Catégorie : chercheur

# Vue stratégique sur l'ingénierie des méthodes [1]

**Jolita Ralyté**

Centre Universitaire d'Informatique, Université de Genève
24, rue de Général Dufour, CH-1211 Genève 4
ralyte@cui.unige.ch

**Résumé**

La discipline de l'ingénierie des méthodes a pour but d'aider à la construction de nouvelles méthodes d'ingénierie des systèmes d'information. L'article prend en compte la perspective processus de l'ingénierie des méthodes. Il utilise un cadre de référence qui inclut les différentes démarches de construction de méthodes dans un modèle de processus stratégique. C'est un état de l'art qui survole les approches d'ingénierie des méthodes existantes, les classe suivant le cadre de références, les compare et met en évidence les tendances et l'évolution de l'ingénierie des méthodes.

**Mots-clés**

Ingénierie des méthodes, composant de méthode, CAME, méthode situationelle, assemblage.

## 1. Introduction

De plus en plus de phénomènes de notre société sont touchés par les Systèmes d'Information (SI). La complexité des SI ne cesse de croître et par conséquent leur développement devient de plus en plus complexe, coûteux et difficile. L'utilisation de méthodes d'ingénierie pour conduire le cycle de développement d'un SI aide à mieux maîtriser la complexité des problèmes à informatiser. La définition de nouvelles méthodes d'ingénierie des SI est prise en compte par la discipline de l'ingénierie des méthodes. S. Brinkkemper [BRIN96] définit l'ingénierie des méthodes comme *"une discipline de conceptualisation, de construction et d'adaptation de méthodes, de techniques et d'outils pour le développement des systèmes d'information"*.

Même s'il existe à ce jour plusieurs méthodes d'ingénierie des systèmes, l'expérience a montré que ces méthodes ne sont pas universelles et elles ne peuvent pas prévoir toutes les situations possibles. Avec la croissance de la complexité des domaines d'application, la construction de nouvelles méthodes devient nécessaire. Une méthode qui a fait ses preuves dans un domaine d'application peut être inadéquate dans d'autres domaines car la situation d'ingénierie de chaque

[1] Ce travail a été réalisé dans le Centre de Recherches en Informatique de l'université Paris 1 – Sorbonne. http://panoramix.univ-paris1.fr/CRINFO/

application est différente. Les méthodes classiques ne sont pratiquement jamais suivies à la lettre. Les ingénieurs d'application sont souvent amenés à les adapter pour pouvoir les appliquer dans le contexte de leurs projets [HIDD94]. De plus, les méthodes ne sont pas toujours suffisamment flexibles pour être facilement modifiées et adaptées. Les ingénieurs de méthodes ne cessent pas de se poser une question essentielle : comment construire une méthode qui soit assez flexible pour être facilement adaptée à la situation spécifique de chaque projet. Par conséquent, la discipline de l'ingénierie des méthodes est toujours d'actualité. Elle ne cesse pas d'évoluer et de proposer des nouvelles techniques de construction des méthodes. C'est ainsi qu'au début des années 90 naît une nouvelle approche de construction des méthodes *l'ingénierie des méthodes situationnelles* qui est définie dans [WELK92a] comme *"la discipline visant à construire et à adapter une méthode de développement de SI et les outils associés à chacun des projets spécifiques auxquels elle est appliquée"*. Selon les exigences de l'ingénierie des méthodes situationnelles, une méthode doit permettre la standardisation des activités, mais elle doit cependant rester suffisamment flexible pour prendre en compte la spécificité des situations rencontrées. A.F. Harmsen [HARM94] utilise le terme de flexibilité *contrôlée* pour définir cette exigence. Cette flexibilité est obtenue dans la majorité des approches d'ingénierie des méthodes, en définissant une phase préliminaire dans le projet consistant à caractériser la situation du projet de manière globale et, à l'aide de cette caractérisation, de composer une méthode adaptée à cette situation. De plus, le monde de la pratique des méthodes demande des processus rapides de construction de démarches définies "à la volée", pour s'adapter le mieux possible aux situations particulières de chaque projet. Le besoin de méthodes situationnelles a été introduit à ce propos [HARM94], [EURO94].

Ces constatations mettent en évidence le besoin de mieux comprendre la notion de méthodes, de savoir les décrire, d'être capable de les représenter, de les modéliser, mais aussi de les construire et de les utiliser dans différents contextes de projet.

L'objectif de ce travail est de survoler toutes les techniques de construction des méthodes d'ingénierie des SI en les comparant et les classant selon un cadre de référence qui est présenté à la section 2. La section 3 se concentre sur les différents objectifs de construction des méthodes tandis que la section 4 présente les différentes stratégies qui peuvent être suivies afin de construire une méthode. La section 5 conclut notre travail.

# 2. Cadre de référence

Afin d'atteindre notre objectif, c'est-à-dire de présenter toutes les techniques d'ingénierie des méthodes en les comparant les unes aux autres, nous utilisons un cadre de référence. Ce cadre de référence permet de classer les approches d'ingénierie des méthodes en terme de démarches de construction qu'elles proposent. Nous l'exprimons par un modèle de processus prenant en compte toutes les techniques de construction de méthodes. C'est un modèle de processus de type stratégique [ROLL99], [BENJ99] appelé *carte*. Il est basé sur deux notions : *intention* et *stratégie.* Le choix de ces deux concepts comme base du modèle de processus stratégique est fondé sur les hypothèses suivantes :

1. Le processus d'ingénierie des systèmes est orienté-intention. A tout moment, l'ingénieur d'applications est confronté à une intention, un objectif, qu'il désir réaliser.
2. Il y a en général plusieurs stratégies pour réaliser une même intention.

La carte identifie l'ensemble des intentions qui doivent ou peuvent être accomplies pour aboutir au produit escompté et l'ensemble des stratégies qui définissent comment accomplir ces intentions. Du point de vue structurel, une carte est un graphe directionnel dans lequel les nœuds

sont des *intentions* nécessaires à la réalisation du processus d'ingénierie concerné et les arcs entre les intentions sont des *stratégies* représentant les manières pour réaliser les intentions. Les stratégies connectent les intentions et permettent de progresser d'une intention vers une autre. Grâce à ces deux notions la carte permet d'abstraire plusieurs démarches d'ingénierie des méthodes dans un modèle de processus générique.

Notre cadre de référence est représenté à la Figure 1.

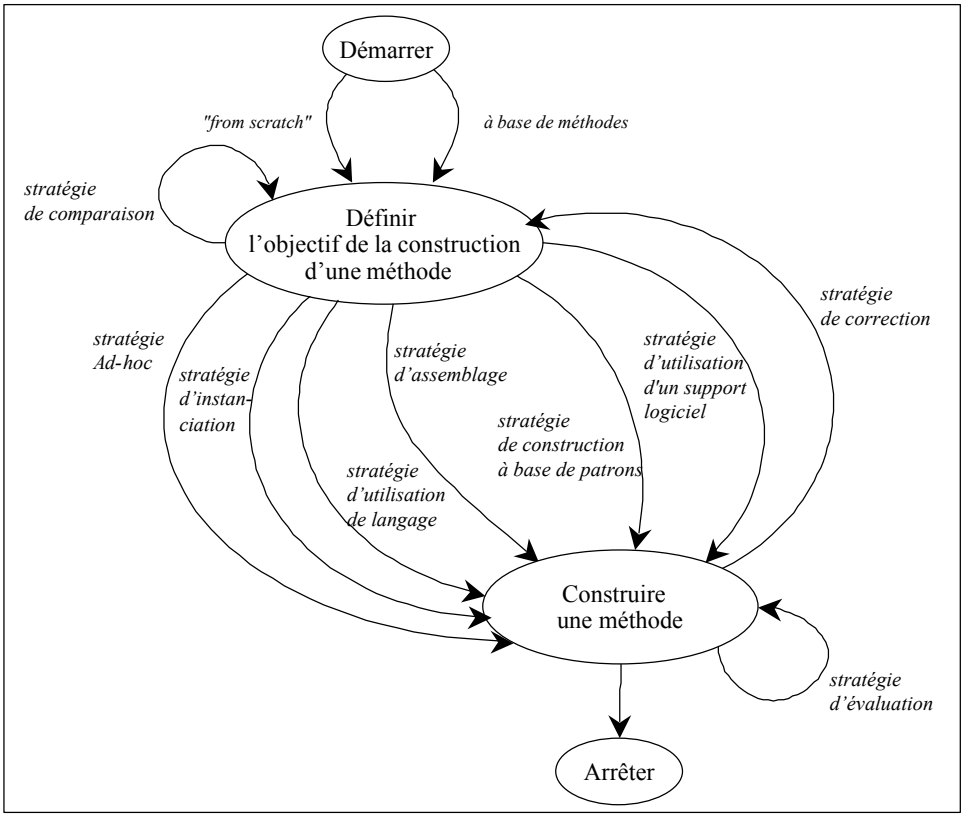


*Figure 1 : Cadre de référence*

Nous pensons que le processus de l'ingénierie des méthodes peut être exprimé en termes de deux intentions principales : tout d'abord il est nécessaire de *définir l'objectif de la construction d'une méthode*, ensuite, de *construire la méthode* répondant à cet objectif en sélectionnant une des techniques de construction.

L'intérêt d'utiliser le modèle de processus stratégique est de pouvoir proposer plusieurs stratégies différentes permettant de satisfaire ces intentions. Selon la source avec laquelle on démarre la construction d'une méthode on peut classer les objectifs de la construction en deux catégories : la première concerne les cas classiques où les méthodes sont créées sans aucune source particulière ("from scratch") tandis que la deuxième prend comme source des méthodes existantes. Suivant ce raisonnement nous avons défini deux stratégies relatives à l'accomplissement de l'intention *Définir l'objectif de la construction d'une méthode* : *"from scratch"* et *à base de méthodes*.

La réalisation de l'intention *Construire une méthode* dépend de la technique de construction appliquée. Un certain nombre de techniques de construction est disponible actuellement. Dans le domaine des systèmes d'information, les techniques de construction exploitent la notion de méta-modèle et utilisent deux techniques principales qui sont l'*instanciation* et l'*assemblage*. Dans le domaine du génie logiciel les principales techniques de construction utilisées sont basées sur les *langages de modélisation* et les *patrons*. Cependant, les techniques utilisées dans le passé dans

les deux domaines se basaient essentiellement sur l'expérience personnelle des ingénieurs de méthodes. Ces techniques étaient de ce fait de nature *"ad-hoc"*. Toutes ces techniques de construction sont représentées dans notre cadre de référence en termes des stratégies permettant de satisfaire l'intention *Construire une méthode*.

