# Collaborative Construction of Telecommunications Services. An Enterprise Architecture and Model Driven Engineering Method

Vanea Chiprianov

**N° d'ordre** : 2012telb0192

**Sous le sceau de l'Université européenne de Bretagne**

# Télécom Bretagne

**En habilitation conjointe avec l'Université de Bretagne Sud**

École Doctorale – SICMA

---

# Collaborative Construction of Telecommunications Services. An Enterprise Architecture and Model Driven Engineering Method

---

## Thèse de Doctorat

Mention : « Sciences et Technologies de l'Information et de la Communication »


Présentée par **Vanea Chiprianov**

Département : LUSSI

Laboratoire : UMR CNRS 3192 Lab-STICC Pôle : CID

Directeur de thèse : Prof. Yvon Kermarrec Co-directeur : MdC Siegfried Rouvrais


Soutenue le 16 Janvier 2012

**Jury :**

| | |
|---|---|
| Rapporteurs : | Prof. Rolv Bræk, NTNU-Trondheim, Norway |
| | Prof. Noël Crespi, Télécom SudParis, France |
| Examinateurs : | MdC Emmanuel Bertin, associé Télécom SudParis, RD Orange Labs Caen, France |
| | Prof. Isabelle Borne, IUT Vannes, France |
| | Prof. Dorina C. Petriu, Carleton University, Canada |
| Directeurs : | Prof. Yvon Kermarrec, Télécom Bretagne, France |
| | MdC Siegfried Rouvrais, Télécom Bretagne, France |
| Invités : | Patrick Alff, AVP, Luxair, Luxembourg |
| | Dr. Jacques Simonin, Télécom Bretagne, France |

# Contents

# Abstract

In the context of world economies transitioning to services, telecommunications services are the primary means of communication between different economic entities and are therefore essential. The focus on the end consumer, the convergence with the Internet, the separation between the software and the hardware implementing a service, and the telecommunications market deregulation have led to a revolution and a new era in the telecommunications industry. To meet these challenges, former national telecommunications providers have to reduce the construction time, from months to days, while affecting non-negatively other parameters (e.g., cost, quality of service, quality of experience) of new telecommunications services. To tackle this broad theme, we propose a telecommunications service construction process, the software tools that are to be used in this process and a tool building process to build them. The telecommunications service construction process reflects current practices in the telecommunications industry. As such, it should be (easily) accepted by practitioners. The software tools (i.e., Domain Specific Modeling Languages designed as profiles of an Enterprise Architecture Modeling Language, graphical editors, code generators, Off the Shelf network simulators, a collaboration Design Rationale Domain Specific Modeling Language) help telecommunications providers face the challenges. The tool building process relies on models and provides a high automation degree, hence software tools can be build more rapidly. We illustrate the telecommunications service construction process and the tools using a multimedia conferencing service. Our proposals contribute to reducing the construction time of new telecommunications services, while providing the possibility of improved quality of service and increased involvement of the consumer. Faster provisioning of new telecommunications services, that better answer the consumers' needs, will increase the rate of development of new economic services in general, and will ultimately have a positive impact on world economic development.

# Acknowledgements

I would like to acknowledge and thank all those who helped me complete this thesis.

Yvon Kermarrec, my principal research advisor, for trusting me from the beginning of this thesis; the research vision he shared with me and on which this thesis heavily relies; the research, the professional and personal improvement, and the teaching opportunities he offered to me; the continuous research, teaching and general life advice and support, especially in difficult moments.

Siegfried Rouvrais, my research advisor, for the analytic and synthetic spirit, and the scientific rigor he taught me; the opening to other research subjects; the introduction to research in education; the practical and moral guidance in teaching, research and life in general.

Rolv Bræk and Noël Crespi, experts on the research subjects tackled in this thesis, who have honored me by accepting to examine it. Emmanuel Bertin, Isabelle Borne, Dorina C. Petriu, members of the jury, for their interest in my work. All the jury, for the scientific rigor with which they examined my work.

Jacques Simonin, department colleague, for the helpful clarifications, discussions and reviews. Patrick Alff and Martin Woods, BT project leaders, for the guidance and the initial research vision. Yannick LeBras, masters, Ph.D. and office colleague, for initiating me to Latex; the thesis template he generously shared; the continuous advice about life in France - 'j'espère que je ne t'ai pas trop dérangé'; the moral guidance regarding teaching, research and life in general.

Iyas Alloush, Télécom Bretagne masters and then fellow Ph.D. student, for contributing to the definition of the Technology layer Telecom ArchiMate Meta-Model, the Xpand templates for Java generation, the Xpand templates for OPNET configuration file generation, the simulation of the multimedia conferencing joining example with OPNET. Adil Meribaa, Mosbah Lassoued, Télécom Bretagne masters students, for the implementation of the Archi editor extension for the Design Rationale Domain Specific Modeling Language.

Philippe Lenca, department colleague, for the publication strategy tips and the teaching advice. Sébastien Bigaret, department colleague, for the helpful reviews and the support with technical aspects. Sorin Moga, department colleague, for recruiting me and offering me the chance to come to a 'grande école'. Ghislaine LeGall, department colleague, for the continuous support in administrative tasks.

Monica Szemethy, my high school English teacher, not only for my knowledge of English, but mainly for the reading, analysis and critical thinking skills which proved to be such an important part of the research activity. Iasmina Iordache, ESIT Paris 3 masters student, for her careful proof-reading and proficient English corrections.

Ivan and Iuleana Chiprianov, my parents, for everything they have done and still do for me.

Friends and colleagues from the department, the school, for the discussions and fun we had together. All my teachers - each contributed to my development. Finally, all those who helped me complete this thesis and whom I have forgotten to mention here.

# Résumé

**Note**: *Theses completed in a French institution but written in another language are required to provide a French abstract.*

**Note**: *Les thèses accomplies dans un établissement français mais écrits dans une autre langue doivent fournir un résumé français.*

La compétitivité dans le secteur des télécommunications s'est trouvée considérablement intensifiée du fait de la déréglementation du marché associé, de la convergence accentuée avec l'Internet et le monde du développement du logiciel et de l'implication accrue des utilisateurs finaux dans les processus de construction des services. Il s'en suit que la productivité des différents acteurs impliqués dans la création de services de télécommunications doit être plus encore améliorée, tandis que les différentes qualités des services doivent être maintenues au moins aux même niveaux.

