Exécutabilité des modèles : réflexions et expériences

Carine Souveyet, Saïd Assar

Exécutabilité des modèles: réflexions et expériences

Carine Souveyet* — Saïd Assar,***

** Université Paris 1 La Sorbonne – Centre de Recherche en Informatique*
90, rue de Tolbiac
F-75013 Paris
carine.souveyet@univ-paris1.fr

*** Institut National des Télécoms – Département Systèmes d'Information*
9, rue Ch. Fourier
F-91011 Evry Cedex
said.assar@int-evry.fr

RÉSUMÉ. *Traditionnellement, les modèles conceptuels utilisés dans l'ingénierie des systèmes d'information sont contemplatifs et non exécutables. Pour qu'un modèle soit productif, il faut que des propriétés opérationnelles soit attachées aux concepts du modèle. On distingue trois catégories de propriétés opérationnelles : la validation, l'exécution et la transformation. L'objectif de cet article est de présenter à travers le développement de plusieurs projets d'outils logiciels complexes pour l'ingénierie des SI comment ces trois stratégies ont été mises en œuvre.*

MOTS-CLÉS: *modèles productifs, méta-modélisation, modèle de processus, exécution des modèles, ateliers de génie logiciel (AGL)*

1. Introduction

La complexité croissante des systèmes d'information dont les organisations ont besoin fait que la conception et le développement de ces systèmes devient de plus en plus complexe, coûteux et difficile. L'usage de méthodes d'ingénierie s'impose pour aider à maîtriser la complexité des problèmes rencontrés. Le terme méthode vient du grec « methodos » qui signifie « moyen d'investigation ». D'après Seligman, ces moyens sont constitués par « *a way of thinking, a way of modelling, a way of working and a way of supporting* ». Toute méthode est définie autour d'une philosophie ou paradigme (*way of thinking*), comporte des modèles (*ways of modelling*) pour définir le produit, propose des modèles de processus ou démarches (*way of working*) et est supportée par des outils logiciels (*way of supporting*) pour aider à la mise en œuvre de la méthode.

Par analogie avec le génie civil visant, par exemple, à construire des ponts, l'ingénierie des SI a comme objectif l'élaboration de produits que sont les systèmes d'information. Les produits SI, comme les ponts, doivent se décrire à différents niveaux de détail et d'abstraction. Traditionnellement, les modèles proposés par l'ingénierie des SI pour de telles descriptions sont dits *contemplatifs* c'est à dire qu'ils sont utilisés uniquement pour faciliter la communication entre les différents acteurs du projet. Néanmoins, un modèle peut être (ou devenir) *productif*, c'est-à-dire capable de montrer un comportement et de fournir un résultat. La nature contemplative ou productive d'un modèle dépend de celui-ci et de l'usage que l'on en fait. Pour que les modèles soient productifs, ils doivent être interprétables et manipulables par une machine. «Quoi faire» avec ce modèle permet de décrire la facette de production du modèle. Le développement de modèles productifs est naturellement apparu, en premier lieu, dans l'ingénierie des processus. Il est ensuite introduit dans le domaine de l'ingénierie des SI grâce aux travaux de l'OMG autour de MOF [MOF 97]. Il est devenu un standard de développement depuis les travaux de l'OMG autour de l'approche MDA [MDA 00] et l'objet d'une nouvelle discipline appelée ingénierie dirigée par les modèles [Atkinson & all 03] [Bezivin & all 05].

Cet article est une réflexion sur le développement de modèles productifs. Il présente plusieurs travaux de recherche qui ont aboutit au développement de modèles productifs à travers la construction d'outils logiciels. Il est structuré comme suit: la section 2 introduit les propriétés de modèles productifs et les différentes stratégies de développement d'outils de production pour de tels modèles. La section 3 présente des projets de recherche dans lesquels certaines de ces stratégies ont été utilisées. La dernière section nous permettra de conclure en évoquant d'autres travaux en cours sur le même thème.

2. Modèles productifs et stratégies de développement

Un modèle est productif ou exécutable si les conditions suivantes sont remplies:

- il contient un ensemble de propriétés opérationnelles permettant l'action de production,

- il est décrit à un niveau de détail suffisant pour qu'une machine puisse exécuter cette action de production,

- il y a une volonté ou un besoin explicite d'outiller l'action de production inhérente au modèle.