Une fois construites les méthodes nécessitent d'être validées. Pour cela nous proposons la stratégie d'évaluation qui prend en compte différentes techniques de validation.

La suite de cet article est divisée en deux sections correspondant chacune à une des intentions de notre cadre de référence. Dans ces sections nous présentons les stratégies permettant de satisfaire ces intentions et nous classons les approches d'ingénierie des méthodes suivant les stratégies qu'elles appliquent.

# 3. Définir l'objectif de la construction d'une méthode

Dans cette section nous analysons les objectifs que l'on cherche à atteindre dans le domaine de l'ingénierie des méthodes.

Les méthodes d'ingénierie de systèmes sont généralement supposées être indépendantes de la situation de leur application. Cependant, l'existence d'une multitude de méthodes montre que chaque méthode a des avantages et des désavantages relatifs au domaine du problème. En outre, l'expérience sur l'usage des méthodes montre que les concepteurs de systèmes adaptent et modifient les méthodes en fonction de la situation et de leurs préférences personnelles. Les concepteurs peuvent avoir besoin de créer une nouvelle méthode à partir de rien ("from scratch"), de modifier (c'est-à-dire améliorer, compléter ou adapter) une méthode existante ou de réutiliser les parties de diverses méthodes et de les assembler pour créer la nouvelle méthode choisie.

## 3.1 "From scratch"

"From scratch" est une attitude classique dans la construction des méthodes. Plusieurs raisons différentes peuvent amener les ingénieurs à construire des nouvelles méthodes :

- L'expérience de conception dans un domaine particulier peut inciter le concepteur à créer sa propre méthode. Dans ce cas, la technique ad-hoc est le plus souvent utilisée. Toutefois, le concepteur peut également instancier un méta-modèle ou appliquer un langage de modélisation existant afin de décrire la méthode.
- Le fait que le concepteur ne trouve pas une méthode adaptée à son problème peut l'encourager à créer une méthode qui lui convienne. De plus, cette dernière peut être généralisée si elle a fait ses preuves lors de son application dans le projet pour lequel elle a été créée.
- Les ingénieurs peuvent avoir des nouvelles idées pour améliorer certaines activités dans le développement des systèmes. Par exemple, le projet CREWS a démontré l'utilité des scénarios dans les processus de la découverte et de la validation des besoins de systèmes [ROLL98a] et a proposé quatre approches relatives à ces activités [ROLL98b], [HAUM98], [HEYMA98], [MAID98].

Tous ces objectifs peuvent être résolus en appliquant de différentes techniques de construction. L'ingénieur de méthodes peut construire la méthode de manière intuitive, autrement dit ad-hoc. Cette technique, très répandue à la naissance de la discipline de l'ingénierie des méthodes, est remplacée par des techniques plus formelles et plus systématiques comme la méta-modélisation et l'instanciation du méta-modèle proposé ou l'application d'un langage formel de modélisation.

## 3.2 A partir de méthodes existantes

La nouvelle tendance dans l'ingénierie des méthodes est basée sur la réutilisation des méthodes existantes. Par exemple, [PUNT96] définit l'ingénierie des méthodes comme *"une approche de construction des méthodes combinant différentes (parties de) méthodes pour développer une solution optimale au regard du problème donné"*. Plusieurs auteurs [SLOO93], [ROLL96], [RALY99b], proposent une structure pour une base de connaissance méthodologique réutilisable dans la construction des nouvelles méthodes. L'existence d'une telle base fait émerger une large panoplie de possibilités pour construire d'autres méthodes :

- La variété et la complexité croissante des domaines à informatiser nécessitent d'adapter les méthodes existantes à chaque fois que l'on développe un nouveau système. Une adaptation peut consister à :
  - étendre une méthode par une nouvelle démarche, si celle de la méthode n'est pas assez riche,
  - améliorer la qualité de la méthode par de nouvelles propriétés (gestion du temps par exemple),
  - compléter une méthode par un nouveau modèle, etc.
- La possibilité de construire des méthodes en réutilisant les différentes parties de plusieurs méthodes. Ceci permet la construction d'une méthode "à la volée".
- Un autre objectif de l'ingénierie des méthodes est d'assembler des composants de méthodes et de construire des outils CAME pour la ré-ingénierie des méthodes existantes. Elle peut consister à :
  - décrire la méthode d'une manière formelle en appliquant un langage particulier ou
  - reconstruire la méthode sous forme de composants réutilisables.
- L'expérience en appliquant une méthode peut faire apparaître des erreurs qui nécessitent d'être corrigées.

Tous ces objectifs peuvent être satisfaits en appliquant des nouvelles techniques de construction comme l'assemblage, l'application des patrons génériques ou l'utilisation des outils CAME. La plupart de ces techniques sont basées sur les techniques comme la méta-modélisation et l'application des langages formels.

## 3.3 Comparaison des méthodes

Pour évaluer les méthodes et pour aider dans la sélection des méthodes, des techniques de comparaison ont été proposées. Par exemple, [IIVA94] propose un cadre de référence pour comparer les méthodes orientées-objet qu'il applique à six méthodes. Ce cadre divise le processus de développement des systèmes d'information en trois niveaux : *organisationnel*, *conceptuel* et *technique*. Chaque niveau est analysé sous trois vues différentes : *structurelle, fonctionnelle* et *comportementale*. [SOUV97] s'inspire du travail de Iivary pour comparer des méthodes oriéntées-objet dans le cadre du projet TOOBIS. Dans ce travail le cadre de référence de Iivary est complété par des relations entre les trois niveaux et les trois vues.

[SONG92] et [HONG93] proposent des approches pour comparer des méthodes d'une manière systématique. Ces approches sont basées sur la méta-modélisation des méthodes à comparer. Elles proposent de construire tout d'abord les méta-modèles des toutes les méthodes à comparer en utilisant le même formalisme, ensuite elles se servent de ces méta-modèles pour comparer différents aspects de ces méthodes. [HONG93] par exemple, compare les méthodes selon trois

perspectives : les étapes d'analyse et de conception, les concepts et les techniques. Une représentation tabulaire est utilisée par les deux approches pour comparer les méthodes.

[ROSS95] propose une approche qui permet de mesurer la complexité des méthodes d'une manière systématique et automatisée. Cette approche utilise un ensemble de métriques permettant de mesurer d'un coté la difficulté de comprendre et d'apprendre la méthode causée par le nombre de différents concepts utilisés dans la méthode et d'un autre coté la complexité des produits causée par le nombre de propriétés des objets et des relations. Ces métriques peuvent être utilisées au moins pour deux objectifs : premièrement par l'ingénieur de méthodes pour valider les propriétés de la méthode, deuxièmement par l'utilisateur de méthodes pour sélectionner des méthodes.

# 4. Construire une méthode

Suivant notre cadre de référence (Figure 1), nous présentons dans cette section six stratégies de construction de méthodes : *ad-hoc*, *instanciation d'un méta-modèle, application d'un langage de modélisation*, *assemblage*, *application des patrons génériques* et *utilisation d'un outil CAME*, puis un résumé des approches d'évaluation des nouvelles méthodes.

## 4.1 Ad-Hoc

La technique de construction de méthodes *ad-hoc* est une technique traditionnelle qui est basée sur l'expérience acquise lors des développements de différents systèmes d'un domaine spécifique. Tant que cette expérience n'est pas formalisée et ne constitue pas une connaissance de base disponible pour les différents ingénieurs d'application, on peut dire que cette connaissance est le résultat d'une technique de construction ad-hoc. Ceci a deux conséquences majeures : la méconnaissance de la manière dont la méthode à été générée et sa dépendance au domaine d'expertise. Si la méthode doit être indépendante du domaine d'expertise, facile à appliquer et à modifier, il est alors nécessaire de sortir du cadre des techniques de construction basées sur l'expérience. Les techniques comme l'instanciation et l'assemblage favorisent la flexibilité et la modularisation des méthodes et facilitent la capitalisation des bonnes pratiques et l'amélioration des modèles existants.

## 4.2 Instanciation d'un méta-modèle

La technique de construction des méthodes par intanciation est basée sur la méta-modélisation. La méta-modélisation consiste à identifier les caractéristiques communes et génériques des différents modèles et à les représenter ensuite par un système de concepts génériques. Une telle représentation, appelée méta-modèle, permet de générer tous les modèles partageant ces mêmes propriétés. Cette technique de génération doit être définie de telle manière qu'elle produise le modèle désiré. Pour résumer, les deux problèmes suivants doivent être résolus :

- l'identification d'un système de concepts génériques inter reliés,
- la définition des techniques d'instanciation.

Le premier problème est résolu par la définition d'un *méta-modèle de méthodes* alors que le second est résolu par la dérivation des modèles de ce méta-modèle à travers son *instanciation.*

### *4.2.1 Méta-modélisation*

A ce jour on peut dire que les recherches les plus intensives dans le domaine de l'ingénierie des méthodes ont été menées dans le domaine de la méta-modélisation. On peut même dire que la méta-modélisation est le fondement de l'ingénierie des méthodes.

Plusieurs méta-modèles de méthodes ont été proposés. La plupart d'entre eux prennent en compte les deux perspectives de modélisation : le produit et le processus [SAEK93], [PLIH96], [PRAK99]. La perspective produit définit les structures des produits qui sont construits lors de l'application de la méthode tandis que les activités réalisées pour construire ces produits sont modélisées dans la perspective processus de la méthode.

Un certain nombre de méta-modèles de produit sont basés sur le modèle sémantique proposé dans [HULL87]. Mais les plus utilisés sont les modèles basés sur les formalismes de ER et de NIAM. Toutefois, il est difficile de représenter les contraintes et les structures hiérarchiques des méthodes en utilisant ces formalismes. Des extensions de ER proposées dans [SORE88], [WELK92b], [SMOL91] cherchent à améliorer le pouvoir d'expression en prenant en compte les contraintes d'intégrité et la représentation des objets complexes. Les méta-modèles de produit basés sur les formalismes de NIAM [BOMM91], [HOFS93b], [HOFS93c] aboutissent à des résultats similaires mais sont fondés sur une base plus formelle que les précédents. Un autre formalisme de méta-modélisation a été adapté du langage de modélisation orienté-objet appelé Object-Z dans [SAEK94a]. [AHIT87] propose un méta-modèle formel qui voit un système d'information comme un flux de données qui progresse d'un état vers un autre.