## Questions de recherche

Dans un tel contexte, le problème de recherche qui motive cette thèse aborde en tout premier lieu la réduction du temps de construction de nouveaux services télécoms (p.ex. de quelques mois ou quelques semaines à quelques jours, voire quelques heures [Pan 08, Bo 10b]), tout en affectant non-négativement d'autres paramètres (p.ex. le coût, la qualité de service - QoS, la qualité de l'expérience - QoE, la créativité des designers). Pour aborder cette problématique, trois questions de recherche ont été levées pour étayer cette thèse. Elles structurent ses différentes approches et solutions :

1. *QR 1 Processus de construction* : Quel processus de construction devraient suivre les fournisseurs de services télécoms pour relever les défis induis par la convergence avec l'Internet, mieux supporter le changement des architectures techniques (des architectures télécoms classiques vers des architectures multi-couches basées sur le protocole IP (Internet Protocol)), et prendre en compte la participation accrue des utilisateurs finaux ?
2. *QR 2 Outils logiciels* : De nombreuses parties prenantes/acteurs jouent un rôle dans le cycle de vie des services télécoms. Après avoir proposé un processus de construction pour répondre à la *QR 1 Processus de construction*, un degré élevé d'assistance incluant du support pour le travail en équipe et pour le test/vérification, permettra de réduire le temps d'intégration, d'augmenter la qualité du logiciel et ainsi de réduire le temps de construction et l'augmentation de la qualité globale. La question qui se pose alors est : quels outils logiciels devraient être utilisés pour mettre en œuvre et suivre un tel processus ? Parmi les parties prenantes à même d'utiliser de tels outils, les résultats présentés se concentrent sur les fournisseurs de services et les développeurs.
3. *QR 3 Processus de construction d'outils* : Les fournisseurs d'outils devraient pouvoir fournir des environnements logiciels directement dédiés aux fournisseurs et

développeurs de services télécoms pour faciliter leurs travaux. Mais le plus souvent dans des temps trop contraints, les outils logiciels peuvent être coûteux à mettre en place et surtout occuper un certain de temps de conception et développement impactant les temps de mise à disposition des services dans un cycle de vie. En outre, de tels outils se doivent d'être adaptés aux besoins spécifiques des parties prenantes, tout autant que d'utiliser au maximum leurs capacités et habiletés spécifiques. C'est pourquoi il convient de déterminer également : quel processus de construction d'outils est le plus adapté et automatisable ?

# Etat de l'Art

Avant de répondre aux trois questions de recherche précédentes, nous examinons tout d'abord dans ce mémoire les approches existantes pour la création de services télécoms. Nous en identifions les limites et nous retenons deux domaines phares comme piliers qui nous permettront de fournir des solutions pour les surmonter : le domaine de l'architecture d'entreprise et celui de l'ingénierie des modèles.

## Création de services de télécommunications

Nous avons choisi de nous concentrer spécifiquement sur les rôles du fournisseur (c.-à-d. 'Service Provider') et du développeur de services (c.-à-d. 'Service Developer'), les travaux de recherche associés ayant été initiés avec un opérateur international majeur en télécommunications (British Telecom), plus intéressé par ces deux rôles car plus sensible aux besoins de réduction des temps de construction de nouveaux services face à la concurrence. De plus, la phase de construction dans le cycle de vie d'un service télécom a été plus particulièrement abordée car elle est de plus en plus centrale pour les opérateurs, afin de maintenir, notamment, les exigences de qualité. Au cours de cette phase, le fournisseur de services est actif dans les sous-phases de définition, spécification et vérification, tandis que le développeur de services s'attache bien sûr aux activités liées à la sous-phase de développement, conformément aux spécifications. Dans notre proposition, le rôle du fournisseur d'outils est également pris en compte pour mieux répondre à la problématique initiale. Toutefois, même si les principaux résultats de ce travail se concentrent sur ces trois rôles et des phases particulières, les approches et solutions proposées se généralisent ou se transposent parfois à l'ensemble d'un cycle de vie et à d'autres rôles.

Lors de la phase de construction d'un service télécom, plusieurs exigences des fournisseurs et développeurs de services sont à prendre en compte. Dans le cadre du processus de construction de services proposés, nous avons choisis de regrouper ces exigences dans la perspective d'offrir un ensemble d'outils logiciels, génériquement appelé environnement de création de services (c.-à-d. 'Service Creation Environment'). Nous avons choisi d'articuler cet environnement de création de services autour de neuf exigences associées à des activités types par rôles:

1. *Exig 1 Un modèle global* : Les fournisseurs de services s'attachent à commercialiser et vendre leurs services dans des environnements très concurrentiels. Les réseaux supports des services sont toujours au cœur de leurs préoccupations. En effet, les parties techniques sont toujours sensibles et impactent fortement la qualité. Notamment, de par sa forte sollicitation, le réseau et sa planification doivent être décrits

dans un modèle précis. Mais de plus en plus, un modèle décrivant le processus de création de services d'un point de vue marketing est aujourd'hui tout aussi crucial [Kosmas 97]. Le plus souvent séparés, ces différents modèles, situés plus en amont dans un cycle, devraient pouvoir s'intégrer aux modèles des activités techniques liées aux phases située plus en aval, de telle sorte qu'un modèle global de création de services prenant en compte les activités métiers, organisationnelles et techniques puisse être obtenu [Hållstrand 94]. La construction et la gestion des nouveaux services télécoms nécessite ainsi un changement de mentalité, en migrant vers des systèmes de gestion d'entreprise unifiés [Blum 09b] et en incluant des processus de création de services, des processus métiers et des processus de gestion.

2. *Exig 2 Spécificité du domaine* : Les fournisseurs de services veulent réduire leurs temps de conception, les opportunités du marché nécessitant une plus forte réactivité. Une façon de réduire les temps de conception consiste à assister les membres participants aux différentes activités, notamment en utilisant des outils spécialisés pour la tâche et le domaine, p.ex. pour spécifier plus aisément les fonctionnalités d'un service [Hållstrand 94].
3. *Exig 3 Prototypage rapide* : Les fournisseurs de services et les développeurs ont besoin d'une fonction de prototypage rapide pour leurs nouveaux services [Pavan 07], afin d'évaluer leurs succès potentiels sur le marché [Hållstrand 94, Khlifi 08], tout en intégrant les commentaires des utilisateurs. Ceci peut être réalisé en utilisant un haut degré d'automatisation [Kosmas 97].
4. *Exig 4 Support collaboratif* : Un service complexe ne se construit pas tout seul. Les fournisseurs et les développeurs de services ont besoin de soutien, de sorte que leurs équipes puissent partager, collaborer et mieux communiquer sur leurs choix de conception [Kosmas 97].
5. *Exig 5 Vérification/simulation précoce* : Les développeurs de services ont besoin de réaliser très tôt des vérifications sur leurs services, de détecter des incompatibilités ou erreurs, tout autant que de simuler leur future offre fonctionnelle et les qualités associées [Pavan 07]. Notamment, dans un contexte de services télécoms largement distribués, il est primordial de garantir que le fonctionnement d'un nouveau service ne produira pas d'effets secondaires indésirables [Kosmas 97]. Ainsi, des outils sont à intégrer pour les phases de vérification et tests.
6. *Exig 6 Intégration* : Tous les rôles dans un cycle de vie ont besoin de la capacité à s'intégrer et à interopérer avec des standards et outils spécifiques à leurs domaines. Ils ne parlent pas tous le même langage et n'utilisent pas les mêmes modèles. L'environnement de création de services doit être une plate-forme qui utilise une bibliothèque de composants et des méthodes "plug-in" bien établies [Kosmas 97].
7. *Exig 7 Réutilisation* : Tout nouveau service, même s'il est innovant, est de moins en moins indépendant des autres services. Il peut être conçu et développé dans le cadre de classes de services existants (et donc réutiliser des parties d'autres services) [Kosmas 97]. Une bibliothèque de composants logiciels peut être établie [Pavan 07] pour cela.
8. *Exig 8 Large gamme de services* : Les fournisseurs doivent prendre en charge des services pour différents types de réseaux [Pavan 07]. Ils préfèrent pouvoir s'abstraire des spécificités de certaines plateformes [Khlifi 08].
9. *Exig 9 Evolution facile des services* : Les fournisseurs de services doivent être capa-