La définition d'un modèle productif nécessite l'identification de propriétés opérationnelles du modèle pour pouvoir dériver l'action de production à appliquer sur les instances du modèle comme le montre la Figure 1.

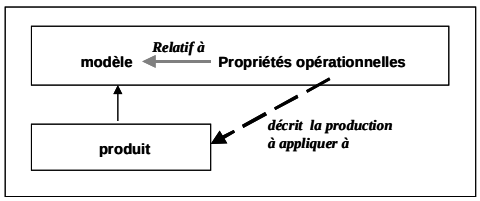

Figure 1. *Cadre de référence d'un modèle productif*

Les différents types d'action de production possibles à partir d'un modèle que nous avons identifiés sont : valider, exécuter et transformer.

– L'action « *valider* » consiste à appliquer un ensemble de règles de validité - définies avec un modèle - à toute instance de ce modèle.

– L'action « *exécuter* » s'applique aux modèles contenant des concepts associés sémantiquement à la description d'un comportement. Les propriétés opérationnelles de tels modèles explicitent l'interprétation sémantique du modèle pour produire un comportement (ou pour conduire le comportement selon les termes du modèle).

– L'action « *transformer* » permet de construire une nouvelle version d'un schéma à l'aide d'un modèle similaire ou ayant d'autres propriétés, telle que l'exécutabilité. Définie d'une manière *ad-hoc*, la transformation d'un modèle est généralement décrite sous la forme de règles de correspondance et de « mapping ».

Pour manipuler des modèle exécutables, l'ingénierie du logiciel introduit deux catégories fondamentales d'outils : l'interpréteur et le compilateur. Un interpréteur est un outil qui induit une action à partir de l'interprétation d'une instance conforme à un modèle ou à un langage. Les compilateurs sont des outils qui, à partir de l'analyse d'une instance conforme à un modèle (ou à langage) génèrent un code logiciel exécutant l'action de production.

Dans le cadre de nos activités de recherche, nous avons été amené à appliquer les cinq types de développement suivants :

(1) le développement d'un **interpréteur** est intéressant lorsque l'on veut avoir un couplage fort entre l'outil et l'instance du modèle. L'évolution de l'instance du modèle impacte directement le comportement de l'outil et, en outre, l'outil de production s'applique à toutes les instances du modèle.

(2) le développement d'un **compilateur** induit un couplage faible entre l'outil et l'instance du modèle. En effet, l'évolution de l'instance du modèle implique la génération d'une nouvelle version de l'outil de production par le compilateur. Le compilateur permet de construire un outil de production spécifique à chaque instance du modèle.

(3) le développement d'un **outil spécifique dirigé par le modèle** s'applique lorsque l'outil de production à développer n'est dédié qu'à une seule instance du modèle et que des règles de développement ou de structuration de l'outil de production sont induites du modèle.

(4) le développement d'un **outil spécifique à l'aide d'une bibliothèque logicielle** facilite le développement de l'outil de production par application de la stratégie précédente.

(5) le développement d'un **interpréteur par méta-modélisation** s'applique lorsque le modèle et les propriétés comportementales permettant de le qualifier de productif sont capturés dans un méta-modèle. L'outil de production dirige l'action de production selon les termes du méta-modèle. Il s'applique aux instances de n'importe quel modèle décrit selon les termes du méta-modèle. Cette méta-modélisation peut s'appliquer pour développer un interpréteur ou un compilateur.

Ces différentes approches de recherche aboutissant à la proposition de modèle ou de méta-modèle productif mettent en évidence une caractéristique commune qui est le souci scientifique d'avoir des modèles ou méta-modèles (1) basés sur des concepts dont la sémantique est clairement définie (formalité sémantique du modèle), (2) dont les règles de validité sont clairement énoncées (validité du modèle) et (3) ayant des propriétés opérationnelles clairement identifiées pour décrire l'action de production (productivité du modèle). Ces trois critères doivent être satisfaits pour atteindre l'exécutabilité des modèles.

En effet, le développement de l'outil de production ne peut se faire que sur un modèle dont *la sémantique est clairement identifiée*, et avec *un énoncé explicite et précis des règles opérationnelles* pour l'action de production. De plus, l'usage de la fonction de production à travers cet outil ne peut se faire que sur *une instance de modèle valide*.