L'aspect processus dans la méta-modélisation est souvent développé séparément de celui de produit. [DOWS88] distingue trois catégories dans la modélisation des processus : *orienté-activité, orienté-produit et orienté-decision*. Nous y ajoutons deux nouvelles catégories récentes : les modèles de processus *contextuels* et les modèles de processus *stratégiques*.

Les méta-modèles de processus comme *Cascade* [ROYC70], *Spirale* [BOEH88], *Hiérarchique en spirale* [IIVA90] ou *Fontaine* [HEND90] font partie de la catégorie des modèles *orientés-activité*. Ces méta-modèles permettent de modéliser le processus par un ensemble de phases de haut niveau organisées en un processus séquentiel. Ils s'adressent donc à des processus tactiques dont l'objectif est de planifier les étapes à suivre.

Les modèles de processus orienté-produit représentent le processus de développement à travers l'évolution du produit qui en est le résultat. Ils ne mettent plus en avant les activités du processus, mais le résultat de ces activités et le produit résultant. Le méta-modèle de processus VIEWPOINTS [FINK90] appartient à cette catégorie. Dans la même catégorie le méta-modèle de processus de [HUMP89] permet de visualiser les processus par des diagrammes de transition d'états du produit en construction. [TOMI89] et [AKMA90] voient les processus comme l'évolution du méta-modèle du produit de conception.

Les modèles de processus *orientés-decision* expriment les transformations de produit comme conséquences de décisions. Les méta-modèles de processus proposés dans les projets IBIS [POTT89] et DAIDA [JARK92] appartiennent à cette catégorie. De tels modèles sont sémantiquement plus riches que les précédents car ils permettent d'expliquer non seulement comment le processus se déroule mais aussi pourquoi les actions sont exécutées.

On classe le méta-modèle de processus proposé par le projet NATURE [ROLL95], [PLIH96], [JARK99] dans la catégorie des processus *contextuels*. Selon ce méta-modèle, à tout moment le comportement de l'ingénieur d'application est conditionné par un contexte particulier : il se situe dans une situation particulière définie par l'état actuel du produit en construction où il a une intention à réaliser. Cette intention fait partie de l'objectif général - construire le produit.

Récemment une toute nouvelle catégorie de processus est apparue proposée dans [ROLL99] et [BENJ99]. Il s'agit d'un nouveau méta-modèle de processus appelé *Carte*. Nous avons appelé cette catégorie *processus stratégique* à cause de son principe de modélisation des processus en termes d'*intentions* d'ingénierie à réaliser afin de construire le produit et de *stratégies* à suivre

pour réaliser ces intentions. Ce méta-modèle a déjà fait ses preuves dans le projet CREWS (voir la note à la section 3.1).

La naissance d'un nouveau domaine dans l'ingénierie des méthodes, l'ingénierie des méthodes situationnelles, a stimulé la parution de méta-modèles permettant de représenter des méthodes sous forme modulaire. [ROLL96b] a proposé une structure pour un composant de méthode. Dans [PRAK97] et [PRAK99] une méthode est vue comme un ensemble de blocs de méthode, chaque bloc incorporant l'aspect produit et l'aspect processus. Le méta-modèle proposé dans [ROLL98a] et [RALY99b] permet de représenter toute méthode sous forme d'un assemblage de composants de méthodes. Chaque composant obtenu en appliquant ce méta-modèle devient un module réutilisable dans la construction des méthodes par assemblage de composants.

### *4.2.2 Instanciation*

La technique d'instanciation d'un méta-modèle de processus contextuel a été utilisée dans [ROLL94a], [ROLL94b], [ROLL96a]. L'ingénieur doit définir les instances de contextes et de relations entre les contextes qui composent le modèle en question. Cette technique a également été utilisée pour construire les bases de connaissance d'Atelier d'Ingénierie des Méthodes [KELL96], [HARM95b], [SISA96].

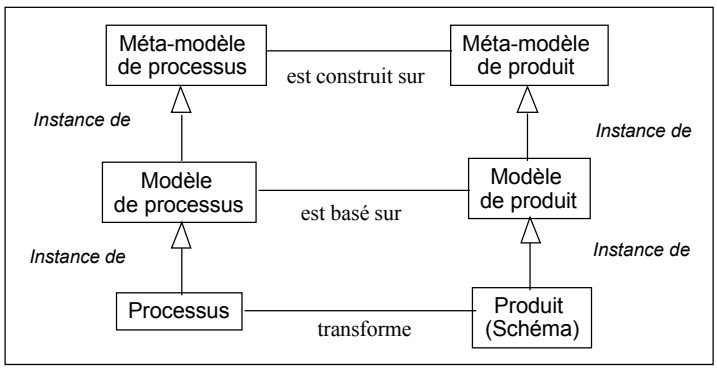


*Figure 2 : Les niveaux d'abstraction*

La technique de construction des méthodes par instanciation peut être résumée à la Figure 2. Au niveau le plus haut, se trouvent le *méta-modèle de processus* et le *méta-modèle de produit*. Ces deux méta-modèles contiennent des concepts génériques qui permettent respectivement, de décrire le *modèle de produit* et le *modèle de processus* d'une méthode spécifique. Tout élément d'un modèle de produit est instance d'un concept du *méta-modèle de produit*. Le *méta-modèle de produit* définit les concepts nécessaires pour caractériser une situation du produit. De même, tout élément du *modèle de processus* est instance d'un concept du *méta-modèle de processus*. En généralisant, nous dirons qu'un *modèle de produit* est construit par instanciation du *méta-modèle de produit* et qu'un *modèle de processus* est construit par instanciation du *méta-modèle de processus*. Les concepts du modèle de processus font référence aux concepts du modèle de produit. Au niveau d'une application réelle, l'ingénieur d'application suit un *processus*. L'exécution de ce processus est contrôlée par une instance du *modèle de processus* qui prescrit la marche à suivre. Le *processus* exécuté transforme un *produit* dont les éléments sont des *instances* du *modèle de produit*.

### *4.2.3 Bilan*

Les avantages d'utilisation de la technique méta-modélisation - instanciation sont nombreux :

- Tout d'abord cette technique aide à définir un certain nombre de modèles.

- L'exploitation du méta-modèle permet de définir des composants de méthodes de manière systématique.
- Cela oblige à chercher et à introduire dans un méta-modèle des solutions génériques aux problèmes traités. Les modèles ainsi dérivés hériteront des caractéristiques de la solution. A travers l'approche par instanciation, le problème crucial n'est plus celui des modèles mais celui du méta-modèle de processus. Cela signifie que la responsabilité de fournir un bon modèle de processus n'incombe plus au concepteur de modèles mais au concepteur du méta-modèle.
- La méta-modélisation est non seulement utile dans la construction des méthodes mais aussi dans la formalisation des méthodes mal définies [TOLV93], dans la comparaison des méthodes [HONG93], [ROSS96], dans la standardisation des méthodes [BOOC97], [OMG 97] et dans la définition des liens entre les méthodes d'ingénierie et les langages de programmation [HILL97].

## 4.3 Application d'un langage de modélisation

[BRIN90] fait l'hypothèse que tout langage conceptuel de modélisation puisse servir de langage de méta-modélisation. Divers langages de méta-modélisation d'application destinés à l'origine à d'autres domaines, tels que LOTOS [SAEK91] semblent démontrer la validité de cette hypothèse. Cependant d'autres auteurs ont des revendications contraires et expriment le fait qu'il y a besoin d'un langage de méta-modélisation et d'un environnement spécifiques. Par exemple, le langage COCOA [VENA93] a été développé et employé dans l'environnement MVIEWS [GRUN96]. Les exigences générales sur l'ingénierie des méthodes définies dans [KOTT84], [MART95], [WELK92a] soulignent la nécessité de constructions sémantiques assez riches pour modéliser la structure conceptuelle et les contraintes des méthodes.

La discipline de l'ingénierie des méthodes devenant de plus en plus mature, plusieurs écoles proposant des différents langages de représentation et de manipulation de méthodes ont vu le jour. Selon [HARM96], on peut distinguer quatre écoles principales. La première opte pour une approche orientée-données accentuant la représentation de l'aspect produit des méthodes. Elle inclut les langages d'ingénierie des méthodes comme GOPRR [SMOL92], [KELL96], PSM-LISA/D [HOFS93a], les diagrammes de structure objet de NIAM [BRIN90], [WIJE91], les modèles sémantiques de données [SOWA92] et ASDM [HEYM92]. Une deuxième école adopte l'approche orientée-objet. Les langages appartenant à cette école sont Telos [MYLO90], Mataview [SORE88] et ObjectZ [SAEK94]. La troisième école concerne les langages développés pour modéliser des processus de construction de logiciels et saisir les motivations de la conception. Les langages comme les Diagrammes de Structure de Tâches [WIJE91], [VERH95], HFSP [KATA89], [SONG92], ALF [BENA89] et MERLIN [EMME91] appartiennent à ce groupe. La dernière école concerne les langages, appelés "langages hybrides", qui prennent en compte les différents aspects de l'ingénierie des méthodes. Le langage MEL proposé dans [HARM95a], [HARM95b] et [BRIN95] fait partie de cette catégorie. MEL est un langage de la spécification et de la manipulation pour la construction des méthodes situationnelles.

Dans [HARM96] quatre langages (un de chaque catégorie) ont été choisis et comparés: Object-Z, MEL, GOPRR et HFSP. La conclusion de l'étude dit qu'il n'y a pas de langage universel pour l'ingénierie des méthodes. Les différents langages ont des propriétés communes et des propriétés complémentaires. Le choix du langage dépend de l'objectif à atteindre dans l'ingénierie des méthodes. Comme solution, [HARM96] proposent d'appliquer la technique d'assemblage sur les

langages d'ingénierie des méthodes eux-mêmes et de construire ainsi un langage adapté à la situation donnée à partir de divers fragments de langages existants.

Les langages de méta-modélisation nécessitent différentes formes de représentations. Ils utilisent traditionnellement des notations informelles comme le langage naturel ou des diagrammes avec des sémantiques informelles. Le fait que les méthodes utilisent souvent des représentations schématiques a influencé le développement visuel des langages de méta-modélisation. [HOFS93c] accentue l'importance de la représentation graphique lors de l'instanciation d'un méta-modèle afin d'obtenir une méthode et propose une approche de la formalisation d'une telle représentation. Des langages pour représenter les notations graphiques des méthodes, les connexions entre celles-ci et les contraintes graphiques sont proposés dans [SOMM87], [SMOL91], [PROT89], [HOFS92]. [KELL94] et [KINN94] ont investi dans l'utilisation des matrices comme moyen de représentation de la méta-modélisation. Les dernières recherches dans ce domaine étendent les représentations précédentes dans de nouveaux paradigmes multi-représentations. Prenons comme exemple, l'environnement MetaEdit+ qui permet de modéliser une méthode en utilisant un diagramme, une matrice ou un tableau et qui propose aussi un mode graphique pour questionner la base de méthodes [LIU95]. Les travaux présentés dans [OEI94], [OEI95] introduisent un langage formel pour modéliser les méthodes et les organiser dans une hiérarchie.

