement » représente la sémantique opérationnelle d'un modèle. On peut à ce niveau utiliser un langage tel que Kermeta

pour spécifier en détail la sémantique opérationnelle au moyen d'expressions rattachées au corps des opérations des classes UML.

Cette vision procédurale (algorithmique) est insuffisante pour les modèles d'applications interactives - tel que le modèle de la carte - qui nécessitent la prise en compte de l'interaction avec des acteurs externes.

La figure 7 correspond au schéma en Remora du méta-modèle de la carte qui intègre la perspective comportementale à la spécification du modèle.

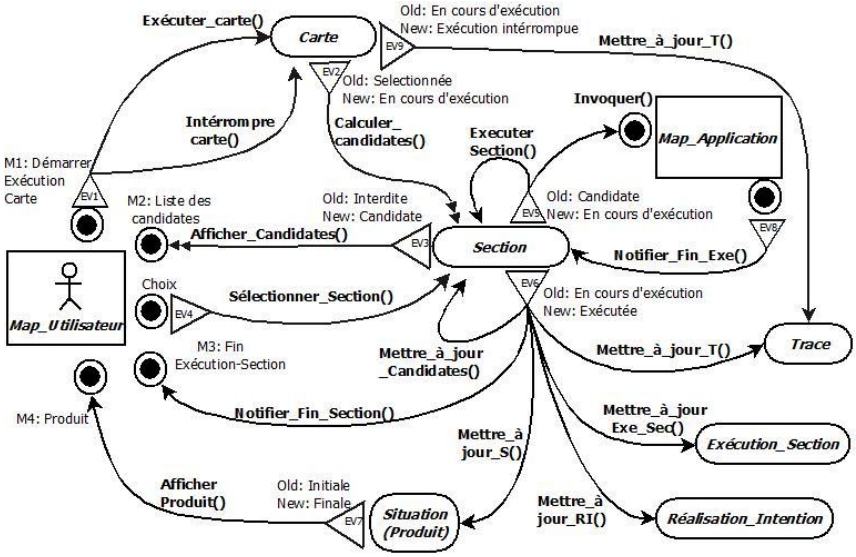

Figure 7. *Représentation de la dynamique du méta-modèle de la carte avec Remora*

Il s'agit d'une représentation graphique qui traduit la vision systémique au moment de l'exécution du modèle de la carte. Pour des raisons de gestion d'espace et de clarté du schéma, nous n'avons pas représenté les paramètres des méthodes ni la totalité des objets du méta-modèle.

Les événements internes sont des changements d'états d'objets. Tandis que les événements externes représentent des stimuli en provenance d'un utilisateur ou d'un système externe. Dans ce qui suit, une description textuelle de quelques événements et des opérations qu'ils déclenchent :

EV1 : L'utilisateur sélectionne une carte et demande son exécution

- déclenche l'exécution de la carte

EV2 : Le changement de la valeur d'« Etat_Carte » de '*Sélectionnée*' à '*En cours d'exécution*'

- cet événement déclenche le calcul des sections candidates

EV3 : Le changement de Etat_Section de '*Interdite*' à '*Candidate*'

- cet événement déclenche l'affichage des sections candidates à l'utilisateur

EV4 : L'utilisateur choisit une section parmi les sections candidates

- cet événement modifie Etat_Section de ‘Candidate’ à ‘Sélectionnée’

EV5 : Le changement d’Etat_Section de ‘*Candidate*’ à ‘*En cours d’exécution*’

- exécute la section sélectionnée: cette opération appelle des méthodes telles que la gestion des contraintes et la gestion de l’affinement comme elle peut invoquer une application externe pour exécuter la tâche.

EV6 : Une fois l’exécution d'une section est terminée, ‘*Etat_Section*’ passe de ‘*En cours d’exécution*’ à ‘*Executée*’.

- Ce changement d’état déclenche les opérations de mise à jour de la trace, du produit, de la liste des candidates, des classes Exécution-Section et Réalisation_Intention, ainsi que celle d’informer l’utilisateur de la fin d’exécution de la section qu’il a sélectionnée.

La perspective comportementale s'exprime par cette description graphique et textuelle au niveau du méta modèle de la carte, et elle représente la spécification conceptuelle de l'exécutabilité de la carte Map.

6. Discussion

Les deux exemples de méta-modélisation montrent que la spécification de l’exécutabilité du méta-modèle de Workflow et de Carte intentionnelle intègrent (1) la spécification de la structure statique des instances du modèle, (2) les traitements qui sont dirigés par cette structure et qui font évoluer cette structure et enfin (3) les règles événementielles permettant de spécifier les interactions entre l’outil d’exécution et les agents constituants son environnement. En effet, le flux d’exécution dirigé par le modèle nécessite :

- la prise en compte d’interactions avec des utilisateurs ou
- la délégation asynchrone à des sous-systèmes.