bles de changer la logique de leurs services de manière rapide et efficace, pour mieux répondre aux attentes de leurs usagers [Yelmo 08]. En cela, il convient de définir les services indépendamment d'une technologie réseau spécifique. Les réseaux actuels, verticalement intégrés, ont besoin de migration vers des structures en couches horizontales offrant des interfaces plus ouvertes [Falcarin 08] (c.-à-d. une plateforme de services, basée sur des services partagés et des activateurs de réseau, qui peut être facilement composée [Pollet 06]). Cette vue en couches permet de supporter des environnements (ouverts) multi-joueurs [Khlifi 08].

L'état de l'art nous a permis d'identifier et de définir six catégories d'approches existantes pour la création de services télécoms :

- trois pour Next Generation Network (NGN):
  - *Cat 1 SIP - dépendant*;
  - *Cat 2 SIP - indépendante*;
  - *Cat 3 NGN composition*;
- deux pour Web:
  - *Cat 4 Parlay X*;
  - *Cat 5 Web mash-up*;
- *Cat 6 Hybride.*

Il est important de préciser que l'inclusion d'un environnement dans une catégorie ou une autre ne permet pas toujours de caractériser pleinement sa portée. Beaucoup de solutions individuelles combinent plusieurs caractéristiques et peuvent ainsi être incluses dans plusieurs catégories ; le choix de les inclure dans une catégorie ou une autre vise à refléter leurs caractéristiques principales (telles que perçues dans leurs présentations dans la littérature).

Un certain nombre de conclusions peuvent être tirées de l'analyse de cet état de l'art :

- *Exig 1 Un modèle global* n'est traitée par aucune des catégories. Pour y répondre, nous étudions et nous nous appuyons sur les architectures d'entreprises dans la section suivante.
- *Exig 4 Support collaboratif*, *Exig 5 Vérification/simulation précoce* et *Exig 6 Intégration* sont liées à l'architecture des environnements de création de services. Pour assurer *Exig 6 Intégration*, une architecture intégrant plusieurs composants (p.ex. autour d'un bus ou basé sur des plug-ins) serait fortement recommandée. En outre, elle permettrait l'intégration de composants, d'outils (existants - composants sur étagère, ou nouveaux) pour *Exig 4 Support collaboratif* et *Exig 5 Vérification/simulation précoce.*
- Les approches pour le Web, contrairement à celles des Next Generation Network, ont un haut degré de *Exig 7 Réutilisation*, un possible *Exig 8 Large gamme de services*, *Exig 9 Evolution facile des services* et *Exig 3 Prototypage rapide.* Ces qualités sont fortement souhaitables.
- Les approches pour le Web, contrairement à celles des Next Generation Network, ont un faible degré de *Exig 2 Spécificité du domaine.* Ceci représente un inconvénient majeur. Au vu des qualités souhaitables identifiées dans le point précédent, les approches Web (en général les TIC), seraient très bénéfiques, sous réserve qu'un moyen d'inclure la spécificité de domaine soit trouvée dans ces environnements. La piste retenue dans nos travaux pour cela consiste à introduire des Domain Specific Modeling Languages.

– Les approches Web se concentrent sur la composition de services primitifs. Ces solutions devraient également être étendues à la création de ces mêmes services.

## Architecture d'entreprise pour les services de télécommunications

Comme base des exigences des fournisseurs et des développeurs de services, notamment *Exig 1 Un modèle global*, la construction et la gestion des nouveaux services télécoms amènent à proposer une approche plus systémique prenant notamment en compte les activités métiers, fonctionnelles et techniques pour les rôles retenus. En ce sens, les architecture d'entreprise visent à obtenir une représentation globale. Comme premier pilier de nos propositions, ce domaine est détaillé ci-après.

Une initiative importante dans le domaine de l'architecture d'entreprise est la définition de cadres (c.-à-d. 'frameworks'), qui visent à structurer les concepts et les activités nécessaires pour concevoir et construire un système complexe. Un cadre spécifique des télécommunications (qui n'est pas à proprement parler un cadre d'architecture d'entreprise) pour l'organisation, la spécification et le développement de nouveaux systèmes de gestion de génération est Frameworx. Mais Frameworx souffre de quelques limitations. Des travaux récents ont établi des synergies riches entre Frameworx et un cadre d'architecture d'entreprise, TOGAF, quant à lui largement reconnu dans le monde des systèmes d'information d'entreprise. Il existe de nombreux exemples d'architecture d'entreprise appliqués au développement de services de télécommunications. Les principales raisons de son utilisation sont liées à la gestion de la complexité et à la croissance de la ré-utilisabilité des composants. Plus précisément, il y a un effort en cours pour transposer le cadre des télécommunications, Frameworx, dans un cadre d'architecture d'entreprise à la TOGAF. Cette thèse défend l'application de TOGAF comme base en tant que cadre d'architecture adapté pour les télécommunications ; il est envisagé comme un cadre général qui peut être étendu et complété au moyen de concepts clés du monde des télécommunications et d'activités spécifiques issues de Frameworx.

Toutefois, un cadre est juste théorique et ne propose pas d'outils. Pour être appliqué à la modélisation effective des systèmes, un cadre requiert des langages de modélisation. Un langage de modélisation d'architecture d'entreprise très proche de TOGAF est ArchiMate. Ainsi, notre proposition s'est concentrée sur ArchiMate en y intégrant des concepts spécifiques au domaine des télécommunications. Le langage de modélisation de services et systèmes télécoms en résultant, défini comme une extension d'un langage de modélisation d'architecture d'entreprise (dans notre cas ArchiMate), bénéficie ainsi de la plupart des avantages de l'architecture d'entreprise.