## 4.4 Assemblage

La technique de construction de méthodes par assemblage est la dernière-née du domaine d'ingénierie des méthodes situationnelles. Elle est fondée sur l'idée que l'on peut décomposer les méthodes existantes en composants (aussi appelés fragments, modules ou blocs) réutilisables indépendamment. La construction d'une nouvelle méthode consiste donc à sélectionner des fragments de différentes méthodes correspondant à la situation d'un projet en cours et à les assembler. Ceci fait apparaître les problèmes suivants :

- la redéfinition des méthodes existantes sous forme de composants,
- le stockage de ces composants dans une base de méthodes et
- le guidage dans la sélection et l'assemblage des composants.

[HARM94], [HARM95a], [BRIN98] proposent une approche d'assemblage à base des fragments de méthodes de différents niveaux de granularité. Afin de le distinguer parmi les autres approches nous lui avons donné le nom de *Fragments de méthodes.*

L'approche proposée dans [RALY99b], [RALY99a], [RALY01] permet de redéfinir les méthodes existantes sous forme des composants de méthodes et de les assembler suivant les exigences en cours. Nous appelons cette approche *Composants de méthodes*.

[ROLL96b] propose une structure pour représenter et stocker des composants de méthodes de différents niveaux d'abstraction (composant, patron, cadre de référence) et de différents niveaux de granularité (contexte, arbre). Nous avons appelé cette approche *Composants contextuels.*

[PRAK97], [PRAK99] proposent un méta-modèle pour représenter des méthodes sous forme modulaire. Les modules sont appelés des blocs de méthodes d'où le nom de l'approche *Blocs de méthodes.*

[SONG97] propose une approche d'intégration des méthodes de conception qui est basée sur l'adaptation des méthodes existantes. Elle permet de compléter une méthode par une fonctionnalité complémentaire prise dans une autre méthode et/ou d'améliorer sa qualité en lui associant des propriétés issues d'autres méthodes.

[PUNT96] propose une approche appelée MEMA comportant un méta-modèle pour représenter les fragments de méthodes ainsi qu'un processus de sélection et d'assemblage de fragments afin de construire une méthode pour un projet concret.

L'approche proposée dans [SLOO93] et [SLOO96] se base surtout sur la caractérisation des projets en utilisant des facteurs de contingence.

Nous analysons maintenant comment ces approches réagissent aux problèmes dont la liste a été donnée plus haut dans ce paragraphe.

### *4.4.1 Structure de composant*

Dans la plupart des cas la définition des composants est basée sur la technique de méta-modélisation. Ceci permet de résoudre le problème de représentation des méthodes sous forme de composants. La plupart des approches existantes ne prennent en compte que la structure des composants et ne se préoccupent pas du processus de reconstruction des méthodes sous forme de composants.

Pratiquement chaque approche propose plusieurs types de composants qu'elle classifie selon différentes typologies. Par exemple, [BRIN98] classifie les composants selon trois dimensions : la *perspective*, l'*abstraction* et la *granularité*, que nous utilisons comme fondement de classification.

#### *4.4.1.1 Dimension de perspective*

La dimension de *perspective* considère les méthodes du point de vue du *produit* et de celui du *processus*. Certaines approches font une séparation entre la perspective produit et la perspective processus et proposent deux types de composants : ceux du produit et ceux du processus. Dans cette catégorie nous retrouvons l'approche *Fragments de méthodes* qui définit des fragments de produit et des fragments de processus. Les premiers représentent des modèles, des diagrammes, des documentations tandis que les seconds représentent des étapes, des activités des tâches à réaliser. Chaque fragment de processus a un fragment de produit associé et inversement.

L'approche proposée dans [SONG97] fait également la séparation entre la perspective produit et la perspective processus. Dans la première elle propose deux types de composants appelés *modèle artefact* et *composant de modèle* appartenant à des niveaux de granularité différents. Dans la seconde elle propose deux types de composants appelés *processus* et *action* qui appartiennent également à deux niveaux de granularité différents.

D'autres approches s'appuient sur le fait que la perspective processus ne peut pas être séparée de celle de produit, que le processus est toujours basé sur le produit qu'il manipule. Par conséquent, elles affirment qu'il est préférable de coupler les deux perspectives dans le même composant de méthode. Dans cette catégorie on trouve les *Composants contextuels* [ROLL96b], l'approche *Blocs de méthodes* [PRAK99], l'approche MEMA [PUNT96] ainsi que Composants de méthodes [ROLL98c], [RALY99b]et [RALY01].

#### *4.4.1.2 Dimension d'abstraction*

Selon [BRIN98], la dimension d'*abstraction* est constituée de deux niveaux : *conceptuel* et *technique*. Le niveau conceptuel comporte des fragments de méthodes de conception tandis que le niveau technique propose des fragments qui représentent des parties opérationnelles de méthodes, c'est-à-dire des outils.

La plupart des approches ne sont considérées que dans un des niveaux d'abstraction – celui de conception. Parmi ces approches figurent *Blocs de méthodes*, MEMA*, Composants contextuels*.

L'approche *Fragments de méthodes*, au contraire, prend en compte les deux niveaux d'abstraction en précisant que chaque fragment technique correspond à un fragment conceptuel.

[ROLL96a] et [ROLL96b] ont un autre point de vue sur les différents niveaux d'abstraction de composants. Selon ce niveau d'abstraction, le composant de méthode est réutilisé en tant que tel ou doit être instancié pour être assemblé dans la méthode en construction. L'approche [ROLL96b] propose trois types de composants : *cadre de référence*, *patron de construction* et *composant de méthode*. Le *cadre de référence* est un composant qui formalise d'une manière abstraite la connaissance commune à plusieurs méthodes. Le *patron* modélise le comportement commun dans la construction des méthodes. Il est générique dans le sens qu'il est utilisé par un ingénieur de méthodes dans le processus de construction de chaque nouvelle méthode. Par conséquent, le composant *patron* est plus abstrait que le *cadre de référence.* Ces deux termes ont été choisis par analogie avec les approches de réutilisation dans le domaine orineté-objet. Les patrons dans ce domaine sont définis comme des solutions aux problèmes génériques qui apparaissent dans plusieurs applications (voir la section 4.6) tandis que les cadres de référence sont dépendants du domaine d'application [WIRF90].

#### *4.4.1.3 Dimension de granularité*

La dimension de *granularité* est la plus importante et la plus discriminante des classifications proposées. Elle est basée sur la décomposition des méthodes en différents niveaux de détail. Par exemple, du point de vue du processus une méthode peut être organisée en étapes qui, à leur tour, sont structurés en activités qui sont décomposées en actions etc. Une décomposition similaire peut également être appliquée sur le produit de la méthode car celle-ci peut être représentée par un ensemble de modèles ayant différents diagrammes, qui à leur tour sont décomposés en concepts, etc.

L'approche *Fragments de méthodes*, propose cinq niveaux de granularité de fragments appelés: *méthode*, *étape*, *modèle*, *diagramme* et *concept*. Le niveau *méthode* s'adresse à des méthodes entières. Toute méthode d'ingénierie de systèmes, comme par exemple OMT, OOSE, est vue comme un fragment de méthode. Le niveau *étape* s'adresse à des segments dans le cycle de vie d'un système d'information. Par exemple, *Analyse de domaine*, *Outil CASE*, *Rapport technique de la conception d'un système*, sont des fragments de niveau étape. Un fragment de type *modèle* prend en compte une perspective d'un système d'information comme *Modèle de donnés*, *Modèle d'interface utilisateur* etc. Un fragment de type *diagramme* correspond à une représentation possible d'un composant de type modèle comme un *Diagramme d'objets* ou un *Diagramme de classes*. Le niveau *concept* s'adresse à des concepts et à des associations qui existent entre eux dans le niveau *diagramme* d'une méthode ainsi que les manipulations qui peuvent être faites avec eux. *Entité*, *Relation entre deux entités* et *identifier une entité* sont des exemples des fragments de niveau concept.

L'approche [SONG97] divise tous les composants de méthode en deux niveaux de granularité qu'elle appelle : le *niveau haut* et le *niveau bas*. Le niveau haut correspond au niveau des modèles de méthodes tandis que le niveau bas représente les différents éléments de ces modèles. De plus, cette approche sépare non seulement les notions de produit et de processus, mais en outre leurs représentations et leurs propriétés sont considérées à part dans des composants spécifiques. Le niveau *haut* propose cinq types de composants : *modèle artefact, propriété, représentation* et *processus.* Les *modèles artefacts* représentent des structures, des modèles (par exemple le modèle objet de la méthode OMT), les *propriétés* considèrent les caractéristiques des modèles artefacts, les *principes* décrivent des règles devant être suivies en construisant des artefacts (par exemple les principes d'abstraction et d'encapsulation dans les méthodes orientées-objet), les *représentations* concernent le moyen d'expression des modèles artefacts (par exemple

le diagramme d'objet ou le diagramme de flux de donnés) et les *processus* décrivent des étapes que le concepteur doit utiliser dans le développement d'un système en appliquant un modèle artefact. Les composants appartenant au niveau *bas* sont partagés en six catégories : *modèles composants, critères, directives, mesures, notations* et *actions*. Les *modèles composants* sont des éléments d'un *modèle artefact* (par exemple, une classe dans le modèle objet de la méthode OMT), les *critères* sont des règles que le concepteur doit appliquer pour décider si un artefact est une instance d'un modèle composant (par exemple, l'ingénieur d'application devrait utiliser ces règles pour décider si l'artefact "porte" peut être une classe dans le système de contrôle d'ascenseur), les *directives* sont des stratégies, des heuristiques ou des techniques concrètes pour identifier et décrire les artefacts, les *mesures* concernent les aspects des artefacts qui peuvent être mesurés, les *notations* sont des parties de la représentation utilisée pour exprimer les artefacts (par exemple une boite pour représenter les objets dans le diagramme objet de OMT), les *actions* sont des étapes pour développer les artefacts (une action peut créer, modifier, utiliser ou évaluer un artefact). Les différents composants appartenant au même niveau sont reliés entre eux par des liens d'association tandis que les composants de niveaux différents sont reliés par des liens de composition.

