# TopPROCESS : vers une ingénierie des procédés dirigée par les modèles

**Mots clés**

Ingénierie des procédés, Ingénierie dirigée par les modèles, SPEM (*Software & Systems Process Engineering Metamodel*), Méta-modélisation, Transformations de modèle, Vérification de modèle

Angel GARCIA[1], Benoît COMBEMALE[2], Xavier CREGUT[3], Jérôme VANDEUR[1]
*Tectosages – Verniolle[1], AtlanMod (INRIA & EMN)[2], IRIT, Université de Toulouse[3]*

Si la maîtrise du processus de développement est un critère important de la réussite d'un projet, il n'est encore que très partiellement formalisé à des fins de construction d'un outillage adapté. Nous présentons l'approche TopPROCESS qui vise à combler ce manque. Nous expliquons comment l'ingénierie dirigée par les modèles permet de faciliter la modélisation des processus et le développement des outils associés.



## 1. Introduction

La notion de modèle est essentielle dans toutes les disciplines techniques et scientifiques. Elle permet par exemple de raisonner sur un système qui n'existe pas encore, d'en vérifier les propriétés, de le valider, de prédire son comportement, ses performances, de planifier son développement ou son industrialisation, de réduire les coûts de sa conception et réalisation, etc. Ces modèles sont maintenant manipulés au moyen d'outils informatiques. Ainsi, le Falcon 7X de Dassault Aviation est le premier avion à avoir été développé entièrement dans une filière numérique. Ceci a permis d'améliorer la qualité de conception au point de réduire la durée d'assemblage de moitié et de limiter les essais en vol aux phases les plus critiques.

Si la notion de modèle est présente en informatique depuis longtemps, elle prend depuis quelques années une importance particulière au travers de l'ingénierie dirigée par les modèles (IDM) qui vient compléter les technologies à objets et à composants dans le développement et la maintenance des systèmes informatiques. Ainsi, pour faire face à la montée en complexité des systèmes à construire, les notions de modèle et métamodèle sont aujourd'hui au cœur des nouveaux systèmes. Ces modèles (semi-)formels portent dans leur sillage un ensemble de méthodes et technologies : métamodélisation, transformation, vérification ou simulation.

Dans cette nouvelle vision, le code exécutable ne représente plus le référentiel unique dans le cycle de développement. De nombreux modèles (métier, de test,

**L'ESSENTIEL**

Cet article présente TopPROCESS, une approche proactive de pilotage des processus qui permet de mettre en place les bonnes pratiques (ITIL, CMMI...) de manière pragmatique et opérationnelle sur une architecture modulaire. L'approche est globale, transverse et fait intervenir de manière collaborative les parties prenantes en concentrant les efforts sur les processus apportant de la valeur. TopPROCESS a permis la conception et la modélisation des processus du projet Topcased dont l'objectif est d'offrir un environnement *open source* pour le développement de systèmes critiques. L'ingénierie des modèles (IDM) a été utilisée pour le développement d'un éditeur de modèles de processus et la vérification de leur cohérence par rapport à des contraintes exprimées en termes de ressources, ordonnancement, temps...

**SYNOPSIS**

This article presents TopPROCESS, a proactive approach to manage processes. Based on a modular architecture, TopPROCESS favors the integration of best practices (e.g., ITIL, CMMI, etc.) in a pragmatic and operational way. The approach is global and transverse. It involves stakeholders in a collaborative fashion and focuses on value added by the processes. TopPROCESS has been used to design and model the processes used in the Topcased project which aims at providing an open-source environment for critical systems development. Model Driven Engineering (MDE) was used to define a graphical editor for process models and also to implement a verification tool – based on model-checking techniques – in order to check whether a process is able to finish or not, according to scheduling, time and resources constraints.