Prenant en considération les exigences des fournisseurs et des développeurs de services pour l'environnement de création de services, l'usage des architectures d'entreprise nous amène potentiellement à contribuer à:

– *Exig 1 Un modèle global* : Obtenir un modèle intégré global de la création de services télécoms prenant en compte des activités métiers, de gestion et techniques. En effet, les architecture des entreprises présentent une vue unifiée de l'entreprise, à partir de différents points de vue, à différents niveaux ;
– *Exig 6 Intégration* : Contribuer à l'intégration/l'interopérabilité des modèles produits à différents niveaux. En effet, le langage spécifique de modélisation d'architecture d'entreprise ArchiMate permet de définir des relations possibles entre ses couches.

Toutefois, l'interopérabilité entre les points de vue reste encore un problème scientifique ;
- *Exig 7 Réutilisation* : Réutiliser des composants pour offrir des composants à un plus haut niveau d'abstraction. En effet, en introduisant une architecture de l'entreprise en couches, des composants des couches basses (p.ex. réseaux) peuvent être capitalisés ;
- *Exig 8 Large gamme de services* : Permettre la séparation des activités liées aux descriptions fonctionnelles et métiers des services de télécommunication, de celles de la plateforme/réseau, grâce à l'architecture en couches ;
- *Exig 9 Evolution facile des services* : Mieux assurer l'évolution des architectures.

## L'ingénierie dirigée par les modèles pour les services de télécommunications

Les architectures d'entreprise fournissent une représentation unifiée de l'entreprise. Nous avons choisi un cadre d'architecture d'entreprise, TOGAF, et un langage de modélisation pour l'appliquer, ArchiMate. Toutefois, ArchiMate est un langage pour la modélisation de n'importe quelle architecture d'entreprise, alors que, pour augmenter les performances, les fournisseurs de services doivent bénéficier de l'intégration des concepts de télécommunication directement dans le langage.

Une approche pour définir des langages de modélisation est supportée par le domaine de l'ingénierie dirigée par les modèles, notre second pilier. Cette approche permet notamment de générer des langages graphiques dédiés, qui offrent une expressivité suffisante pour les futurs concepteurs. En plaçant les modèles au centre de la démarche, l'ingénierie des modèles offre de sérieux atouts pour la génération d'outils associés à ces mêmes langages (p.ex. des éditeurs graphiques). De plus, les modèles permettent d'introduire un haut degré de formalité, permettant un large éventail d'opérations, comme des transformations de modèles automatisables.

Il y a de nombreux exemples de l'utilisation de l'ingénierie dirigée par les modèles, dans toutes les phases des cycles de vie en ingénierie logicielle. Par conséquent, l'ingénierie des modèles est tout autant à même d'aider à la création de services de télécommunication. Parmi les exigences pour l'environnement de création de services que nous avons retenues, l'ingénierie dirigée par les modèles nous amène potentiellement à contribuer à :
- *Exig 2 Spécificité du domaine* : Définir des langages de modélisation spécifiques aux télécommunications, grâce à l'approche de méta-modélisation pour la définition des langages ;
- *Exig 3 Prototypage rapide*: Assurer un haut degré d'automatisation, en s'appuyant sur l'utilisation de transformations de modèles à différents niveaux d'abstraction ;
- *Exig 5 Vérification/simulation précoce* : Intégrer des outils de test et vérification, soit en utilisant des outils de test dirigés par les modèles, soit en intégrant des outils de test spécifiques au domaine des télécoms, en s'appuyant auparavant sur des transformations de modèles ;
- *Exig 6 Intégration*: Utiliser des transformations de modèles pour décrire des traductions (syntaxiques) entre les outils. En effet, il est à prévoir que plusieurs langages de modélisation et outils associés seront utilisés; l'utilisation d'une approche par méta-modélisation pour la définition des langages permet l'introduction de transfor-

mations de modèles génériques ;
- *Exig 7 Réutilisation* : Faciliter la capture, par les designers, des éléments de constructions spécifiques des services télécoms à travers les langages de modélisation offerts. En effet, en raison des nombreux concepts, des mécanismes sont nécessaires pour décrire les points communs et différences entre les différents services télécoms (p.ex. lignes de produits logiciels) et les partager entre concepteurs ;
- *Exig 8 Large gamme de services* : Supporter plusieurs types de réseaux pour un même service. L'indépendance vis-à-vis de la plate-forme est assurée grâce à la séparation des modèles, en partant des modèles indépendants des plateformes et des modèles spécifiques aux plateformes cibles ;
- *Exig 9 Evolution facile des services* : Propager plus facilement, et ainsi mieux supporter, l'évolution des différentes exigences. En effet, en raison des associations entre les modèles et leurs différents niveaux d'abstraction, un degré d'automatisation plus élevé des évolutions peut être assuré dans les différentes phases du cycle de vie.

L'ingénierie dirigée par les modèles nous apporte des bénéfices pour deux types de profils d'acteurs: quand elle est utilisée par les fournisseurs et les développeurs de services (c.-à-d. les concepteurs d'un produit/service), et lorsqu'elle est utilisé par les fournisseurs d'outils (c.-à-d. ceux qui fournissent des outils logiciels pour les concepteurs d'un produit/service). Pour le premier profil, l'ingénierie des modèles amène à plus de formalisation (avec des impacts non négligeables sur la réduction du nombre d'erreurs), une plus grande automatisation (impacts sur la réduction du temps et des coûts, meilleure réutilisation, plus de disponibilité des designers pour des tâches plus créatives), ainsi qu'une meilleure interopérabilité. Tous ces éléments contribuent à réduire considérablement le temps de construction d'un nouveau produit/service. Pour le deuxième profil (c.-à-d. les fournisseurs d'outils), à travers ses transformations de modèles, l'ingénierie des modèles augmente le taux de réutilisation, tout en permettant des extensions et des combinaisons de langages existants. De plus, il devient plus aisé et rapide de produire des outils logiciels spécifiques à un langage, de définir et générer des ponts pour l'interopérabilité des langages et des outils. Cependant, le coût des outils demeure important, même en utilisant une approche dirigée par des modèles. Par conséquent, un processus uniforme qui fournit des outils pour tous les intervenants profiteraient considérablement aux fournisseurs d'outils.

## CONTRIBUTIONS

Cette thèse propose un processus de construction outillé de services télécoms, inspiré du cycle de développement logiciel en cascade. Pour construire les outils dédiés utilisés dans ce processus de construction, nous proposons également un second processus de construction d'outils piloté par les modèles. Nous avons construit les outils à utiliser pour les rôles de fournisseur et développeur de services selon notre second processus. Une application du processus de construction de services télécoms, utilisant les langages proposés et plusieurs outils générés, est proposé pour refléter une étude de cas.