Van Slooten ([SLOO93], [SLOO96]) propose deux types de composants de méthodes appartenant à deux niveaux de granularité, appelés la *carte de routes* et le *fragment de méthode*. Une *carte* de *routes* est un plan représentant une stratégie de développement. Il est composé d'activités de développement et de produits concernant le développement d'un système ainsi que la gestion du projet. Par exemple, une carte de routes peut représenter une stratégie de planification du projet, une stratégie de fourniture du projet à l'utilisateur (en entier, incrémentale, évolutive), une stratégie de réalisation, une stratégie d'organisation du projet, une stratégie de la gestion du projet etc. Un *fragment de méthode* est une partie cohérente d'une méthode, d'une technique ou d'un outil destiné au développement d'un système ou à la gérance d'un projet. Les fragments de méthodes peuvent être incorporés dans la carte de routes pour établir une approche complète pour un projet.

De ce point de vue, l'approche *Composants de méthodes* distingue deux types de composants : atomiques et agrégats. Les composants atomiques peuvent être reliés par des liens de composition, d'alternative ou d'imbrication afin de construire des composants plus importants que l'on appelle des agrégats.

### *4.4.2 Stockage de composants*

Le deuxième problème concernant la technique d'assemblage est le stockage de composants et l'accès aux composants. La plupart des approches d'assemblage utilisent la notion de base de méthodes pour stocker les composants. On trouve une base de méthodes dans les approches *Fragments de méthode*, *Composants contextuels*, MEMA, celle de Van Slooten ainsi *Composants de méthode*. Toutefois la plupart des approches ne précisent pas quelle est la structure de cette base ni comment les composants peuvent y être atteints. [SONG97], au contraire, ne parle pas d'une base de méthodes et ne dit pas comment décrire ni comment stocker les composants. L'approche *Blocs de méthodes* propose un formalisme pour décrire les blocs de méthodes mais ne dit pas non plus comment les stocker.

Les différents types de composants sont en général stockés dans la même base. Par exemple, [SLOO93] propose de stocker les cartes de routes dans la même base que les fragments de méthodes. De la même manière les cinq types de fragments (voir la section précédente) de l'approche *Fragments de méthodes* sont également stockés dans la même base.

Le même principe est appliqué dans les approches *Composants contextuels* et *Composants de méthodes*. Ces deux approches divisent la base de méthodes en deux niveaux : le premier est

appelé *connaissance méthode* et le deuxième *méta-connaissance*. Le niveau *connaissance méthode* comporte la connaissance qui est effectivement réutilisable, c'est-à-dire les composants eux-mêmes tandis que le niveau de *méta-connaissance* comporte l'information nécessaire à la sélection et la réutilisation des composants. Dans le domaine de la réutilisation ces deux niveaux sont appelés la *connaissance réutilisable* et la *connaissance pour la réutilisation*.

Pour représenter la méta-connaissance l'approche *Composants contextuels* ainsi que *Composants de méthode* utilisent la notion de *descripteur* qui permet de personnaliser les composants. Ceci facilite l'accès aux composants ainsi que leur sélection dans la base de méthodes. Chaque composant de méthode a un descripteur qui décrit son contexte de réutilisation. Un descripteur relie la situation dans laquelle le composant est pertinent à l'intention qu'il permet de satisfaire dans le processus d'ingénierie de systèmes. La situation de l'approche *Composants contextuels* détermine des caractéristiques qu'un projet doit avoir pour que le composant soit applicable dans son développement. Ces caractéristiques sont basées sur les résultats obtenus dans le projet EUROMETHOD [FRAN94]. La situation dans l'approche *Composants de méthode* fait référence aux domaines dans lesquels le composant est applicable et les activités de conception dans la réalisation desquelles le composant peut participer. L'intention dans les deux approches représente une intention d'ingénierie de systèmes qui peut être réalisée en appliquant le composant.

Une nouvelle manière de présenter les composants de méthodes pour qu'ils soient accessibles à un public plus large est de les offrir sous forme de bibliothèques électroniques accessibles par Internet. Ceci à été proposé dans [RALY99b] et [BRIN00]. [BRIN00] souligne que la technologie Internet apporte des nouvelles perspectives dans la création et l'utilisation des méthodes. Tout d'abord, la disponibilité des méthodes sur Internet garantit à tout ingénieur d'application un accès facile à ces méthodes. L'évaluation des méthodes est plus rapide car grâce à Internet les utilisateurs de méthodes peuvent envoyer leurs appréciations, remarques et suggestions aux créateurs de méthodes ainsi que les futurs utilisateurs. Les méthodes peuvent utiliser différentes technologies multi-média pour les formations sur l'application de la méthode en utilisant des diapositives, des vidéos, des animations etc. La présentation hyper-texte des méthodes est plus attractive que la description sur le papier. On peut accéder rapidement à des parties de la description qui nous intéressent au moment donné.

### *4.4.3 Guidage dans la sélection et l'assemblage de composants*

Le troisième problème que la stratégie de construction de méthodes par assemblage tente de résoudre, est l'existence d'un processus guidé qui permet de sélectionner des composants répondant aux besoins du projet en cours et d'assembler ces derniers afin d'obtenir une méthode adaptée à la situation de ce projet. Comme nous avons montré dans les sections précédentes, plusieurs modèles ont été proposés pour définir les composants de méthodes. Malgré cela, le processus concernant leur sélection et leur assemblage reste un domaine de recherches peu évolué. Les approches comme *Composants contextuels* et *Blocs de méthodes* se limitent à la définition et à la représentation des composants. D'autres approches comme *Fragments de méthodes, MEMA*, [SONG97] ou celle de Van Slooten ne proposent pour le moment que des processus très génériques qui n'offrent aucun guidage à l'ingénieur de méthodes concernant la sélection et l'assemblage des composants.

Dans les approches *Fragments de méthodes,* MEMA ou celle de Van Slooten la première étape dans la construction d'une méthode situationnelle concerne la caractérisation du projet en cours. Ceci est basé sur la définition des facteurs de contingence qui peuvent être identifiés à l'aide des interviews, des questionnaires et d'autres techniques d'acquisition de la connaissance. MEMA, par exemple, utilise un cadre de référence défini pour ce propos. [SLOO96] définit une liste de

facteurs de contingence comme l'importance du projet, son impact sur l'organisation existante, la pression du temps, l'expérience de l'équipe de développement, la taille, les relations avec d'autres systèmes, la dépendance des autres projets, la clarté, la stabilité, la complexité, l'innovation etc. Selon les auteurs de ces approches, les facteurs de contingence sont utilisés ensuite pour sélectionner les composants appropriés. L'approche MEMA par exemple, a un processus qui permet de déterminer la correspondance entre la caractérisation du projet à l'aide de leur cadre de référence et les descriptions des composants stockés dans une base de méthodes. Cependant, les autres auteurs restent assez flous dans leurs explications concernant l'évaluation de l'ensemble de facteurs de contingence et leur impact sur le choix d'un composant de méthode plutôt qu'un autre.

Le processus d'assemblage des composants sélectionnés reste également assez flou dans les approches MEMA et celle de Van Slooten. Par contre, l'approche *Fragments de méthodes* propose un processus assez détaillé guidant l'assemblage des fragments de méthodes sélectionnés au préalable. Ce processus consiste à assembler les fragments de produits et les fragments de processus correspondants. L'assemblage de deux fragments de produit consiste à identifier un ou plusieurs liens entre différents concepts des deux fragments. Des nouveaux concepts peuvent être créés pour permettre la connexion entre les fragments de produit. En ce qui concerne l'assemblage des fragments de processus, des liens de précédence entre les différentes activités des composants doivent être établis. Des contraintes d'assemblage sous forme des règles formalisées sont proposées pour assurer la cohérence et la complétude de l'assemblage obtenu. Cependant, cette approche se limite à l'assemblage des fragments complémentaires qui n'ont pas d'éléments communs.

[RALY01] propose un processus d'assemblage qui prend également en compte l'assemblage des composants qui se recouvrent partiellement, par exemple des composants qui permettent de satisfaire le même objectif mais qui proposent des solutions différentes. Le nom d'*intégration* est attribué à cette technique. L'intégration de tels composants permet d'obtenir un nouveau composant encore plus riche que les deux initiaux. De plus, ce processus d'assemblage est fondé sur des opérateurs formalisés et offre également des règles de validation de la cohérence de l'application de ces opérateurs et de la complétude du résultat obtenu.

[SONG97] parle d'adaptation et d'amélioration des méthodes existantes plutôt que de la construction d'une nouvelle méthode. Il propose deux types d'assemblage qu'il appelle *basé fonction* et *basé qualité*. Le premier permet d'ajouter des nouvelles fonctionnalités dans une méthode tandis que le second permet d'améliorer la qualité de la méthode, d'augmenter son efficacité en lui ajoutant de nouvelles propriétés. Les deux types d'assemblage concernent les deux niveaux de composants (voir la section 4.4.1.3). Les processus d'assemblage restent cependant informels, proposant uniquement des raisons, des heuristiques, des exemples mais pas de guidage systématique. L'approche de [RALY01] prend également en compte ces deux objectifs d'assemblage en proposant un processus guide à chaque fois.

## 4.5 Utilisation d'un outil CAME

Une croissance importante du nombre de méthodes d'ingénierie de systèmes et de leurs environnements du support a influencé l'apparition d'une nouvelle technique d'ingénierie des méthodes appelée Atelier d'Ingénierie des méthodes (AIM) (ou CAME - Computer Aided Method Engineering) [SLOO93], [KUMA92]. L'ingénierie des méthodes dans ce domaine est définie selon [HEYM93] comme un processus discipliné pour construire, évaluer ou modifier une méthode par le moyen de spécification des composants de méthode et des relations entre eux. Si les ateliers de génie logiciel (AGL) sont des outils qui aident dans le développement de

systèmes d'information, les outils CAME visent à aider dans le développement de méthodes. L'objectif d'un outil CAME est d'aider l'ingénieur de méthodes à construire de nouvelles méthodes et à modifier des méthodes existantes. Selon [HARM94], un outil CAME doit offrir les fonctionnalités suivantes :

- *La définition et l'évaluation de facteurs de contingence*. Pour pouvoir choisir des composants de méthodes correspondants aux exigences du projet en cours, des règles et des facteurs de sélection propres à ces composants doivent être définis par l'ingénieur de méthodes. Pour un profil de projet donné et une base de méthodes, l'outil CAME sélectionne et assemble une méthode appropriée.
- *Le stockage de composants de méthodes*. Afin de pouvoir manipuler les composants de méthodes avec un outil CAME, il est nécessaire de les stocker dans une base de méthodes. Les nouvelles méthodes peuvent être ajoutées dans la base, les composants peuvent être modifiés ou éliminés de la base.
- *L'extraction et l'assemblage des composants*. L'ingénieur de méthodes doit avoir la possibilité de sélectionner des composants dans la base de méthodes. Un langage d'interrogation pour accéder au contenu de la base a besoin d'être défini. La connaissance permettant le développement d'une nouvelle méthode doit être disponible.
- *La validation et la vérification de la méthode construite*. L'outil CAME devrait non seulement aider à réaliser les tâches d'assemblage et de sélection mais aussi à vérifier la méthode résultante. L'outil devrait donc incorporer des directives permettant d'assurer l'exactitude de la méthode.
- *L'adaptation de la méthode obtenue*. L'outil CAME devrait offrir la fonctionnalité de l'adaptation dynamique de la méthode. Il doit également permettre de compléter la base de méthodes compte tenu de l'expérience acquise.