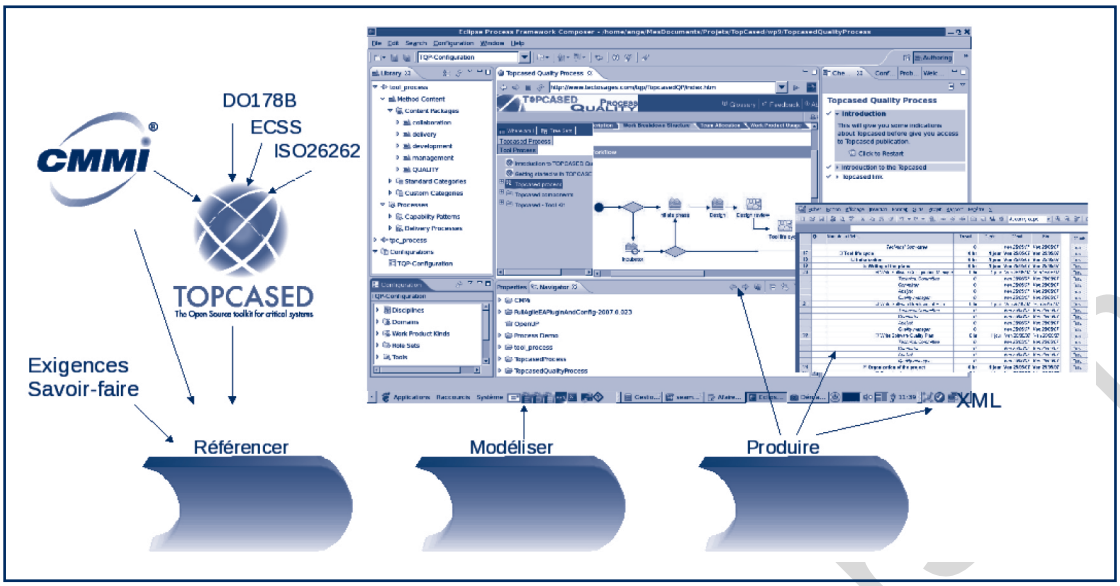


*Figure 1. Les trois principales étapes de la démarche TopPROCESS.*

d'architecture, de déploiement, de processus, etc.) sont développés, documentés et maintenus en dehors du code et des documentations textuelles habituelles. Le modèle n'est plus un élément de documentation mais l'élément de base de la production des systèmes. Cette approche a déjà connu de nombreux succès [3, 5].

Par ailleurs, il est depuis longtemps reconnu qu'un logiciel est un produit manufacturé complexe dont la réalisation doit s'intégrer dans une démarche méthodologique : *le procédé de développement*[1]. Pour améliorer la qualité du produit développé, il est important de maîtriser son procédé de développement. En particulier, il ne s'agit pas de faire un audit ponctuel mais de fournir à une entreprise les moyens techniques et organisationnels de maîtriser ses processus et d'améliorer sa production. L'industrialisation du développement doit devenir une réalité pour permettre : un gain de productivité dès la conception des systèmes, la maîtrise des processus du système d'information et des logiciels, une communication ciblée, claire et compréhensible par tous, la réduction du coût de maintenance et d'évolution du système d'information, ainsi que la capitalisation, la diffusion et la réutilisation du savoir-faire. Autant d'éléments moteurs de la compétitivité et de l'innovation.

Cependant, modéliser et exploiter le procédé de développement est une tâche ardue qui s'apparente au développement de logiciel [4]. Aussi, il paraît naturel d'appliquer aux procédés les techniques du génie logiciel, en particulier l'IDM.

Dans cet article, nous présentons, section 2, une approche pour la modélisation, la capitalisation et l'industrialisation des processus de développement et son application pour définir le processus qualité du projet Topcased[2]. Dans la section 3, nous expliquons comment l'IDM a facilité la mise en œuvre en facilitant le développement de l'outillage associé. La section 4 conclut et présente quelques perspectives.

## 2. L'approche TopPROCESS

TopPROCESS est une approche proactive de pilotage des processus qui permet de mettre en place et de capitaliser les bonnes pratiques de manière pragmatique et opérationnelle. L'objectif est de fournir aux dirigeants des entreprises les moyens efficaces de conception, de pilotage et de maîtrise de leurs processus, tout en se basant sur leur cœur de métier et les activités engendrant de la valeur pour leurs entreprises. TopPROCESS s'appuie sur une modélisation, une formalisation des processus dans le but de :

- faciliter la communication entre les acteurs du projet grâce à un formalisme standard, partagé et compris de tous, évitant ainsi les approximations et incompréhensions ;
- permettre un pilotage opérationnel des projets : managers et collaborateurs ont une vision claire et partagée d'un projet. Les collaborateurs connaissent les tâches à réaliser et rendent compte de leur avancement. Les managers ont alors une bonne visibilité sur le projet et peuvent le piloter proactivement pour (ré)agir au plus tôt.

### 2.1. Démarche TopPROCESS

L'approche TopPROCESS s'appuie sur une architecture modulaire et s'organise selon une démarche en trois étapes principales : référencement, modélisation et production (figure 1).