### Un processus de construction de services de télécommunications

Dans cette section, nous présentons notre proposition à *QR 1 Processus de construction*, un processus de création de services de télécommunications, inspiré du cycle de

développement logiciel en cascade. Il se compose de quatre activités principales que chaque partie prenante effectue, après quoi le modèle passe en aval, à l'intervenant suivant, qui va faire une série d'activités similaires. Ces activités principales sont :

1. *AM 1 Modeliser* le service de télécommunications de son point de vue ;
2. *AM 2 Tester* le modèle ;
3. *AM 3 Collaborer* avec d'autres designers qui partagent son point de vue ;
4. *AM 4 Inter-operer* avec les outils logiciels de parties prenantes d'autres points de vue.

Afin de soutenir l'activité de modélisation, nous proposons que les concepteurs utilisent des langages de modélisation spécifiques à leur domaine. Afin de soutenir les tests, nous offrons la possibilité de simuler des modèles. Afin de soutenir la collaboration, nous nous concentrons sur les justifications de conception et proposons que les concepteurs utilisent un langage de modélisation spécifique au domaine pour les capturer. Pour soutenir l'interopérabilité entre les outils, nous proposons une approche automatique.

Bien que nous proposions un processus impliquant quatre activités de modélisation, il y a d'autres activités des concepteurs qui n'ont pas été abordées (p.ex. la planification du projet, la consultation avec d'autres équipes). Ces besoins n'entrent pas dans le cadre de cette thèse.

Notre proposition à *AM 1 Modeliser*, définit un langage de modélisation spécifique au domaine pour chaque rôle, tout en promettant l'augmentation du pouvoir d'expression et donc de meilleures performances lors des phases de modélisation. C'est en effet très souvent une tâche difficile et coûteuse. Il serait avantageux que certains de ces inconvénients soient réduits. Dans ce contexte, proposer de définir des langages de modélisation spécifiques au domaine comme des extensions des langages de modélisation existants constitue un atout.

Nous abordons *AM 2 Tester* en suggérant l'intégration avec des logiciels existants spécifiques au secteur des télécoms et largement utilisés, p.ex. des simulateurs de réseaux. Toutefois, la prise en compte des résultats de tests (ou résultats de simulation), directement dans les modèles, serait très utile. En outre, une intégration à de nombreux autres outils de test, capables d'analyser les différents aspects des modèles de télécommunications, est nécessaire.

Comme les tests, *AM 3 Collaborer* est un sujet vaste. Nous avons abordé l'un des enjeux les plus importants, le manque de justification des décisions lors des étapes de modélisation. En s'appuyant sur les derniers résultats de recherche concernant la justification des décisions en modélisation, nous avons retenu de proposer un langage de modélisation spécifique au domaine basé sur des schémas et finalement intégré aux langages de modélisation spécifiques aux télécoms. De la même manière, d'autres outils logiciels et des dispositifs de collaboration (p.ex. éditeurs collaboratifs multi-participants, temps réel, distribués, supportant des fonctions de partage de tableau et d'écran [Prechelt 11]; table tactile) devraient être intégrés.

Pour aider un designer dans son/sa *AM 4 Inter-operer*, nous proposons d'assurer l'interopérabilité des langages de modélisation spécifiques au domaine automatiquement, à la fois aux niveaux syntaxique et sémantique. Toutefois, cela ne résout pas le flux d'informations en amont, et il appartient au designer d'intégrer les contraintes qui peuvent provenir d'autres vues, rôles.

Notre processus de création de services de télécommunications propose un ensemble d'activités semblables pour les designers de tous les rôles. L'avantage est que les mêmes activités sont proposées pour chaque rôle, ce qui signifie que le vendeur d'outils peut réutiliser la même solution à *QR 3 Processus de construction d'outils*, et ceci pour tous les rôles. Toutefois, étant basé sur un modèle en cascade, notre processus présente l'inconvénient potentiel de manquer de souplesse, propriété que l'on retrouve plutôt dans les cycles itératifs et agiles. Bien qu'il y ait quelques informations qui passent en amont, celles-ci sont le plus souvent - et heureusement - plutôt réduites. Une fois qu'un rôle a terminé ses activités, il n'y a pas lieu de revenir en amont. Le processus proposé illustre l'intégration des langages de modélisation spécifiques au domaine, dans le processus du génie logiciel pour la création de services de télécommunications, et l'interaction de la *QR 1 Processus de construction* avec les activités d'autres designers.

Nous avons présenté dans cette section notre proposition pour un processus de création de services de télécommunications. Il reflète les pratiques actuelles dans l'industrie des télécommunications, selon lequel un service télécom est défini encore bien souvent en utilisant une approche en cascade, sans revenir à la phase précédente, et dans l'isolement des services. En tant que tel, ce processus devrait être (facilement) accepté par les praticiens. Bien sûr, pour l'avenir, il convient d'envisager des processus plus flexibles pour intégrer le retour de l'utilisateur final plus rapidement et supporter plus de flexibilité. Ces processus pourraient être inspirés par des méthodes itératives de développement logiciel. Il faut toutefois noter que dans ce contexte plus large, le processus que nous proposons est une étape transitoire nécessaire.

## Un processus de construction d'outils pour les fournisseurs d'outils

Dans cette section, nous présentons notre proposition à *QR 3 Processus de construction d'outils*. Pour définir nos langages de modélisation spécifiques au domaine (pour la modélisation de services et la collaboration), construire leurs outils logiciels associés, générer systématiquement des fichiers de configuration pour des outils existants de simulation de réseau, ou encore assurer l'interopérabilité de manière automatique, nous proposons un processus de construction d'outils (cf. Figure 1). Ce second processus est directement destiné aux fournisseurs d'outils. Les tâches pour définir des langages de modélisation spécifiques au domaine sont inspirées de l'approche méta-modélisation pour la définition du langage de modélisation. La mise à disposition des tests par simulation est basée sur des approches génératives qui prennent des modèles en entrée. L'intégration du langage de modélisation traitant de la collaboration à nos langages de modélisation de services est basée sur la combinaison de modèles. Quant à elle, l'interopérabilité est assurée en remodelant les méta-modèles sous la forme d'ontologies, afin de les aligner, pour enfin générer des transformations de modèles réutilisables.

## Langages de modélisation et des outils logiciels spécifiques pour la construction de services télécoms

Pour le processus de création de services télécoms proposé, nous avons construit des outils logiciels en suivant le processus proposé précédemment. Ces outils constituent notre réponse à la *QR 1 Processus de construction*. Pour permettre la *AM 1 Modeliser*, nous


Figure 1 : Le processus de construction d'outils pour les fournisseurs d'outils.

avons défini plusieurs langages de modélisation spécifiques pour deux acteurs, le fournisseur et le développeur de services. Le développement des langages de modélisation consiste en la définition de syntaxes abstraites et concrètes, et de leurs sémantiques. Pour réutiliser les définitions et les outils logiciels des langages de modélisation, nous avons conçu ces langages comme des profils/extensions d'un langage de modélisation d'architecture d'entreprise retenu précédemment (c.-à-d. ArchiMate). Pour permettre la *AM 2 Tester*, nous avons généré des modèles d'entrée pour un outil de simulation de réseau existant, OPNET. Pour permettre la *AM 3 Collaborer*, nous avons défini un langage de modélisation spécifique pour la capture des justifications de conception. La *AM 4 Inter-operer* est principalement automatique, basée sur les relations définies entre les couches d'ArchiMate. Nous présentons l'intégration de ces langages de modélisation et des composants sur étagère dans notre environnement de création de services.