L'environnement d'un outil CAME doit être composé de deux sous-environnements: celui de l'ingénierie d'applications et celui de l'ingénierie des méthodes. Le lien entre les deux sous-environnements peut être établi grâce au concept de la *base de méthodes* qui est utilisée par les deux. L'environnement d'ingénierie d'applications doit permettre d'exécuter les fonctionnalités de l'outil CAME comme la définition et l'évaluation des facteurs de contingence, la sélection des composants dans la base de méthodes, l'assemblage de composants et l'évaluation de la méthode obtenue. L'environnement d'ingénierie des méthodes doit permettre de définir et d'améliorer les composants de méthodes.

Même si aujourd'hui les fonctionnalités qu'un l'outil CAME devrait fournir sont bien définies, des travaux considérables doivent encore être réalisés pour implémenter ces fonctionnalités. Cependant, un nombre de produits meta-CASE et de prototypes ont été développés en réalisant partiellement certaines fonctionnalités. Nous pouvons citer pour exemple, MEET [HEYM93] MetaEdit+ [KELL96], Decamerone [HARM95], Mentor [SISA96] et MERU [PRAK99]. Decamerone est dans un état préliminaire de développement mais permet l'assemblage de fragments de processus et de produit qui ont été sélectionnés à l'aide des facteurs de contingence du projet. Cependant, il ne fournit pas de guidage ni pour l'ingénierie des méthodes ni pour l'ingénierie d'applications. L'outil MEET propose un modèle de représentation de méthodes qui peut être utilisé pour standardiser, comparer et intégrer des différentes méthodes. MetaEdit+ inclut un nombre d'instanciations principalement sur les aspects produit d'une vingtaine de méthodes. L'accent, dans l'outil Mentor, a été mis sur l'unification des aspects de produit et de processus des méthodes ainsi que sur le guidage et le support qui peut être fourni pour l'ingénieur de méthodes et l'ingénieur d'applications. MERU consiste en deux parties : la première aide à

formuler les exigences sur la méthode à construire, la deuxième génère la méthode. Les exigences sur la méthode sont exprimées en utilisant un méta-modèle spécifique appelé Method View. La génération de la méthode utilise la technique d'assemblage. La notion de composant de méthode est définie pour permettre ceci.

L'utilisation de la méta-modélisation associée à un environnement CAME dans l'ingénierie des méthodes permet de réaliser deux objectifs simultanément : premièrement nous pouvons comparer les méthodes de manière analytique, deuxièmement nous pouvons placer les méthodes dans un environnement où elles ont une plate-forme qui permet de les stocker et de les représenter. Les premiers travaux dans ce domaine ont été concentrés sur la définition des langages de méta-modélisation [BRIN90], [TOLV93] ou sur la construction des environnements pour eux [CHEN91], [SORE88], [SMOL91].

## 4.6 Utilisation des patrons génériques

Le concept de *patron* est né grâce au travail de Alexander et al. qui a proposé d'utiliser celui-ci dans le domaine de l'architecture des bâtiments [ALEX79]. Puis, ce concept a été tout d'abord emprunté dans le domaine du développement de logiciels et surtout dans celui des approches orientées-objet. Il a été appliqué dans la programmation de logiciels [BECK97], [BUSH96], la conception de systèmes [COAD96], [GAMM94], [COPL95], [VLIS96], [RIEU99] la modélisation des données [HEY96] et l'analyse de systèmes [FOWL97].

Alexander définit un patron comme "une solution à un problème qui se produit souvent dans notre environnement décrite d'une telle manière que l'on peut utiliser cette solution un million de fois sans pour autant faire la même chose deux fois". L'essence d'un patron est dans le fait qu'il propose une solution réutilisable dans toute situation où le problème concerné par le patron apparaît. Grâce à son pouvoir de réutilisation, ce concept a été introduit récemment dans le domaine de l'ingénierie des méthodes [ROLL96a], [ROLL96b], [ROLL00a], [ROLL00b], [DENE01].

Le travail proposé dans [ROLL96a] et [ROLL96b] introduit la notion de patron comme un moyen pour modéliser le comportement commun dans la construction des méthodes. [ROLL96a] définit plusieurs patrons de construction référencés par les verbes d'intentions qu'ils permettent de satisfaire dans le processus de construction d'une méthode, comme par exemple *Identifier*, *Décrire*, *Construire*, *Définir*, *Valider* et *Affiner*. Ces patrons peuvent être appliqués dans la construction de plusieurs méthodes. Ces patrons sont génériques dans le sens où ils peuvent être utilisés par un ingénieur de méthodes dans le processus de construction de plusieurs méthodes.

Le projet ELEKTRA propose un langage à base de patrons, appelé *patrons de décision*, pour décrire des meilleures expériences dans la gestion de changements dans des systèmes gérant des processus d'organisation des entreprises. Une méthode pour définir les patrons est également définie permettant d'organiser et d'affiner les patrons existants et de découvrir de nouveaux patrons [ROLL00a], [ROLL00b].

[DENE01] propose une approche d'extension de méthodes existantes en utilisant des patrons génériques. De plus, cette approche comporte une technique d'organisation de ces patrons avec un guidage pour leur réutilisation ainsi qu'une technique de conception de nouveaux patrons d'extension à l'aide de méta-patrons.

## 4.7 Evaluation des méthodes

L'étape de la validation d'une nouvelle méthode est très importante surtout si la méthode est sensée être générique et indépendante d'un projet particulier. Plusieurs techniques de validation

des méthodes ont été proposées à ce jour. Pratiquement chaque nouvelle méthode, chaque nouvelle approche passe par l'étape de validation avant d'être diffusée. Le plus souvent, les méthodes sont évaluées en les appliquant pour modéliser des cas appartenant aux différents domaines d'application.

Une autre technique très répandue est basée sur l'utilisation des groupes de personnes qui ne connaissent pas la méthode et qui sont invitées à l'appliquer sur un cas, en général le même pour tout le monde. Un questionnaire suit le plus souvent la séance du travail pour évaluer la clarté de la méthode, son adaptabilité au problème, sa facilité d'utilisation, le temps nécessaire pour assimiler les notions utilisées etc.

Si la méthode a un support outillé, elle peut être évaluée en expérimentant les capacités de cet outil. Ceci peut être fait par les études des cas individuels ou groupés en appliquant l'outil correspondant. Les remarques des testeurs sont prises en compte dans la nouvelle version de l'outil. Mais aussi, elles peuvent faire apparaître des problèmes méthodologiques qui doivent être résolus avant la modification de l'outil.

Toute utilisation de la méthode sur un cas réel est également une sorte d'évaluation de celle-ci. Tout ingénieur d'application est invité à donner ses appréciations après avoir utilisé une méthode dans la conception d'un système réel. Son expérience, ses remarques et ses propositions sont cruciales pour l'évolution de la méthode.

# 5. Conclusion

Dans ce travail nous avons considéré la discipline d'ingénierie des méthodes comme une méthode elle-même. C'est une méta-méthode pour construire des méthodes d'ingénierie de systèmes. Nous avons étudié sa perspective processus en modélisant tout d'abord celle-ci par un modèle de processus stratégique.

Ce modèle nous a permis de caractériser les différentes approches d'ingénierie des méthodes à l'aide des intentions identifiées dans ce domaine et les stratégies pour satisfaire ces intentions.

En parcourant les stratégies de ce modèle nous avons identifié les approches qui nous semblaient suivre ces stratégies et nous les avons comparées les unes aux autres.

Ce parcours a permis de constater que le fondement de la plupart des techniques de construction, d'évaluation et de comparaison de méthodes est basé sur la méta-modélisation. De plus, les techniques de construction de méthodes rigides sont pratiquement remplacées par des techniques de construction dynamique à base des composants de méthodes et des patrons et proposant souvent un support outillé. Ces techniques permettent en général de construire des méthodes spécifiques aux projets de telle manière que, à son tour, elles peuvent être réutilisées dans la construction de nouvelles méthodes. Toutefois, des études empiriques de ces nouvelles techniques sont nécessaires afin d'évaluer leur efficacité sur des projets réels.

# 6. Références


[AHIT87] N. Ahituv, A metamodel of information flow: a tool to support information systems theory. Communications of the ACM 30 (9), pp. 781-791, 1987.

[AKMA90] V. Akman, P.J.W. ten Hagen, T. Tomiyama*, A fundamental and theoretical framework for an intelligent CAD System*. Computer Aided Design, 22 (6), 1990.

[ALEX79] C. Alexander, S. Ishikawa, M. Silverstein et al., *A Patron Language.* Oxford University Press, New York, 1997.

[BECK97] K. Beck, *Smalltalk : Best Practice patrons*. Volume 1: Coding, Prentice Hall, NJ, 1997.

[BENA89] K. Benali, N. Boudjlida, F. Charoy, J. C. Derniame, C. Godart, Ph. Griffiths, V. Gruhn, Ph. Jamart, D. Oldfield, F. Oquendo, *Presentation of the ALF project*, Proc. of the Int. Conf. on System Development Environments and Factories, 1989.

[BENJ99] A. Benjamen, *Une Approche Multi-démarches pour la modélisation des démarches méthodologiques.* Thèse de doctorat en informatique de l'Université Paris 1, Octobre 1999.

[BOEH88] B. Boehm, A Spiral Model of Software Development and Enhancement, IEEE Computer 21(5), 1988.

[BOMM91] P. van Bommel, A.H.M ter Hofstede, Th.P. van der Weide, *Semantics and verification of object-role models.* Information Systems 16 (5), pp. 471-495, 1991.