[1] Nous emploierons indifféremment les termes de *procédé* et *processus*.
[2] *Toolkit in Open Source for Critical Applications & Systems Development*, http://www.topcased.org/

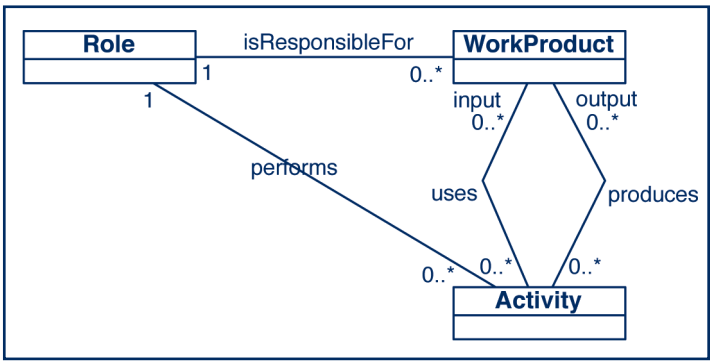


*Figure 2. Métamodèle conceptuel de SPEM.*

Le **référencement** consiste à capitaliser l'expérience et les documents de travail sous forme de modèles informatiques structurés. Ainsi, du plus général au plus détaillé, les référentiels de domaine comme CMMI (*Capability Model Management*), les référentiels métiers (par exemple, la DO178B pour l'aéronautique), les exigences émises par d'autres disciplines (qualité, gestion de projet...) et les savoir-faire sont transformés en composants de processus qui pourront ensuite être réutilisés et adaptés dans les étapes suivantes de TopPROCESS.

La **modélisation** consiste à construire les processus en s'appuyant sur les composants de processus élaborés pendant le référencement. Ces processus pourront être raffinés pour les rendre de plus en plus précis et les adapter à des besoins particuliers. On peut ainsi modéliser d'abord des macro-processus, processus structurants de haut niveau, qui seront ensuite affinés pour un domaine ou un métier spécifique, puis pour un type de projet donné.

La **production** consiste à s'appuyer sur les processus décrits dans la phase de modélisation pour construire des outils d'aide à la *production* des systèmes et aux équipes de développement. Les processus s'intègrent ainsi au cœur du développement, explicitent les contraintes, et permettent aux équipes de se concentrer sur leurs activités. Les modèles de processus deviennent aussi des outils d'aide au pilotage des projets en engendrant automatiquement des modèles de planning (par exemple, pour MS-Project).

### 2.2. SPEM : modélisation des processus selon l'OMG

TopPROCESS utilise le standard SPEM (*Software & Systems Process Engineering Metamodel* [7]) proposé par l'OMG[3] pour la modélisation des procédés logiciels et systèmes. SPEM vise à offrir un cadre conceptuel pour modéliser, échanger, documenter, gérer et présenter les processus et méthodes de développement. Ses concepts sont décrits par un métamodèle qui repose sur l'idée qu'un procédé de développement est une collaboration entre des entités actives et abstraites : les *rôles*, qui abstraient des compétences et effectuent des opérations, les *activités*, sur des entités concrètes et réelles, et les *produits* (figure 2).

La version 2 de SPEM propose une évolution intéressante en séparant les aspects contenus relatifs aux méthodologies de développement de leurs possibles instanciations dans un projet particulier. Ainsi, pour pleinement exploiter ce standard, la première étape devrait être de définir toutes les phases, activités, produits, rôles, guides, outils, etc., qui peuvent composer une méthode pour ensuite, dans une deuxième étape, choisir en fonction du contexte et du projet, le contenu de la méthode appropriée pour l'utiliser dans la définition du procédé. On retrouve nos étapes référencement et modélisation.

### 2.3. Le processus qualité de Topcased : une application de TopPROCESS

Topcased est un projet du pôle de compétitivité Aerospace Valley. Il vise à définir un atelier *open source* pour le développement de systèmes et applications critiques. Pour que l'intégration d'un nouveau *plugin* à l'atelier soit acceptée, il est nécessaire que ses auteurs respectent le processus de développement logiciel de Topcased, TQP (*Topcased Quality Process*). Ce processus a pour objectif de concilier développement *open source* et exigences qualité liées aux standards de l'aéronautique (DO178B), du spatial (ECSS-E40) et de l'automobile (ISO26262). Il est composé du processus global d'intégration et du processus auquel doivent répondre les contributeurs de l'atelier Topcased.