3. Exemples de développement de modèles productifs

Nous allons présenter brièvement dans cette section des travaux de recherche qui ont aboutit – directement ou indirectement – à définir et à manipuler des modèles productifs.

3.1. *Transformation et validation : approche de développement d'outils par méta-modélisation*

L'atelier logiciel Rubis [Rolland & all 87, Lingat 88] supporte la conception, la validation et le prototypage d'un schéma conceptuel conforme au modèle Remora [Rolland 82, 88]. Ce modèle est contemplatif puisqu'il offre une vision synthétique du comportement de l'application. Néanmoins, il est considéré aussi comme productif dans l'atelier Rubis puisque cet outil offre des fonctions de prototypage du comportement de l'application. Le modèle est devenu productif parce que d'une part, il avait des propriétés comportementales et d'autre part, la spécification des opérations, des conditions et des prédicats d'événements est formulée dans le langage exécutable Proquel [Lingat & al. 88].

Pour valider d'une instance du modèle Remora décrite dans l'atelier Rubis, nous avons proposé un méta-modèle de produit permettant de décrire les règles de validité du modèle à construire [Souveyet & Rolland 90, Souveyet 91]. Comme le synthétise la Figure 2, le méta-modèle de produit est associé à un méta-type contrainte-type pour capturer les règles de validité des éléments du modèle : éléments de produit et associations d'éléments de produit au niveau modèle.

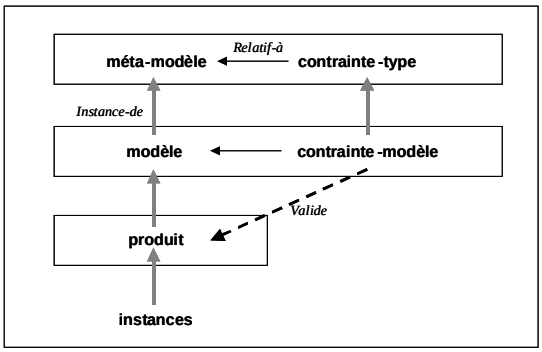

Figure 2. *Prise en compte des règles de validation par méta-modélisation*

Cette spécification des règles de validité est intégrée dans une architecture orientée objet multi-niveaux présentée dans la Figure 3. Les règles de validité sont des objets au niveau modèle (instance d'une classe de type contrainte-type) associées à l'élément de produit ou association qu'il permet de valider.

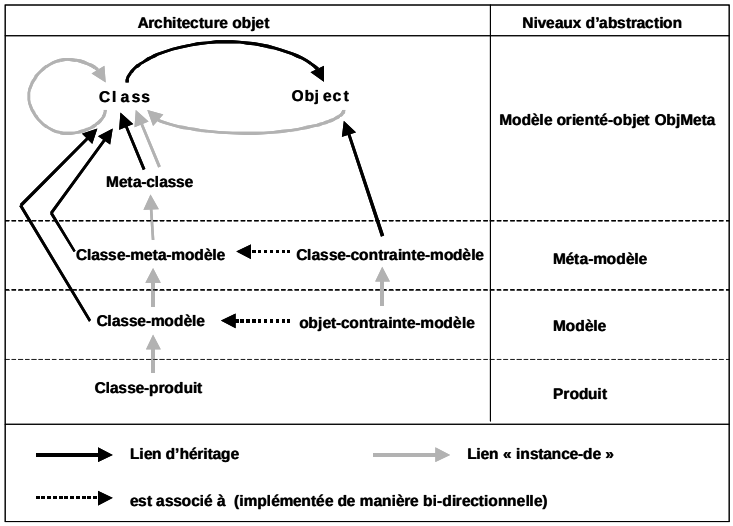

Figure 3. *Réalisation du méta-modèle et règles da validité dans une architecture objet*

Cette stratégie permet d'intégrer dans l'implémentation objet du méta-modèle les fonctionnalités de validation qui seront intégrées au niveau modèle. Cette implémentation est de type compilateur car le changement de modèle nécessite une recompilation des classes obtenues par instanciation du méta-modèle.