[BOOC97] G. Booch, I. Jacobson, J. Rumbaugh, *Unified Modeling Language* : version 1.0. Rational Software Coorporation, 1997.

[BRIN00] S. Brinkkemper, *Method engineering with Web-enabled methods.* In Information Systems Engineering : State of the Art and Resaerch Themes, S. Brinkkemper, E. Lindencrona, A. Solvberg (Eds.), pp. 124-133, Springer-Verlag, 2000.

[BRIN90] S. Brinkkemper, *Formalisation of information systems modelling*, Ph. D. Thesis, University of Nijmegen, Thesis Publishers, Amsterdam, 1990.

[BRIN95] S. Brinkkemper, *Method engineering: engineering of information systems developments methods and tools*. Information & Software Technology, 37 (11) pp. 1-6, 1995.

[BRIN96] S. Brinkkemper, *Method engineering: engineering of information systems development methods and tools*, Information and Software Technology, Vol. 38, No.4, pp.275-280, 1996.

[BRIN98] S. Brinkkemper, M. Saeki, F. Harmsen, *Assembly Techniques for Method Engineering*. Proc. of the 10th Conference on Advanced Information Systems Engineering, CAiSE'98. Pisa Italy, 8-12 June, 1998.

[BUSH96] F. Bushmann, R. Meunier, Rohnert et al*., Patron-Oriented Software Architecture – A System of Patrons*. John Wiley, 1996.

[CHEN91] M. Chen, J.F. Nunamaker, G. Mason, The architecture and Design of a Collaborative Environment for System Definition. DATA BASE, pp. 22-28, 1991.

[COAD96] D. Coad, D. North, M. Mayliefd, *Object Models – Strategies, patterns and applications*, Yourdon Press Computing Series, 1996.

[COPL95] J.O Coplien, D.O. Schmidt (Eds.), *Patron Languages of Program Design*. Addison-Wesley, Reading, MA, 1995.

[DENE01] R. Deneckere, *Approche d'extension de méthodes fondée sur l'utilisation de composants génériques*. Thèse de doctorat en informatique de Université Paris 1, Janvier 2001.

[DOWS88] M. Dowson, *Iteration in the Software Process*, Proc. of the 9th Int. Conf. on Software Engineering, 1988.

[EMME91] W. Emmerich, G. Junkermann, B. Peuschel, W. Schäfer, S. Wolf, *MERLIN: Knowledge based process modelling.* 1st European Workshop on Software Process Modeling. A. Fuggetta, R. Conradi, V. Ambriola (Eds), Mila, Italy, May 1991.

[EURO94] *Euromethod Architecture*, Euromethod Project, Deliverable Workpackage 3, 1994.

[FINK90] A. Finkelstein, J. Kramer, M. Goedicke, *ViewPoint Oriented Software Development*, Actes de Conférence "Le Génie Logiciel et ses Applications", Toulouse, p 337-351, 1990.

[FOWL97] M. Fowler, *Analysis Patterns : reusable objects models*, Addison-Wesley, 1997.

[FRAN94] M. Franckson, *The Euromethod deliverable model and its contribution to the objectives of Euromethod*, Proc. IFIP-TC8 Int. Conf. on Methods and Tools for the Information Systems Life Cycle, Verrijn-Stuart and Olle (eds), North-Holland, pp131-149, 1994.

[GAMM94] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns : Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.

[GRUN96] J.C. Grundy, J.R.Venable, *Towards an integrated environment for method engineering*, Proc. IFIP WG 8.1 Conf. on method Engineering, Chapman and Hall, pp 45-62, 1996

[HARM94] A.F. Harmsen, S. Brinkkemper, H. Oei, *Situational Method Engineering for Information System Projects*. In Olle T. W. and A. A. Verrijn Stuart (Eds.), Methods and Associated Tools for the Information Systems Life Cycle, Proc. of the IFIP WG8.1 Working Conference CRIS'94, pp. 169-194, North-Holland, Amsterdam, 1994.

[HARM95a] F. Harmsen, S. Brinkkemper, *Description and Manipulation of Method Fragments for Situational Method Assembly.* Proc. of the Workshop on Management of Software Projects, Pergamon Press, London, 1995.

[HARM95b] F. Harmsen, S. Brinkkemper, *Design and implementation of a method base management system for situational CASE environment*. Proc. 2nd APSEC Conference, IEEE Computer Society Press, pp 430-438, 1995.

[HARM96] F. Harmsen, M. Saeki*, Comparison of four method engineering languages*, IFIP 8.1 Conference on Method Engineering, 1996, Chapman and Hall, pp 209-231, 1996

[HAUM98] P. Haumer, K. Pohl, K. Weidenhaupt, *Requirements Elicitation and Validation with real world scenes.* IEEE Transactions on Software Engineering, 24(12), December. 1998.

[HEND90] B. Henderson-Sellers, J.M. Edwards, *The Object-oriented Systems Life-Cycle*. Communications of the ACM (9), 1990.

[HEY96] D. Hey, *Data Model Patrons : Conventions of Thought*. Dorset House, NY, 1996.

[HEYM92] M. Heym, H. Österle, *A reference model of information systems development*. The Impact of Computer Supported Technologies on Information Systems Development, K.E. Kendall, K. Lyytinen and J.I. DeGross (Eds.), Amsterdam, North-Holland, pp. 215-240, 1992.

[HEYM93] M. Heym, H. Osterle, *Computer-aided methodology engineering*. Information and Software Technology 35 (6/7), pp. 345-354, 1993.

[HEYMA98]P. Heymans, E. Dubois, Scenario-Based Techniques for Supporting the Elaboration and the Validation of Formal Requirements. Requirements Engineering Journal, 3 (3-4), 1998.

[HIDD94] G.J. Hidding, *Methodology information: who uses it and why not?* Proc. WITS-94, Vancouver, Canada, 1994.

[HILL97] J.van Hillegersberg, *Metamodeling-based integration of objet-oriented systems development*. Dissertation, Thesis Publishers, Amsterdam, 1997.

[HOFS92] A.H.M. ter Hofstede, T.F. Verhoef, E.R. Nieuwland and G.M. Wijers, *Specification of Graphic Conventions in Methods*. Proceedings of the 3rd Workshop on Next Generation in Methods, B. Theodoulidis, A. Sutcliffe (Eds.), pp. 185-215, UMIST, Manchester, UK, 1992.

[HOFS93a] A.H.M. ter Hofstede, *Information modelling in data intensive domains.* Dissertation, University of Nijimegen, the Netherlands, 1993.

[HOFS93b] A.H.M. ter Hofstede, H.A. Proper, Th. P. van der Weide, *Formal definition of a conceptual language for the description and manipulation of information models*. Information Systems 18, pp. 489-523, 1993.

[HOFS93c] A.H.M. ter Hofstede, Th. P. van der Weide, Expressiveness in data modelling. Data Knowledge Engineering (10), pp. 65-100, 1993.

[HONG93] S. Hong, G. van der Goor, S. Brinkkemper, *A Formal Approach to the Comparison of Object-Oriented Analysis and Design Methodologies*. Proc. of the 26th Hawaii International Conf. on Systems Science, J. Nunamaker, R. Sprague (Eds.), 4, IEEE Computer Society Press, 1993.

[HULL87] R. Hull, R. King, *Semantic Database Modeling Survey, Applications and Research Issues*. ACM Computing Surveys 19(3), pp.201-206, 1987.

[HUMP89] W.S. Humphrey, *Managing the software process*. The SEI series in Software Engineering. Addison-Wesley, 1989.

[IIVA94] J. IIvari, *Object-Oriented Information System Analysis: Comparative Analysis of six Object-Oriented Analysis Methods*. IFIP Transactions: Methods and Associated Tools for the Information System Life cycle, A. A. Verrijn-Stuart & T. W. Olle (Eds) North-Holland, 1994.

[IIVA90] J. Iivari, *Hierarchical Spiral Model for Information Systems and Software Development*. Information and Software Technoology, 32 (6-7), 1990.

[JARK92] M. Jarke, K. Pohl, *Information systems quality and quality information systems*. Pro. of the IFIP 8.2 Working Conf. on the impact of computer-supported techniques on information systems development, Minneapolis, NM, June 1992.

[JARK99] M. Jarke, C. Rolland, A. Sutcliffe, R. Domges, *The NATURE requirements Engineering*. Shaker Verlag, Aachen 1999.

[KATA89] T. Katayama, *A hierarchical and functional software process description and its enacting*. Proc. of the 11th Int. Conf. on Software Engineering, pp. 343-352, 1989.

[KELL94] S. Kelly, A Matrix for a Meta CASE Environment. Information and Software Technology, 36 (6), pp. 361-371, 1994.

[KELL96] S. Kelly, K. Lyyttinen, M. Rossi, *Meta Edit+: A fully configurable, multi-user and multi-tool CASE and CAME environment*, Proc. of the CAiSE'96 Conf., 20-24 may, Heraklion, Crete, Greece, Springer Verlag, 1996.

[KINN94] K. Kinnuenen, M. Leppänen, *O/A Matrix and a Technique for Methodology Engineering*. Proc. of the 4th Int. Conf. on Information Systems Development, J. Zupansis and S. Wrycza (Eds.), Moderna Organizacija, Kranj, Slovenia, 1994.

[KOTT84] J.E. Kottemann, B.R. Konsynski, *Dynamic Metasystems for Information Systems Development*. Proc. of the 5th Int. Conf. on Information Systems, pp. 187-204, 1984.

[KUMA92] K. Kumar, R.J. Welke, *"Methododlogy Engineering : A Proposal for Situation-Specific Methodologies Construction"*. In Challenges and Strategies for Research in System Development, Cotterman, W.W. and Senn, J.A. eds. , Wiley & Sons Ltd., 1992.

[LIU95] H. Liu, *A visual Interface for Queryig a CASE Repository*. Proc. of the 11th IEEE Symposium on Visual Languages (VL'95), Darmstadt, 1995.

[MAID98] N.A.M. Maiden, CREWS-SAVRE: Scenarios for Acquiring and Validating Requirements. Journal of Automated Software Engineering, 1998.

[MART95] P. Marttiin, K. Lyytinen, M. Rossi, V-P. Tahvanainen, J-P. Tolvanen, *Modeling requirements for future CASE : issues and implementation considerations*. Information Resources Manegement Jpournal 8 (1), pp. 15-25, 1995.

[MYLO90] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis, *Telos : Representing Knowledge About Information Systems*. ACM Transactions on Information Systems, 8(4), pp. 325-362, 1990.