Pour construire ce processus, nous nous sommes appuyés sur l'approche TopPROCESS et avons utilisé l'outil développé par le projet Eclipse EPF[4]. EPF s'appuie sur un métamodèle compatible avec SPEM et permet de décrire les trois parties d'un processus présentes dans notre approche : la définition du contenu du processus (*référencement*), la définition du processus (*modélisation*) et la création des vues (*production*).

#### *2.3.1. Référencement*

Définir le contenu consiste à définir tous les éléments qui vont ensuite pouvoir être échangés, partagés ou réutilisés par d'autres départements de l'entreprise, des filiales

[3] *The Object Management Group*, http://www.omg.org/

[4] Eclipse Process Framework, http://www.eclipse.org/epf/

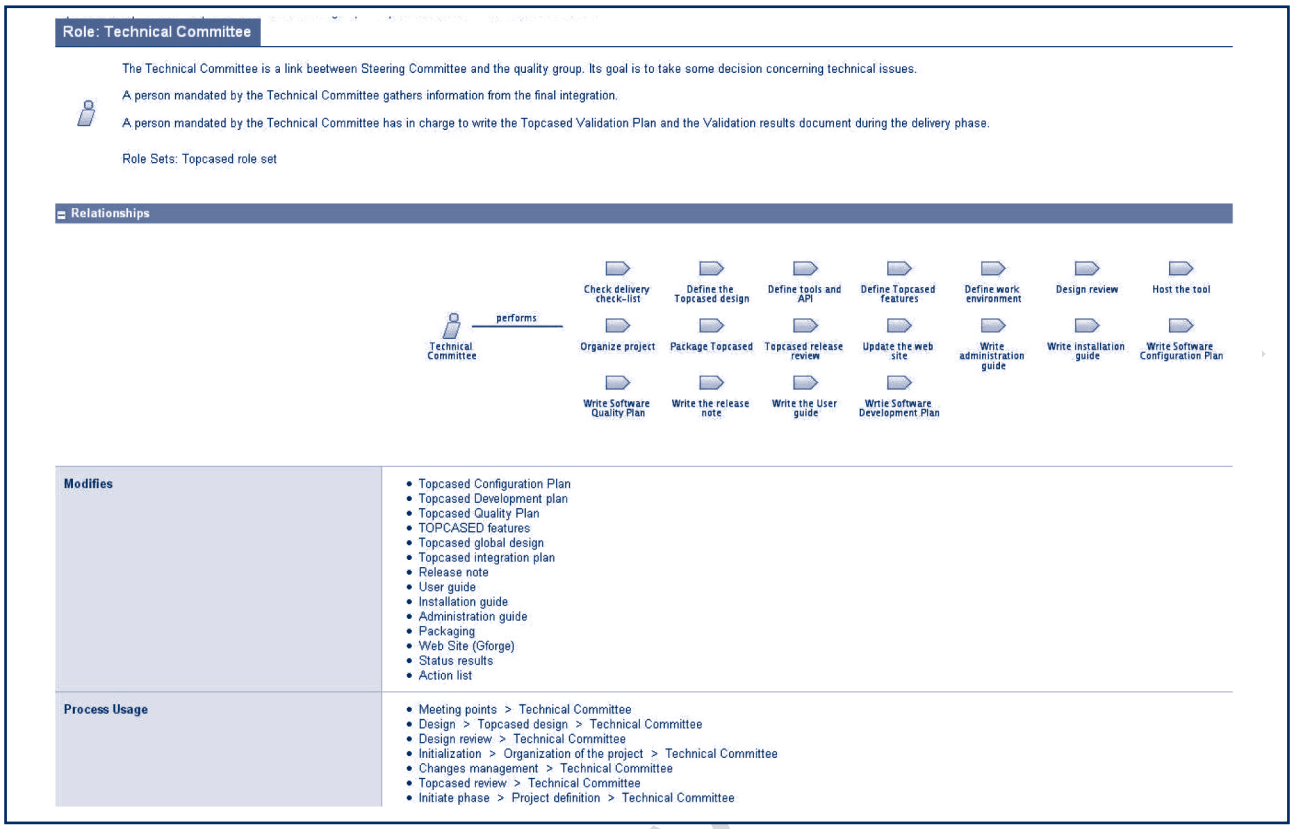


*Figure 3. Description du rôle « comité technique ».*

ou des partenaires. On peut ainsi capitaliser les bonnes pratiques pour ensuite les réutiliser. Dans Topcased, les exigences de l'assurance qualité sont relatives aux standards métiers. Celles-ci sont consignées dans le plan qualité et le kit d'aide au développement. Les exigences proviennent des disciplines comme la gestion de configuration, des changements, des documents, etc. Prenons comme exemple une exigence de la gestion de changement :