L’objectif de ce papier est d’illustrer que le troisième point relatif à la perspective « comportement » avec sa vision événementielle est indispensable à la complétude de la spécification de l’exécution dirigée par le modèle. En effet, cette perspective permet de connaître les éléments-type constituant l’environnement de l’outil d’exécution et comment ces éléments agissent ou/et interagissent sur le flux d’exécution. La stratégie adoptée par la proposition combine les avantages des approches déclaratives et impératives. En effet, la perspective « traitement » est spécifiée selon une approche impérative et facilement exécutable alors que la perspective « comportement » avec sa vision événementielle permet de décrire le flux d’exécution selon une approche déclarative qui est néanmoins facilement exécutable sur des systèmes événementiels interactifs.

Le modèle Remora connu dans l’ingénierie des SI est ici, utilisée pour spécifier l’exécutabilité d’un modèle de processus. Ce modèle permet de construire des méta-modèles de la dynamique et du comportement qui, d’une part offrent une représentation graphique lisible et précise, et d’autre part, disposent d’une sémantique claire et bien définie qui prend en compte la nature indéterministe de

l'exécution d'un processus tel qu'une carte Map. Par rapport à une spécification de type méta-programmation (tel que Kermeta) ou purement déclarative (tel qu'avec ConceptBase), la représentation graphique apporte un complément indéniable dans la mise en évidence des points d'interaction entre le processus qui s'exécute et son environnement.

7. Conclusion

Nous avons dans cet article abordé le problème de l'usage des trois perspectives de la modélisation dans la spécification d'un méta-modèle. En utilisant un modèle de représentation de la dynamique qui allie une sémantique comportementale rigoureuse et une représentation graphique adéquate, nous avons montré l'apport de cette spécification dans l'expression de l'exécutabilité d'un modèle de processus. Nous avons appliqué cette approche sur un modèle intentionnel de processus, dont la nature indéterministe rend difficile toute formalisation de la sémantique.

En analysant l'état de l'art, on constate que la communauté du génie logiciel et celle de l'IDM en particulier se sont penchées sur ce problème, et que les solutions proposées reposent sur le modèle UML et le modèle générique MOF, et s'inspirent des travaux autour de l'exécution du méta-modèle de processus SPEM. Dans le domaine de l'ingénierie des méthodes, les approches sont différentes et c'est l'assemblage de composants qui est privilégié pour définir de nouvelles méthodes. Cependant, on constate que la problématique de l'exécutabilité est commune dans le sens où un composant méthodologique de processus doit être spécifié en détail et que sa sémantique exécutable doit être explicitée.

Le travail présenté ici fait partie d'une recherche autour de la conception et la spécification d'outils logiciels de type CASE pour le modèle de carte Map (Mallouli, 2007; Souveyet *et al.*, 2007). Nous sommes actuellement en train d'étudier, d'expérimenter et de comparer divers environnements de méta-modélisation (Kermeta, MetaEdit, ConceptBase) dans le but d'évaluer la pertinence des approches de construction d'outils logiciels pour un modèle intentionnel de processus. Le travail présenté ici suggère que les approches de méta-modélisation sont multiples et complémentaires, mais aussi incapables de prendre en compte tous les besoins du concepteur en termes de représentation graphique, de prise en compte de l'interactivité des modèles, et de l'expression formelle et détaillé de l'exécutabilité. Et c'est dans cette perspective que nous envisageons dans un travail futur la proposition d'un langage et d'un environnement de méta-modélisation qui prend en compte tous ces besoins.

8. Bibliographie

Bendraou R., Gervais M-P., Blanc X., « UML4SPM: A UML2.0-Based metamodel for Software Process Modeling », Proceedings of the ACM/IEEE 8th International

Conference on Model Driven Engineering Languages and Systems (MoDELS'05), Montego Bay, Jamaica, LNCS, Vol. 3713, pp. 17-38, Octobre 2005.

Bendraou R., Sadovykh A., Gervais M.P., Blanc X., « Software Process Modeling and Execution: The UML4SPM to WS-BPEL Approach », Proceedings of the 33rd EUROMICRO Conference of Software Engineering Advanced Application (SEAA), pp. 314-321, Lübeck, Germany, IEEE Computer Society, 28-31 août 2007.

Bendraou R., Combemale B., Crégut X., Gervais M-P., « Definition of an Executable SPEM 2.0 » Proceedings of the 14th Asia-Pacific Software Engineering Conference (APSEC '07), IEEE Computer Society, 5-7 décembre 2007.

Bendraou R., Gervais M-P., Blanc X., Jézéquel J-M., « Vers l'Exécutabilité des Modèles de Procédés Logiciels », actes conférence LMO (Langages et Modèles à Objets), Montréal, Canada, mars 2008.

Breton E., Bézivin J., « Towards an understanding of model executability », Proceedings of the International Conference on Formal Ontology in Information Systems - Volume 2001, Ogunquit, Maine, octobre 2001.

Brinkkemper S., Saeki M., Harmsen F., « », Proceedings of 13th Int. Conf. on Advanced Information Systems Engineering (CAiSE'01), LNCS, Volume 2068/2001, pp. 473-476, 2001.