[OEI94] J.L.H. Oei, E.D. Falkenberg, *Harmonisation of information systems modelling and specification techniques.* In Methods and Associated Tools for the Information Systems Life Cycle, pp. 151-168, A.A. Verrijn-Stuart and T.W. Olle (Eds.), No. A-55, Elsevier Science publishers, 1994.

[OEI95] J.L.H. Oei, *A meta model transformation approach towards harmonisation in information system modeling.* In Information Systems Concepts – towards a consolidation of views, pp. 106-127, E.D. Falkenberg, W. Hesse and A. Olivé (Eds.), Chapman & Hall, London, 1995.

[OMG 97] http://www.omg.org

[PLIH96] V. Plihon, *Un environnement pour l'ingénierie des méthodes*, Thèse de doctorat de l'Université Paris 1, janvier 1996.

[POTT89] C. Potts, *A Generic Model for Representig Design Methods*. Proc. of the 11th Int. Conf. on Software Engineering, 1989.

[PRAK97] N. Prakash, *Towards a formal definition of methods*, in Requirements Engineering, 2 (1), 1997.

[PRAK99] N. Prakash, *On Method Statics and Dynamics*. Information Systems 24 (8), pp. 613-637, 1999.

[PREE95] W. Pree, Design Patterns for Object-Oriented Software Development. Addison Wesley, 1995.

[PROT89] L.B. Protsko, P.G. Sorenson, J.P. Tremblay, *Mondrian/ system for automatic generation of dataflow diagrams.* Information and Software technology, 31 (9), pp/ 456-471, 1989.

[PUNT96] H.T. Punter, K. Lemmen, *The MEMA model: Towards a new approach for Method Engineering*. Information and Software Technology, 38 (4), pp.295-305, 1996.

[RALY99a] J. Ralyté, C. Rolland, V. Plihon, *Method Enhancement by Scenario Based Techniques*. Proc. of the 11th Conf. on Advanced Information Systems Engineering, Heidelberg, Germany, June 14-18, 1999.

[RALY99b] J. Ralyté, *Reusing Scenario Based Approaches in Requirement Engineering Methods: CREWS Method Base*. Proc. of the First International Workshop on the Requirements Engineering Process - Innovative Techniques, Models, Tools to support the RE Process, Florence, Italy, September 1999.

[RALY01] J. Ralyté, *Ingénierie des méthodes à base de composants*. Thèse de doctorat en informatique de l'université Paris 1, Janvier 2001.

[RIEU99] D. Rieu, J-P. Giraudin, *Patrons Orintés-objet*. Edition Hermes, 1999.

[ROLL00a] C. Rolland, S. Nurcan, G. Grosz, *A Decision Making Pattern for Guiding the Enterprise Knowledge Development Process*. Journal of Information and Software Technology, 42, pp. 313-331, 2000.

[ROLL00b] C. Rolland, J. Stirna, N. Prekas, P. Loucopoulos, G. Grosz, *Evaluating a Pattern Approach as an Aid for the Development of Organisational Knowledge : An Empirical Study*. Proc. of

the 12th Conf. on Advanced Information Systems Engineering (CAISE 2000), Stockholm, Sweden, May 2000.

[ROLL94a] C. Rolland, *Modelling the evolution of artefacts*, 1st IEEE Int. Conf. on Requirements Engineering, Colorado Springs, Colorado, 1994.

[ROLL94b] C. Rolland, G. Grosz, *A General Framework for Describing the Requirements Engineering Process*, C. IEEE Conf. on Systems Man and Cybernetics, CSMC94, San Antonio, Texas, 1994.

[ROLL95] C. Rolland, C. Souveyet, M. Moreno. *An Approach for Defining Ways-Of-Working*, in the Information Systems Journal, 1995.

[ROLL96a] C. Rolland, V. Plihon, *Using generic chunks to generate process models fragments,* Proc.of 2nd IEEE Int. Conf. on Requirements Engineering", ICRE'96, Colorado Spring, 1996.

[ROLL96b] C. Rolland, N. Prakash, *A proposal for context-specific method engineering*, IFIP WG 8.1 Conf. on Method Engineering, Chapman and Hall, pp 191-208, Atlanta, Gerorgie, USA, 1996.

[ROLL98a] C. Rolland, C. Ben Achour, C. Cauvet, J. Ralyté, A. Sutcliffe, N.A.M. Maiden, M. Jarke, P. Haumer, K. Pohl, Dubois, P. Heymans, *A Proposal for a Scenario Classification Framework*. Requirements Engineering Journal 3 (1), 1998.

[ROLL98b] C. Rolland, C. Souveyet, C. Ben Achour, *Guiding Goal Modelling Using Scenarios*. IEEE Transactions on Software Engineering, special issue on Scenario Management, 24 (12), 1055-1071, Dec. 1998.

[ROLL98c] C. Rolland, V. Plihon, J. Ralyté, *Specifying the reuse context of scenario method chunks*. Proc. of the 10th Conf. on Advanced Information Systems Engineering, CAiSE'98. Pisa Italy, 8-12 June, 1998.

[ROLL99] C. Rolland, N. Prakash, A. Benjamen, *A multi-model view of process modelling*. Requirements Engineering Journal, p. 169-187,1999.

[ROSS95] M. Rossi, S. Brinkkemper, *Metrics in Method Engineering.* Proc. of the 7th Int. Conf. on Advanced Information Systems Engineering CAISE'95, Jyvaskyla, Finland, June 1995.

[ROSS96] M. Rossi, S. Brinkkemper, *Complexity Metrix for Systems Development Methods and Techniques*. Information Systems, 21 (2), pp. 209-227, 1996.

[ROYC70] W. W. Royce, *Managing the development of large software systems*; Proc. of the IEEE WESCON, August, 1970.

[SAEK91] M. Saeki, T. Kaneko, M. Sakamoto, *A method for software process modelling and description using LOTOS*, Proc. 1st Intl. Conf. on the Software Process, IEEE Computer Society Press, Los Alamitos, CA, USA, pp 90-104, 1991.

[SAEK93] M. Saeki, K. Iguchi, K Wen-yin, M Shinohara, *A meta-model for representing software specification & design methods*. Proc. of the IFIP¨WG8.1 Conf. on InformationSystems Development Process, Come, pp 149-166, 1993.

[SAEK94] M. Saeki, K. Wen-yin, *Specifying Software Specification and Design Methods*. Proc. of Conf. on Advanced Information Systems Engineering, CAISE'94, Lecture Notes in Computer Science 811, Springer Verlag, pp. 353-366, Berlin, 1994.

[SISA96] S. Si-said, G. Grosz, C. Rolland, *Mentor, A computer aided Requirements Engineering Environment*, Proc. of the 8th CAISE Conf. Chalenges In Modern Information Systems, Heraklion, Crete, Greece, May 1996.

[SMOL91] K. Smolander, K. Lyytinen, V.-P. Tahvanainen, P. Marttiin, *Meta-Edit - A flexible graphical environment for methodology modelling*, Advanced Information Systems Engineering, (Eds. Andersen R., Bubenko J. and Solvberg A.), LNCS#498, Springer-Verlag, 1991.

[SMOL92] K. Smolander, *OPRR – A Model for Methodology Modeling*. Next Generation of Communication Systems, IOS press, pp. 224-239, 1992.

[SOMM87] I. Sommerville, R. Welland, S. Beer, *Describing software design methodologies*. The Computer Journal. 30 (2), pp. 128-133, 1987.

[SONG92] X. Song, L.J. Osterweil, *Towards objective, systematic, design-method comparison*. IEEE Software, 34 (5), May, pp. 43-53, 1992.

[SONG97] X. Song, Systematic Integration of Design Methods. IEEE Software, 1997.

[SORE88] P.G. Sorenson, J.P. Tremblay, A.J. McAllister, *The Metaview System for Many Specificatio Environments*. IEEE Software, pp. 30-38, 1988.

[SOUV97] C. Souveyet, R. Deneckere, C. Rolland, *State of art : a comparaison of six oriented analysis methods*. TOOBIS project, deliverable T23D1.2, part 1, 1997.

[SOWA92] J.F. Sowa, J.A. Zachmen, *Extending and formalising the framework for information systems architecture*. IBM Systems Journal, 31 (3), pp. 590-616, 1992.

[TOLV93] J.P. Tolvanen, K. Lyytinen, *Flexible method adaptation in CASE environements – The metamodeling approach.* Skandinavian Jurnal of Information Systems, 5 (1), pp. 51-77, 1993.

[TOMI89] T. Tomiyma, T. Kiriyama, H. Takeda, D. Xue, H. Yoshikaya, *Metamodel : A Kaey to Intelligent CAD Systems*. Research in Engineering Design, 1, pp. 19-34. 1989.

[SLOO93] K. van Slooten, S. Brinkkemper, *A Method Engineering Approach to Information Systems Development.* In Information Systems Development process, N. Prakash, C. Rolland, B. Pernici (Eds.), Elsevier Science Publishers B.V. (North-Holand), Amsterdam, 1993.

[SLOO96] K. van Slooten, B. Hodes, *Characterising IS development project*, IFIP WG 8.1 Conf. on Method Engineering, Chapman and Hall, pp 29-44, 1996

[VENA93] G.R. Venable, CoCoA: a conceptual data modelling approach for complex problem domains, Ph.D. dissertation, SUNY, Binghampton, 1993

[VERH95] T.F. Verhoef, A.H.M. ter Hofstede*, Feasibility of Flexible Information Modelling Support*. Advanced Informtion Systems Engineering, J. Iivari, K. Lyytinen and M. Rossi (Eds.), Springer-Verlag, pp. 168-185, 1995.

[VLIS96] J.M. Vlissides, J.O. Coplien, N.L. Kerth (Eds.), *Patron Languages of program Design 2*. Addison-Wesley, 1996.

[WELK92a] R.J. Welke, K. Kumar*, Method Engineering: A Proposal for Situation-specific Methodology Construction*, in Systems Analysis and Design : A Research Agenda, Cotterman and Senn(eds), Wiley, pp257-268, 1992.

[WELK92b] R.J. Welke, *The CASE Repository : More than another database application*. In Challenges and Strategies for Research in Systems Development. W.W.Ctterman and J.A. Senn (Eds.), Wiley, Chichester UK, 1992.

[WIJE91] G. M. Wijers, *Modeling Support in Information Systems Development*, PhD Thesis, Thesis Publishers Amsterdam, 1991.

[WIRF90] J. Wirfs-Brock, H.E. van Dort, *Experiences with the use of CASE tools in the Netherlands*. Advanced Information Systems Engineering, pp. 5-20, 1990.