Un autre travail de recherche basé sur l'usage de méta-modèle a abordé la question de la transformation des schémas (prototypés et validés dans Rubis) en code transactionnel pour un SGBD cible [Assar 95]. L'approche choisie consiste à utiliser un méta-modèle de produit pour décrire les schémas source et cible et un méta-modèle de processus pour décrire les transformations interactives qu'il faut appliquer au modèle source pour aboutir au modèle cible (Figure 4),.

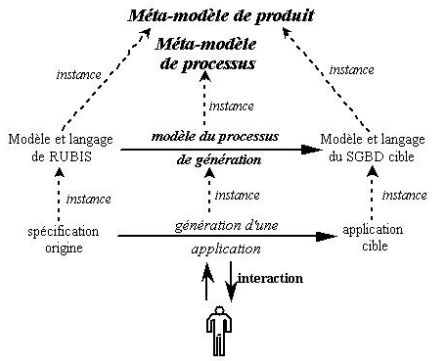

Figure 4. *Cadre général pour l'automatisation du processus de dérivation*

Le méta-modèle de produit est dérivé de celui utilisé pour l'expression des contraintes de validation (c.f. début de cette sous-section). Le méta-modèle de processus utilisé est une variante de celui utilisé dans le projet Nature décrit dans la prochaine sous-section.

4.2. *L'interpréteur du Méta-modèle de processus Nature : Mentor*

La formalisation des démarches de méthode d'ingénierie des SI nous a conduit à proposer le méta-modèle de processus orienté-contexte Nature. Pour démontrer l'efficacité de l'approche orientée contexte dans la formalisation et le guidage des processus d'ingénierie, nous avons développé l'outil Mentor. Le méta-modèle de processus Nature est utilisé dans ce cadre là comme un modèle productif puisque l'on veut pouvoir contrôler l'exécution des processus d'ingénierie selon les termes du méta-modèle. [Si-Said 96, Grosz & al. 97]

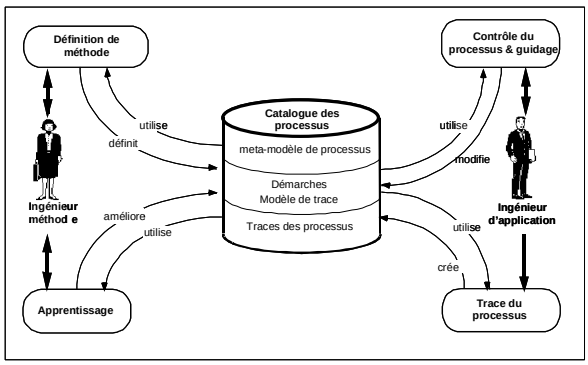

Figure 5. *Architecture de Mentor*

L'environnement Mentor combine un outil CAME pour la définition d'une méthode (un modèle de produit et un modèle de processus) et l'outil de gestion, d'exécution et de guidage du processus (Figure 5).

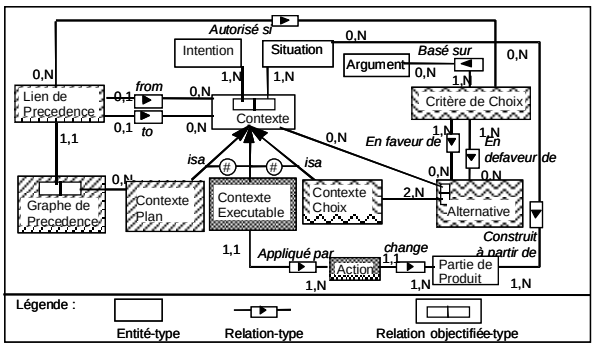

Figure 6. *Vue générale des concepts du méta-modèle de processus.*

Les propriétés opérationnelles du modèle doivent être définies pour pouvoir envisager le développement d'un outil. Le méta-modèle Nature permet de décrire le raisonnement intentionnel d'un ingénieur d'application dans son activité de production d'un produit. Ce raisonnement est construit autour du concept de contexte et de ses sous-types : exécutable, choix et plan (Figure 6).