> « TPC_QKIT_CHG1 — The changes management of the project is described in the Software Configuration Plan (SCP) of the tool. See the template in §5.2.3 for all the mandatory information. »

Nous l'avons transformée en un composant de processus. Il contient une activité, « réalisation du plan de gestion de configuration » dont le produit attendu est le « plan de gestion de configuration ». L'exigence mentionne l'existence d'un *template* qui devient le *template* du produit de sortie attendu. L'exigence ne précise rien sur le rôle devant intervenir.

### 2.3.2. Modélisation

Pour la modélisation d'un processus, nous avons réalisé une extension de ce composant pour choisir le rôle qui intervient et est responsable de cette activité. Réutiliser le composant garantit le respect des exigences attendues et accélère la construction du processus.

La modélisation du processus nous a conduit à distinguer deux cycles : le premier s'adresse à un contributeur de l'atelier, le second concerne l'équipe Topcased chargée de l'intégration et de la livraison finale.

### 2.3.3. Production

La production a consisté à créer des vues sur le processus avec deux objectifs : communiquer efficacement les façons de travailler et fournir aux équipes projet des modèles prêts à l'emploi. Dans TQP, nous avons créé une vue dédiée aux équipes souhaitant développer un plugin. Très souvent une publication ciblée sera plus productive car plus pertinente qu'une publication globale. Cette publication extrait automatiquement du modèle les informations relatives à un élément, par exemple pour visualiser un rôle et toutes ses activités. La figure 3 présente le rôle « comité technique ».

## 3. Production d'outillage grâce à l'IDM

Montrons maintenant comment l'IDM nous a aidés à développer un éditeur graphique de modèles de processus et un outil d'analyse de faisabilité d'un processus. Une étape préalable dans un contexte IDM est la définition du modèle des données manipulées, le métamodèle.

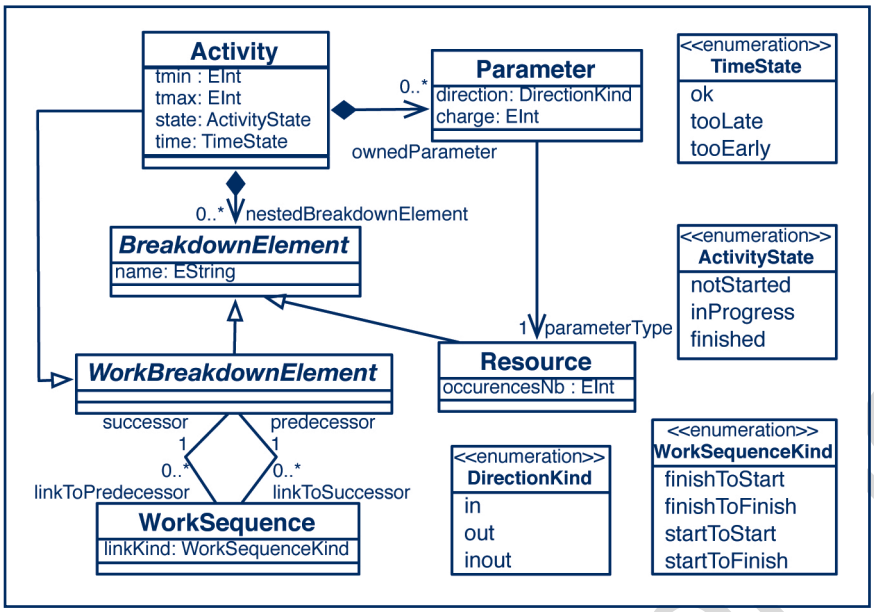


*Figure 4. Extrait du métamodèle xSPEM.*

### 3.1. Le métamodèle xSPEM pour l'exécution des processus

Une lacune de SPEM concerne l'exécution des processus modélisés. Si cet aspect faisait partie des objectifs du standard, il n'a malheureusement pas été traité, l'approche préconisée consistant à traduire le modèle de procédé pour alimenter un outil de gestion de projet.