Combemale B., Rougemaille S., Crégut X., Migeon F., Pantel M., Maurel C., « Expériences pour décrire la sémantique en ingénierie des modèles », 2ème journée IDM, juin 2006a.

Combemale B., Rougemaille S., Crégut X., Migeon F., Pantel M., Maurel C., Coulette B., « Towards Rigorous Metamodeling », 2nd International Workshop on Model-Driven Enterprise Information Systems (MDEIS), Paphos, Chypre, mai 2006b.

Harmsen F., Saeki M., « Comparison of four method engineering languages », IFIP 8.1 Conference on Method Engineering, Chapman and Hall, pp. 209-231, 1996.

Henderson-Sellers B., Ralyté J., « Situational Method Engineering: State-of-the-Art Review » Journal of Universal Computer Sciences, vol.16, n°3, pp. 424-478, 2010.

Jarke M., Jeusfled M.A., Nissen H.W., Quix C., Staudt M., « Metamodelling with Datalog and Classes: ConceptBase at the Age of 21 » 2nd Int. Conf. Object Oriented Data Bases (ICOODB'09), LNCS 5936, pp.95-112, Springer, 2010.

Jeusfled M.A., Jarke M., Mylopoulos J., (Eds), *Metamodeling for Method Engineering*, The MIT Press, 2009.

Kelly, S., Lyytinen, K., Rossi, M., « MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE Environment », Proceedings of 8th Int. Conf. on Advanced Information Systems Engineering (CAiSE'96), LNCS 1080, Springer-Verlag, pp. 1–21, 1996.

Kelly S., Tolvanen J., *Domain-Specific Modeling: Enabling Full Code Generation*, Wiley & Sons, New Jersey, 2008.

Koudri A., Champeau J., Aulagnier D., « Une sémantique opérationnelle pour une meilleure méta-modélisation », Atelier sur la Sémantique des Modèles (SéMo'07), Toulouse, 29 - 30 mars 2007.

Kermeta, http://www.kermeta.org, 2011.

Lingat J.Y., « Rubis : un système pour la spécification et le prototypage d'applications Bases de Données », Thèse de doctorat de l'Université P&M Curie, Juillet 1988.

Mallouli S, « Proposition d'un environnement de développement de systèmes d'information orienté modèle basé sur la technologie XML », Mémoire de Mastèr Systèmes d'Information et Nouvelles Technologies, Université de Sfax – Centre de Recherche en Informatique, septembre 2007.

MetaCase, http://www.metacase.com/, 2011.

Muller P., Fleurey F., Jézéquel J., « Weaving Executability into Object-Oriented Meta-Languages », Proceedings of the ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS'05), Montego Bay, Jamaica, Oct. 2005, LNCS, Vol. 3713, pp. 264-278.

Mylopoulos J., Borgida A., Matthias J., Koubarakis M., « Telos: representing knowledge about information systems », ACM Transactions on Information Systems, vol. 8, Issue 4, Octobre 1990.

Olle T.W., Hagelstein J., MacDonald I.G., Rolland C., Sol V, van Assche F.J.M., *Information Systems Methodologies: A Framework for Understanding,* Addison-Wesley, 1991.

Rolland C., Foucaut O., Benci G., *Conception des systèmes d'information: la méthode REMORA*, Eyrolles, 1988.

Rolland C., « L'ingénierie des méthodes : une visite guidée », revue e-TI, n°1, 2005, disponible sur http://www.revue-eti.net/document.php?id=726

Rolland C., « Capturing System Intentionality with Maps », dans Krogstie J.; Opdahl A., Brinkkemper S., *Conceptual Modelling in Information Systems Engineering*, Springer-Verlag, Berlin Heidelberg, Germany, 2007, pp. 141 - 158.

Rolland C., « Method Engineering : Trends and Challenges », Conférence invité à la WG8.1 Working Conference Situational Method Engineering, 12-14 septembre 2007, Genève, Suisse.

Rolland C., « Method engineering: towards methods as services », Software Process: Improvement and Practice, vol. 14, n°3, pp. 143-164, 2009.

Seidita V., Ralyté J., Henderson-Sellers B., Cossentino M., Arni-Bloch N., « A comparison of deontic matrices, maps and activity diagrams for the construction of situational methods », CAiSE Forum, 19th Int. Conf. on Advanced Information Systems Engineering, Trondheim, Norway, 11-15 June, 2007.

Souveyet C., Assar S., « Exécutabilité des modèles : réflexions et expériences », Actes de l'atelier MADSI, INFORSID, mai 2007, Perros-Guirec - France.

SPEM. *Software & Systems Process Engineering Meta-Model Specification*. OMG document formal/2008-04-01, http://www.omg.org/spec/SPEM/2.0/PDF, avril 2008.

WfMC, The Workflow Reference Model, Document Number TC00-1003, 1995, http://www.wfmc.org/standards/docs/tc003v11.pdf