Cet environnement de création de services permet de satisfaire toutes les exigences des fournisseurs et développeurs de services identifiés dans l'introduction. Les langages de modélisation spécifiques au domaine, grâce à leur expressivité ciblée, permettent aux concepteurs d'obtenir à la fois de meilleures performances et une précision accrue dans les phases de conception, développement et vérification. Comme nous l'avons implémenté, la transformation vers un simulateur de réseau permet par exemple aux concepteurs de simuler leurs modèles, de découvrir des erreurs ou des problèmes de dimensionnement au plus tôt. La capture des choix de conception au sein des modèles est bénéfique pour la collaboration, car elle favorise la coordination, permet d'exposer différents points de vue, et favorise les consensus au sein des équipes. Assurer l'interopérabilité entre les outils logiciels réduit de manière considérable les problèmes d'échange et le temps de conception avant la mise en production.

## Application du processus et outils logiciels pour la construction des services télécoms à travers une étude de cas complète

Nous avons appliqué notre processus de création de services télécoms pour les rôles du fournisseur et du développeur à un service de conférence, en y incluant les outils proposés au sein de Archimate. Nous avons commencé avec *AM 1 Modeliser* du ce service en utilisant les langages de modélisation spécifiques aux télécommunications et définis comme des extensions du langage d'architecture d'entreprise retenu. Nous avons poursuivi avec la *AM 2 Tester*, en utilisant le simulateur de réseau OPNET. Pour la *AM 3 Collaborer*, les décisions de conception sur ce modèle ont été capturées en utilisant le langage de modélisation d'architectures d'entreprises. La *AM 4 Inter-operer* des deux langages de modélisation de couches adjacentes, en raison du choix de les concevoir comme des extensions de ArchiMate, se trouve grandement simplifiée par les relations inter-couches pre-définies par ArchiMate.

La plupart des modèles de services de téléconférence disponibles dans la littérature manquent de formalisme, s'appuyant très rarement sur un langage de modélisation. Des auteurs présentant de tels services utilisent habituellement des images pour aider à transmettre leur signification, et en tant que tels, ces modèles n'ont qu'une valeur de communication. Ils ne peuvent pas être utilisés pour générer du code exécutable ou effectuer des tests, des simulations. Par ailleurs, en raison du manque de formalité, des ambiguïtés sur le sens de certains concepts ou sur leurs relations peuvent exister. De notre côté, la manière

dont nous modélisons les services télécoms, tel qu'un service de téléconférence, s'appuie sur un formalisme clair et permet de générer du code exécutable pour enfin le simuler.

Pour conclure, notre travail permet de réduire le temps de construction et la fourniture des nouveaux services télécoms et représente la base pour d'autres améliorations qui permettront de le réduire encore plus. Ces gains de temps dans le domaine concurrentiels des télécommunications, tout en tenant compte des exigences de qualité et d'une réponse réactive aux besoins des consommateurs et utilisateurs, vont augmenter considérablement le taux de développement de nouveaux services économiques en général, et pourrait avoir un impact sur le développement économique mondial.

# Personal Publications

The contributions of these thesis have been presented also in:

– Journal articles and book chapters:

1. [Chiprianov 12a] CHIPRIANOV Vanea, KERMARREC Yvon, ROUVRAIS Siegfried. *Integrating DSLs into a Software Engineering Process: Application to Collaborative Construction of Telecom Services*. Ed. M. Mernik, Formal and Practical Aspects of Domain-Specific Languages: Recent Developments, IGI Global, 2012, (submitted).

   Abstract: "The development of large and complex systems involves many people, stakeholders. Engineeringly speaking, one way to control this complexity is by designing and analyzing the system from different perspectives. For each perspective, stakeholders benefit from means, tools, languages, specific to their activity domain. A Domain Specific Language (DSL) per perspective is such a dedicated means. While DSLs are used for modeling, other means, tools, languages, are needed for other connected activities, like testing or collaborating. However, using together such different types of tools, integrating DSLs into stakeholders' software process, is not straightforward. In this chapter we advance an integration process of DSLs with other tools. To each stakeholder, we propose they have their own DSL with associated graphical editor, operational semantics and generation of scripts for off the shelf simulators for e.g. testing. Additionally to the integrated stakeholders' software process, we introduce a model driven process dedicated to the tool vendor which creates the DSLs and its associated tools. Due to the integration of DSLs into this process, we contend that stakeholders will significantly reduce system construction time. We illustrate the two processes on Telecommunications service construction.
2. [Chiprianov 12b] CHIPRIANOV Vanea, KERMARREC Yvon, ROUVRAIS Siegfried, SIMONIN Jacques. *Extending Enterprise Architecture Modeling Languages for Collaboration. Application to Telecommunications Service Design*. Software and Systems Modeling, 2012, (submitted).

   Abstract: "The competitive market forces organizations to be agile and flexible so as to react robustly to complex events. To achieve this, Enterprise Modeling proposes to decompose complex organizations into concerns like geographical position of the enterprise, business processes, applicative functions and technical deployment architecture. As such, it permits to analyze more finely relationships between them. These various concerns need to be aligned with the enterprise strategy to strengthen it.
   To model such a complex environment that is an enterprise, many stakeholders, with different expertise, must work together. However, they all have a common need: taking decisions on different issues, using specific criteria and justifying them with shareable arguments (i.e. decision rationale). These decisions and

their rationale are not always captured explicitly, in a standard, formal manner. Hence, many of them are never communicated to other stakeholders and are forgotten once the ones that took them leave the enterprise or project. This may result in a loss of knowledge, high difficulty in maintaining and evolving existing systems, encumbrance of communication and common understanding, poor collaboration, lack of traceability, low quality.
Decision rationale is as an essential component of collaboration. However, the main problem is enticing stakeholders in capturing this rationale. This article synthesizes an approach for capturing and using the rationale behind enterprise modeling decisions. It promotes coordination, enables exposing different stakeholder points of view, facilities participation and collaboration in modeling activities - activities focused here on Enterprise Architecture viewpoints. It ultimately results in overall enterprise increased reactivity so as to enhance performance and reduce costs for the enterprise.
The approach is implemented through a Domain Specific Modeling Language for capturing decision rationale. This language is independent of the domain in which is applied and can be used for any kind of decision rationale. To entice stakeholders to use it, this Domain Specific Modeling Language is defined as an extension of a standard Enterprise Architecture Modeling Language. As the collaboration artifacts are integrated with the enterprise models, this reduces stakeholders' inhibition to communicate and capture decision rationale. The associated graphical editor for this extension is build on top of Eclipse using a Model Driven Engineering approach, to leverage the automation, easy configuration and evolution of meta-tools.
To present benefits of the approach, such as rapid prototyping, code generation, easy integration with existing tools, the Domain Specific Modeling Language for capturing decision rationale is applied to large organizations in the context of Telecommunications service design. It is exemplified on modeling and capturing decisions on a conference service."