Comme notre objectif est de pouvoir exécuter et vérifier un processus, nous proposons dans [1] l'extension xSPEM, qui ajoute les informations nécessaires à l'exécution tout en restant compatible avec SPEM. La figure 4 en présente les principaux concepts. Certaines des informations ajoutées doivent être données lors de la définition du processus : 1) les ressources, humaines et matérielles, affectées au projet, 2) celles nécessaires pour réaliser une activité et 3) le délai dans lequel l'activité doit être réalisée (*tmin* et *tmax*). Les autres correspondent aux états du processus que l'on souhaite observer pendant l'exécution. Ils dépendent de l'objectif de cette exécution. Ainsi, pour vérifier la faisabilité d'un processus, il faut connaître l'état (non démarrée, en cours, terminée) d'une activité (attribut *state*) et si elle est dans les temps, en avance ou en retard (attribut *time*). Pour suivre un processus réel, on définirait des informations plus précises : le taux de réalisation d'une activité et le temps passé dessus.

Pour définir le métamodèle de xSPEM, nous avons utilisé le langage Ecore du projet EMF[5]. Il ressemble à une version simplifiée du diagramme de classe d'UML (*Unified Modeling Language*). Il permet ainsi de décrire les propriétés structurelles des modèles. Un langage de contrainte comme OCL [6] (*Object Contraint Language*) permet d'exprimer des contraintes supplémentaires (par exemple, l'unicité du nom d'une activité). Un modèle est dit conforme quand il respecte ces deux types de contraintes. On appelle métamodélisation l'activité consistant à définir des langages de modélisation au moyen de métamodèles.

Le projet EMF propose un éditeur graphique de modèles Ecore et un ensemble d'outils pour considérer ces modèles Ecore comme des métamodèles : génération du code Java pour gérer les modèles correspondants, sérialisation XML (*Extensible Markup Language*) et un éditeur simple et arborescent pour saisir un modèle conforme à ce métamodèle.

### 3.2. Construction d'un éditeur graphique

Avec l'outil EPF, l'utilisateur doit saisir toutes les informations sur son processus sous forme tabulaire avant de pouvoir le visualiser graphiquement grâce à la génération automatique réalisée par la fonction publication de l'outil. Dans TopPROCESS, à l'instar de l'utilisation d'UML par les programmeurs, nous pensons préférable que le concepteur commence par dessiner graphiquement les principaux éléments du processus avant d'en donner une description plus précise. L'accent est donc mis sur l'architecture générale du processus.

Aussi, nous avons défini un éditeur graphique pour xSPEM, *TopPROCESS Modeler,* qui permet de décrire les activités et leurs dépendances, la décomposition d'une activité en sous-activités, les produits manipulés et les ressources utilisées.

Le projet Eclipse GMF[6] nous a permis d'engendrer rapidement et facilement cet éditeur. Partant du métamodèle Ecore xSPEM et du code Java engendré par EMF, il

[5] *Eclipse Modeling Framework*, http://www.eclipse.org/emf/

[6] *Graphical Modeling Framework*, http://www.eclipse.org/gmf/

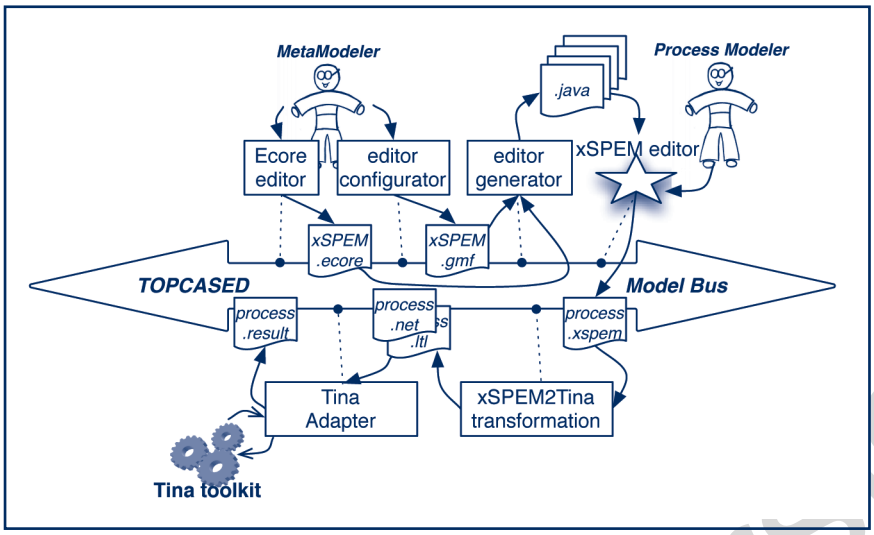


*Figure 5. Architecture de Topcased pour la vérification de modèles xSPEM.*

a suffit de décrire les éléments de l'éditeur souhaité en s'appuyant sur trois modèles :