4.3. *Le compilateur du modèle WebWorkflow : WebWorkFlow Generator*

Dans le cadre du projet IST E-Utilities, une démarche de développement d'application Web par génération à partir d'un modèle appelé WebWorkflow a été développée. L'objectif de ce projet était d'élaborer le concept d'une plateforme européenne B2C de vente d'énergie aux consommateurs européens, dans l'hypothèse d'un marché en libre concurrence [Souveyet & all, Rolland & al. 03]. L'objectif de la démarche de développement par génération est de faciliter l'intégration d'un nouveau fournisseur d'électricité à la plateforme en accélérant le développement (et l'évolution future) des pages qui lui sont spécifiques pour les contrats et la facturation.

Le modèle productif est le modèle de Workflow adapté pour le Web décrit à la figure 7, et l'action de production dans ce cas est de transformer cette spécification en code Java. Les règles opérationnelles à identifier sont les règles permettant de transformer les tâches et leur relations en pages JSP effectuant des activités internes pour finalement générer des pages web contenant des liens permettant d'invoquer la prochaine tâche. Le processus décrivant une fonctionnalité de l'application web est décrite par un workflow comportant un ensemble de tâches connectées entre elles par des connecteurs de type séquence, OR-split (branchement conditionnel) et OR-join (point de rendez vous ou synchronisation).

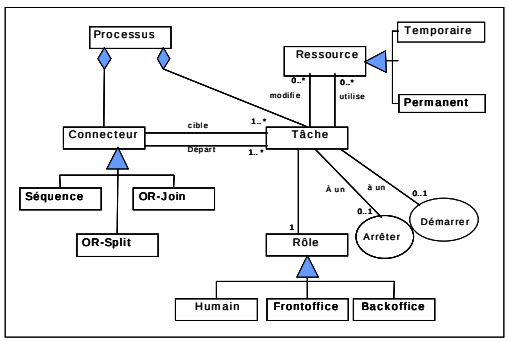

Figure 7. *Le modèle Web WorkFlow*

Pour que le modèle devienne productif, il a fallu décrire les tâches comme un enchaînement d'actions. Une typologie des actions-types est introduite : importer, exporter, extraire, calcul, afficher, etc. Tous ces types d'actions sont définis de manière générique et implémentés en JAVA dans une bibliothèque de classes JAVA facilitant la génération du code [Dimassi & al. 04].

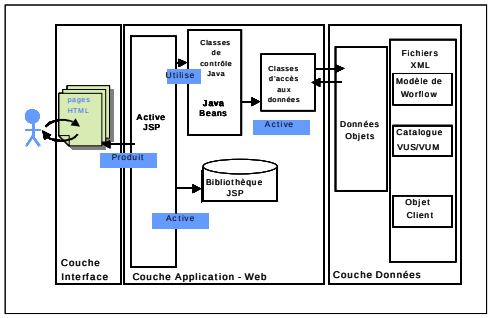

Figure 8. *Architecture de l'application Web générée*

L'outil de transformation du modèle de navigation web de type WebWorkflow en code Java a été réalisé en suivant la stratégie de type compilateur. Le type compilateur comme outil de transformation est suffisant dans ce cas car cette approche par génération doit accélérer le développement et faciliter la maintenance. Cet outil est décrit dans la figure 8.

4.4. *Le développement de CREWS-L'Ecritoire dirigé par le modèle de carte*

CREWS-L'Ecritoire a été développé pour démontrer la pertinence de l'approche à base de but dans la découverte et l'explicitation des besoins d'un projet SI [Rolland & al. 98]. Dans cette approche, les besoins sont exprimés à l'aide des fragments de besoin reliant un but au scénario permettant de le satisfaire. Le processus d'ingénierie des besoins (IB) est composé de deux activités – écrire un scénario à partir d'un but et découvrir de nouveaux buts à partir des scénarios. Le processus de découverte correspond à un mouvement bidirectionnel du but vers le scénario et d'un scénario vers des buts. Dans le sens direct, le scénario est un moyen de pallier au caractère abstrait d'un but concrétisant une des ses réalisations dans un scénario. Grâce à cette association, les buts irréalistes peuvent être détectés et ainsi pallier aux limites des approches d'ingénierie des besoins dirigées par les buts. Dans le sens inverse, du scénario vers les buts, la relation permet de découvrir des buts réalistes.