– Communications in international peer-reviewed conferences with ISBN proceedings:
  1. [Chiprianov 11a] CHIPRIANOV Vanea, KERMARREC Yvon, ROUVRAIS Siegfried. *Extending Enterprise Architecture Modeling Languages: Application to Telecommunications Service Creation*. The 9th Enterprise Engineering track at the 27th Symposium on Applied Computing (SAC), 26-30 march, Trento, Italy, 2012, 6pp, (accepted) - rank B [ERA 2010], acceptance rate 26%.

     Abstract: "Enterprise Engineering offers a global view on multiple concerns such as processes, stakeholders, supporting technology. This global view is sustained by Enterprise Architecture frameworks, languages, tools and standards. The current effort has been focused on general purpose Enterprise Architecture frameworks, modeling languages and tools, which allow describing a wide range of domains. While they are expressive enough at the business layer, at the technical layer, where more detail is needed to describe a domain specific system, such general purpose Enterprise Architecture Modeling Languages sometime lack the semantic strength required. The concepts present in the language are too abstract, they need refinement and specification. To provide the necessary

specific semantic strength, this paper proposes an approach to extend Enterprise Architecture Modeling Languages with domain specificity. The proposed approach is a model-driven one, allowing a high degree of automation in the building of tools for the language extension. To better show its benefits, the approach is applied on the domain of Telecommunications, for defining an Enterprise Architecture Modeling Language extension for service creation. The so defined language and its associated tools are illustrated on an IP Multimedia Subsystem conferencing service example."

2. [Chiprianov 11b] CHIPRIANOV Vanea, KERMARREC Yvon, ROUVRAIS Siegfried. *On the Extensibility of Plug-ins*. 6th International Conference on Software Engineering Advances (ICSEA), 23-28 october, Barcelona, Spain, 2011, pp 557-562, ISBN 978-1-61208-165-6 - rank C [ERA 2010], acceptance rate 30%.

   Abstract: "There are software engineering tooling problems for which the solution benefits from being encapsulated as a plug-in. Among these problems, to ensure higher leverage, there are categories for which is important that their solution is extensible. However, extending a plug-in in practice often takes a long time, requires expertise, involves hacks and produces low quality code. In this paper, we advocate that assuring early in the design that a plug-in is extensible, by providing the necessary extension points, increases its re-usability, improves its evolution, and ultimately reduces the development time of the extender plug-in. We identify categories of software engineering problems whose solutions benefit from being extensible plug-ins, and review existing approaches to extending plug-ins. Finally, we report on our experience, with some of these approaches, in extending an Eclipse plug-in for a domain specific modeling language graphical editor."

3. [Chiprianov 11d] CHIPRIANOV Vanea, ALLOUSH Iyas, KERMARREC Yvon, ROUVRAIS Siegfried. *Telecommunications Service Creation: Towards Extensions for Enterprise Architecture Modeling Languages*. Enterprise Software Technology area of the 6th International Conference on Software and Data Technologies (ICSOFT), 18-21 july, Seville, Spain, 2011, vol. 1, pp. 23-29, ISBN 978-989-8425-76-8, DOI 10.5220/0003441100230028 - rank B [ERA 2010], acceptance rate 12%+28%=40%.

   Abstract: "From the 90's, the telecommunications service creation industry has undergone radical change. Services have shifted from being based on a switching environment to being mainly based on software. To remain competitive in these new dynamic conditions of an open market, telecommunications organizations need to produce high quality services at low prices within short periods of time. Concerning Service Providers, they need an overall representation of service creation taking in all business, management, and technical activities. To reduce their concept-to-market time for new services, they also need tools specialized for their tasks and domain. In this position paper, we argue that a telecommunications profile for an Enterprise Architecture modeling language answers these needs. We also design a telecommunications profile for ArchiMate

that offers conformity to standards through the reuse of a recognized Enterprise Architecture modeling language. Moreover, this profile provides easier adoption by Service Providers due to inclusion of domain specific concepts. The profiling mechanism we propose may be used for defining language extensions specific to other industries as well."

4. [Chiprianov 11e] CHIPRIANOV Vanea, KERMARREC Yvon, ROUVRAIS Siegfried. *Towards semantic interoperability of graphical domain specific modeling languages for telecommunications service design.* 2nd International Conference on Models and Ontology-based Design of Protocols, Architectures and Services (MOPAS), 17-22 april, Budapest, Hungary, 2011, pp. 21-24, ISBN 978-1-61208-005-5 - best paper.

   Abstract: "High competition pressures Telecommunications service providers to reduce their concept-to-market time for new services. Decomposing the service into dedicated views permits to easily manage complexity among several actors. However, there is still a lack of tools to fully support and implement this method. Consequently, in this paper we propose to define a Domain Specific Modeling Language for each viewpoint, regrouped into a family of modeling languages, relying on a meta-modeling approach. To ensure better interaction and coherency between the various modeling viewpoints, we focus on interoperability issues at design time. We defend that in order to adequately manage interoperability between distinct models, interoperability between their meta-models should be established as well. For this we rely on model transformations between meta-models. However, most often model transformations address only the syntactic level of interoperability. To increase the formality of languages and of their interoperability, semantics must be taken into consideration as well. Thus, to address the semantic level of interoperability, we propose lifting the meta-models into ontologies, enriching and matching them into shared ontologies. This allows for semi-automatic generation of model transformations between meta-models from the shared ontologies."

5. [Chiprianov 09c] CHIPRIANOV Vanea, Kermarrec Yvon, ALFF Patrick. *A model-driven approach for telecommunication network services definition.* The Internet of the Future, 15th Open European Summer School and IFIP TC6.6 Workshop (EUNICE), 7-9 september, Barcelona, Spain. Proceedings: Lecture notes in computer science (LNCS), 2009, vol. 5733, pp. 199-207, ISBN 978-3-642-03699-6, DOI http://dx.doi.org/10.1007/978-3-642-03700-9_21 - acceptance rate 23/60=38%.

   Abstract: "Present day Telecommunications market imposes a short concept-to-market time for service providers. To reduce it, we propose a computer-aided, model-driven, service-specific tool, with support for collaborative work and for checking properties on models. We started by defining a prototype of the Meta-model (MM) of the service domain. Using this prototype, we defined a simple graphical modeling language specific for service designers. We are currently enlarging the MM of the domain using model transformations from Network Abstractions Layers (NALs). In the future, we will investigate approaches to

ensure the support for collaborative work and for checking properties on models."