1) le modèle graphique décrit la représentation graphique des éléments et leur nature (nœud, arc, label, etc.). Par exemple, une activité est un nœud représenté par un rectangle avec son nom à l'intérieur (label). Une dépendance est une flèche ;
2) le modèle des outils décrit la palette de l'éditeur sous la forme de plusieurs groupes d'outils (activité, dépendance, etc.) ;
3) le modèle de correspondance fait le lien entre les deux modèles précédents et le métamodèle Ecore. Il permet d'associer l'outil *activité*, au dessin d'une activité (rectangle) et à la création dans le modèle d'une nouvelle activité (instance de *Activity*). De même, l'outil *dépendance* trace un arc entre deux activités et met à jour les références correspondantes dans le modèle de processus.

Une fois ces modèles décrits, l'éditeur graphique est automatiquement engendré. Ainsi, nous disposons d'une syntaxe concrète graphique pour notre langage xSPEM. Si des adaptations non prévues par GMF sont nécessaires, il faut modifier le code engendré. Cependant, l'éditeur initial est tout à fait utilisable.

### 3.3. Vérification des processus xSPEM

OCL n'est pas adapté pour vérifier la faisabilité d'un processus, c'est-à-dire vérifier qu'il peut s'exécuter complètement en respectant les contraintes de dépendance, de temps et de ressources. Aussi, nous nous appuyons sur des outils de *model-checking*, plus précisément ceux de la boîte à outils Tina [2] définie pour les réseaux de Petri. Un *model-checker* consiste, pour un programme donné, à étudier toutes ses exécutions possibles pour vérifier si une propriété est toujours vérifiée ou non. Dans la négative, il exhibe une exécution, appelée contre-exemple, qui contredit la propriété. Considérant la propriété « le processus ne se termine pas », le contre-exemple est un exemple d'exécution du processus qui respecte les contraintes imposées. Cette propriété est écrite en LTL (Logique Temporelle Linéaire) sous la forme : quand le processus n'évolue plus, toute activité est dans l'état « terminée dans les délais ».

Pour utiliser Tina, il nous faut bien sûr traduire notre modèle de processus en un réseau de Petri équivalent. La figure 5 décrit l'approche suivie. Le concepteur édite son modèle de processus avec l'éditeur graphique et demande la vérification de son modèle. Il appelle pour cela un service disponible sur le bus de modèle fourni par l'atelier Topcased. Ce service lui indique ensuite le résultat de la vérification. Le service est implanté en s'appuyant sur des transformations de modèle pour traduire le processus en réseau de Petri et, en retour, afficher les résultats au concepteur. Ainsi, une transformation « modèle à modèle » transforme un modèle de processus en un modèle de réseau de Petri ; une transformation « modèle à texte » transforme le modèle de réseau de Petri en un texte respectant la syntaxe de Tina. Enfin, une dernière transformation construit les propriétés LTL à partir du modèle de processus. Ces transformations ont été écrites en utilisant le langage ATL[7] dédié aux transformations. L'intérêt d'utiliser un langage de transformation est de disposer d'un outil de haut niveau d'abstraction qui permet de se concentrer sur les correspondances entre les éléments des modèles source et cible et non sur les détails de mise en œuvre. La transformation est plus rapide à écrire, plus facile à comprendre et plus facile à faire évoluer. Le principe de cette approche est disponible sous forme d'une étude de cas Eclipse[8] appliquée à un sous-ensemble de xSPEM.

[7] *AtlanMod Transformation Language*, http://www.eclipse.org/m2m/atl/
[8] Voir http://www.eclipse.org/m2m/atl/usecases/SimplePDL2Tina/

## 4. Conclusion

Nous avons décrit dans cet article l'approche TopPROCESS ainsi que xSPEM, un DSML (*Domain Specific Modeling Language*) dédié à l'ingénierie des processus. Nous appuyant sur l'IDM, nous avons défini sa syntaxe abstraite (ses concepts et leurs relations) au moyen de son métamodèle, nous l'avons doté d'une syntaxe concrète graphique (éditeur engendré à partir d'une description de ses constituants) et avons défini sa sémantique par traduction d'un modèle xSPEM en réseau de Petri (langage formel exécutable). Cette traduction nous permet de réutiliser les outils disponibles sur les réseaux de Petri, en particulier le *model-checker* Tina pour vérifier le modèle de processus (faisabilité du processus vis-à-vis des contraintes de dépendance, de temps et de ressources).