L'outil CREWS-L'Ecritoire guide l'application de la démarche et automatise, d'une part, les règles de découverte de buts à partir de scénarios et, d'autre part, la conceptualisation des scénarios écrits en langue naturelle [Tawbi & Souveyet 98, Souveyet & all 00]. Les règles de découverte de buts et les directives de rédaction de scénarios ont été intégrées dans la carte de processus l'Ecritoire comme le montre la Figure 9. L'outil L'Ecritoire est conçu spécifiquement pour cette carte mais une approche de développement dirigée par le modèle de carte est définie et appliquée pour obtenir une architecture flexible construite à partir de celle-ci. La caractéristique de l'approche de développement est de concevoir l'outil comme un assemblage de fragments de logiciels réalisant les fragments de méthode eux même assemblés dans la carte l'Ecritoire.

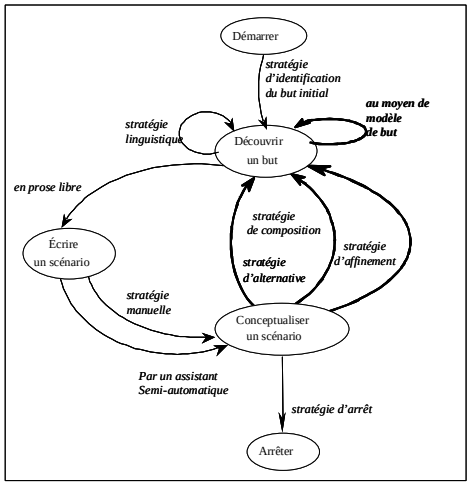

Figure 9. *Carte de processus de l'Ecritoire*

La réalisation des règles méthodologiques de découverte de but ou de rédaction scénario est effectuée dans un composant logiciel réutilisable afin de promouvoir une approche de définition de méthode pour la réutilisation. En effet, cette approche est à coupler à l'approche de construction de méthode par réutilisation proposée par Ralyté [Ralyté 01]. Cet aspect permet d'innover et de prendre en compte la construction de l'outil supportant la nouvelle méthode construite par réutilisation de fragments de méthode.

4. Conclusion

A partir de notre expérience menée dans plusieurs projets de recherche, nous avons présenté dans cet article une réflexion sur le concept de modèle productif. Nous avons identifié cinq stratégies différentes pour opérationnaliser les propriétés productives d'un modèle. La présentation de quelques projets de recherche dans lesquels nous avons participés dans le passé nous a permis d'illustrer quelques unes de ces stratégies. Que ce soit pour la validation ou pour l'exécution, les approches développées utilisent intensivement les techniques de méta-modélisation.

Nos travaux de recherche actuels restent centrés sur cette problématique d'exécutabilité de modèle de haut niveau d'abstraction. Nous travaillons sur le développement d'un moteur d'exécution pour le modèle de cartes de processus [Rolland & al. 99]. Un préalable à cette réalisation est une meilleure compréhension de la sémantique opérationnelle du modèle dont une première formulation a été présentée dans [Assar 00]. Un premier prototype a été élaboré dans [Edme 05] et à partir duquel une nouvelle version basée sur la technologies des web services est en cours de conception et de développement.

5. Bibliographie

Assar S., "*Méta-modélisation pour la transformation des schémas et la génération de code*", Thèse de l'université Paris 6, 15 décembre 1995.

Assar S., Ben-Achour C., Si-Saïd S., "Un Modèle pour la Spécification des Processus d'analyse des Systèmes d'Information", Actes de la conférence *INFORSID 2000*, Lyon, mai 2000.

Atkinson C., Kûhne T., "Model Driven Development : A metamodelling foundation", *IEEE Software*, September 2003.

Bézivin J., Jouault F., Touzet D., "Principles, Standards and Tools for Model Engineering", Proceedings of *10th Int. Conf. on Engineering of Complex Computer Systems*, ICECCS 2005, 16-20 June 2005, Shanghai, China. IEEE Computer Society 2005

Dimassi J., Souveyet C., Rolland C., "The Concept and Implementation of the Market Place E-Utilities*COM.", Proceedings of the *6th Int. Conf. on Enterprise Information Systems*, ICEIS 2004, Porto, Portugal, April 14-17, 2004, pp 337-347.

Edme M-H., *"Proposition pour la modélisation intentionnelle et le guidage de l'usage des systèmes d'information"*, thèse de l'université Paris 1 La Sorbonne, 09 juillet 2005.