La maturité des standards et des techniques de modélisation permet aujourd'hui l'application industrielle de l'IDM à un domaine comme la gestion des processus. Elle permet de construire des composants de processus qui « embarquent » le savoir-faire, de les assembler pour bâtir les cycles de production et enfin piloter de manière proactive l'exécution de ces processus. Un cycle qui va de la conception, où la connaissance explicite est formalisée, jusqu'à la production pour laquelle une approche collaborative du travail laisse la connaissance implicite s'exprimer.

Au-delà de la faisabilité d'un processus, nous souhaitons pouvoir démontrer, avant le démarrage du projet et donc un engagement financier lourd, qu'il répond en tout point aux exigences émises, par exemple par l'assurance qualité. Nous pensons aussi que la modélisation du processus peut être utilisée pour construire des outils de simulation de processus (en cours d'intégration dans Topcased) et, surtout, pour gérer pendant son déroulement un projet. Cet axe s'inscrit dans le MDSM, *Model Driven System Management*.

## Références


[1] R. BENDRAOU, B. COMBEMALE, X. CREGUT, M.-P. GERVAIS, "*Definition of an eXecutable SPEM 2.0*", in APSEC'07, pp. 390–397, Japan, 2007, IEEE Computer Society.

[2] B. BERTHOMIEU, P.-O. RIBET, F. VERNADAT, "*The Tool TINA – Construction of Abstract State Spaces for Petri Nets and Time Petri Nets*", International Journal of Production Research, 42(14):2741–2756, July 2004.

[3] J. BEZIVIN, A. VALLECINO-MORENO, J. GARCIA-MOLINA, G. ROSSI, "*MDA at the Age of Seven: Past, Present and Future*", UPGRADE, The European Journal for the Informatics Professional, IX(2):4-5, April 2008.

[4] J. ESTUBLIER, "*Software are Processes Too*", In Invited Paper at Software Process Workshop, Bejing, China, May 2005.

[5] OMG, "*MDA Success Stories*", http://www.omg.org/mda/products_success.htm

[6] OMG, "*Object Constraint Language (OCL) 2.0 Specification*", May 2006.

[7] OMG, "*Software & Systems Process Engineering Meta-Model (SPEM) 2.0*", April 2008.




### Les auteurs


**Angel Garcia** est ingénieur CNAM (Conservatoire National des Arts et Métiers), membre de la Conférence des Grandes Écoles, spécialiste en ingénierie des modèles logiciels et en amélioration des processus organisationnels de développement. Il intervient en qualité d'enseignant auprès de l'université de Toulouse. Il est responsable du groupe des experts Qualité du CNAM pour la région Midi-Pyrénées.
Après une expérience de 6 ans en qualité de chef de projet en développement logiciel chez un éditeur de logiciel *startup* puis leader dans le domaine de la gestion financière des télécoms, Angel Garcia oriente ses activités vers l'approche processus et passe par le laboratoire de l'IRIT/CNRS. En 2005, il fonde la société Tectosages qu'il dirige actuellement.

**Benoît Combemale** est assistant de recherche dans l'équipe AtlanMod commune à l'INRIA et à l'EMN. Il a obtenu son doctorat d'informatique à l'Université de Toulouse sur le thème de la sémantique d'exécution des langages de modélisation pour la simulation et la vérification de modèles, en particulier de processus. Il investigue actuellement des techniques pour la manipulation de modèles infinis et l'utilisation des modèles à l'exécution pour la gestion de systèmes. Benoît Combemale s'implique également dans la dissémination de l'IDM au sein de différentes formations d'ingénieurs et universitaires en France et à l'étranger.

**Xavier Crégut** est ingénieur en informatique de l'ENSEEIHT, docteur en informatique de l'INPT. Il est actuellement maître de conférences à l'ENSEEIHT et chercheur à l'IRIT. Ses activités de recherche concernent la sémantique des langages de modélisation dédiés (DSML) à des fins de validation et de vérification des modèles construits. L'ingénierie des procédés constitue un domaine d'application de ses travaux qui sont aussi mis en pratique dans le projet Topcased du pôle de compétitivité Aerospace Valley.

**Jérôme Vandeur** est ingénieur développement logiciel chez Tectosages.