Grosz G., Rolland C., Schwer S., Souveyet C., Plihon V., Si-Said S., Ben Achour C., Gnaho C., "Modelling and Engineering the Requirements Engineering Process : An Overview of the NATURE Approach", *Requirements Engineering Journal*, vol. 2, pp. 115 - 131, 1997.

Lingat J-Y., "*Rubis : un système pour la spécification et le prototypage d'applications Bases de Données"*, these de l'Université P&M Curie (Paris VI) , Juillet 1988.

Lingat J-Y., Assar S., Colignon P., Rolland C., "PROQUEL: a PROgramming QUEry Language" dans R. Hull, R. Morrison, D. W. Stemple (Eds.): *Proceedings of the Second International Workshop on Database Programming Languages*, 4-8 June, 1989, Oregon-USA.

Object Management Group, "OMG/MOF Meta object Facility (MOF) Specification", September 1997, http://www.omg.org/mof/ (accede le 18 avril 07).

Soley R., "The OMG staff MDA, Model-Driven Architecture", November 2000, disponible sur http://www.omg.org/mda/presentations.htm (accede le 18 avril 07).

Rolland C., Richard C., "The Remora Methodology for Information Systems Design and Management", dans *Information Systems Design Methodologies : a comparative Review*, T. W. Olle, H. G. Sol, A.A. Verrijn-Stuart (eds), North Holland (pub), 1982.

Rolland C., Cauvet C., Nobecourt P., Proix C., Collignon P., Lingat J-Y., Souveyet C., "The Rubis System" Proceedings of the *IFIP WG 8.1 Working Conference on Computerized Assistance During the Information Systems Life Cycle*, CRIS 88, Egham, England, 19-22 September, 1988. North-Holland 1988, pp. 193-239.

Rolland C., Foucault O., Benci G., *"Conception des systèmes d'information: La méthode Remora"*, Editions Eyrolles, 1988.

Rolland C., Souveyet C., Ben Achour C., “Guiding Goal Modeling Using Scenarios”, IEEE Transaction on Software Engineering (TSE), Vol. 24, n°°1, 1998, pp 1055-1071.

Rolland C., Prakash N., Benjamen A., “A Multi-Model view of Process Modelling”, *Requirements Engineering Journal*, 4, pp. 169-187, 1999.

Rolland C, Loucopoulos P et al., "E-Utilities - transforming utilities into customer-centric multi-utilities" Proceedings of *17th Int. Conf. on Electricity Distribution*, Barcelona, Spain, 12 - 15 May 2003.

Si-Said S., Rolland C., Grosz G., "MENTOR : A Computer Aided Requirements Engineering Environment", Proceedings of *8th Int. Conf. on Advanced information Systems Engineering*, CAISE’96, Heraklion, Crete, Greece, May 1996.

Ralyté J., Rolland C., “An Assembly Process Model for Method Engineering” Proceedings of the 13th Int. Conf. on Advanced Information Systems Engineering, CAISE 2001, Interlaken, Switzerland, 2001.

Souveyet C., Rolland C., “Correction of Conceptual schema”, pp 152-174, Proceedings of *2nd Int. Conf. on Advanced Information Systems Engineering*, CAISE’90, Stockholm, Mai 1990.

Souveyet C., *“Validation des spécifications conceptuelles d'un système d'information”* Thèse de l'Université P&M Curie (Paris VI), Février 1991.

Souveyet C., Tawbi M., “Process Centred Approach for Developing Tool Support of Situated Methods”, Database and Expert Systems Applications (DEXA), Vienna, Austria, August 1998, pp 206-215.

Souveyet C., Tawbi M., Velez J. F., Ben Achour C., “Evaluating the CREWS-L'Ecritoire Requirements Elicitation Process”, Proceedings of *Requirements Engineering: Foundation for Software Quality Conference* (REFSQ), Stockholm, Sweden, June 2000.

Souveyet C., Rolland C., Dimassi J., Atintasli D., *Rapport Technique WP5/T5.3 du projet IST e-utilities : Web development driven by code generation*, Juin 2002.

Tawbi M., Souveyet C., Rolland C., “L’ECRITOIRE a tool to support a goal-scenario based approach to requirements engineering”, *Information and Software Technology journal*, Elsevier Science, B.V, 1998.