– Communications in international peer-reviewed conferences without ISBN proceedings:
  1. [Chiprianov 10] CHIPRIANOV Vanea, KERMARREC Yvon, ROUVRAIS Siegfried. *Meta-tools for software language engineering : a flexible collaborative modeling language for efficient telecommunications service design.* FlexiTools: Workshop on Flexible Modeling Tools at the 32nd ACM/IEEE ICSE Intl. Conf. on Software Engineering, 02 may, Cape Town, South Africa, 2010.

     Abstract: "The increasingly competitive environment pressures telecommunications service providers to reduce their concept-to-market time. This time is influenced by a multitude of factors. For the benefit of telecom service designers, this paper focuses on increasing the degree of automation, offering team collaboration capabilities and bridging heterogeneous technologies. To address these factors, we propose a model-based meta-tool approach, which rapidly and iteratively generates particular tools for software languages. Each language is specific to one of the viewpoints involved in the definition of a service, as identified in the Intelligent Network Conceptual Model. A flexible language prototype for service designers, that blends a higher degree of formality with creative freedom, has already been implemented. The integration of first collaboration capabilities, defined and tooled, into this language, by including the rationale behind the designers' decisions, is currently being pursued. A second language prototype, for network designers, together with syntactic and semantic (partial) automatic interoperability between these two viewpoints, are also proposed."
  2. [Chiprianov 09a] CHIPRIANOV Vanea, KERMARREC Yvon. *An approach for constructing a domain definition metamodel with ATL.* 1st International Workshop on Model Transformation with ATL , 08-09 july, Nantes, France, 2009.

     Abstract: "Present day Telecommunications competitive market requires a rapid definition process of new services. To ensure this, we propose to replace the current paper-based process with a computer-aided one. Central to this later process is an information model that captures domain specific knowledge. We approach its construction by defining model querying and model transformation rules in ATL over existing network abstraction layers. We also report on the way we used ATL to define these rules and the benefits of doing so, and pinpoint issues that may be addressed in future ATL releases."

– Communications in national peer-reviewed conferences with and without ISBN proceedings:
  1. [Chiprianov 11c] CHIPRIANOV Vanea, KERMARREC Yvon, ROUVRAIS Siegfried. *Practical Model Extension for Modeling Language Profiles. An Enterprise Architecture Modeling Language Extension for Telecommunications Service Creation.* Journées nationales IDM, CAL, et du GDR GPL, 07-10 june, Lille, France, 2011, pp. 85-91, ISBN 978-2-917490-15-0.

Abstract: "Model Driven Engineering aims at changing the focus from code to models. To achieve it, enabling model transformation is essential. A type of transformation is meta-model extension. It is particularly salient for the use of models in defining Domain Specific Modeling Languages, especially for profiling existing languages. Meta-models describing language syntax have a low number of components. Accordingly, an expert-driven approach to extending meta-models is both practicable and preferable to an automatic one, which has a higher level of inaccuracy. We propose in this paper three principles for aiding an expert in practically extending meta-models with domain specific concepts. The resulted language profiles are backwards-compatible. We apply these principles to defining an ArchiMate profile for telecommunications service creation."

2. [Chiprianov 09b] KERMARREC Yvon, CHIPRIANOV Vanea. *Model-based DSL Frameworks : A Simple Graphical Telecommunications Specific Modeling Language*. 5èmes journées sur l'ingénierie dirigée par les modèles (IDM), 25-26 march, Nancy, France, 2009, pp. 179-186.

   Abstract: "Since 1989, when it was first formulated, the problem of service and/or service feature interaction in Telecommunications has been widely addressed. We investigate the use of Model Driven Engineering (MDE) to provide an approach for the design of large-scale distributed systems that enables validation of system properties (e.g.; absence of undesirable service interactions). The first step to this design method is the definition of a modeling language which enables formal definition of services and their interactions. We describe here the experience we gained with MDE through the definition of a simple graphical Telecommunications specific modeling language (SGTSML), developed in the context of a master thesis."

# List of Figures

# List of Tables

# Introduction

"The lord to the Living One's Mountain did turn his mind, 1
the lord Bilgames to the Living One's Mountain did turn his mind,
he called to his servant, Enkidu:
'0 Enkidu, since no man can escape life's end,
I will enter the mountain and set up my name.' "

Bilgames and Huwawa: 'The lord to the Living One's Mountain', in
*The Epic of Gilgamesh. A new translation*, Andrew George, Penguin Books, 2000

Competitiveness in telecommunications has been dramatically increased by the deregulation of the telecommunications market, the increased involvement of the End User in the telecommunications service construction process and by the more accentuated convergence with the Internet and with software development. This means that the productivity of different actors involved in telecommunications service creation must be increased, while the quality of service must be maintained at least at the same level. How could the concept-to-market time of new telecommunications services be reduced, while affecting non-negatively other parameters like cost, quality of service (QoS), quality of experience (QoE), designer creativity, user satisfaction? In this thesis we propose a process and software tools to answer this question. We illustrate our proposal using a multimedia conferencing telecommunications service.

We start with an introduction of the concept of telecommunications service. We continue with a brief history of telecommunications, from the ones based on circuit switched to the ones based on packet switched networks. We insist on the challenges that the current convergence between these two types of networks, together with the telecommunications market liberalization, raises for ex-national telephone service providers. From these challenges, we derive the research questions that drive this thesis. Next, we indicate our major contributions to finding an answer to these questions. We finish by presenting the organization of the document and making connections between chapters and research questions by roughly indicating where each research question is tackled.

## Telecommunications Service

### Service Definition

The modern meaning of the term "service" comes from the 1930s U.S. Department of Commerce's Standard Industrial Classification (SIC) codes, which used it to describe one of the three main economic sectors at that time: agriculture, manufacturing and service. From the initial usage as an all-embracing term for the activities that were not classified in the other two sectors, the term has evolved almost beyond having any meaning, designating numerous activities that represent 75% of the U.S. gross domestic product (GDP)

[Pal 05], or that play a similarly important role in all of the Organization for Economic Cooperation and Development (OECD) countries [Demirkan 08]. However, attempts of defining it include that of T. Hill [Hill 77], which has also been accepted by the U.S. government as the basis for defining service products in the North American Product Classification System (NAPCS) [Mohr 02]:

"A service is a change in the condition of a person, or a good belonging to some economic entity, brought about as the result of the activity of some other economic entity, with the approval of the first person or economic entity." Other definitions of the term "service" are indicated for example in [Chesbrough 06].

T. Hill [Hill 77] classifies services into private and public (collective, pure public); into those affecting goods and those affecting persons (sub-divided into mental and physical); into labor, non-labor and capital services; permanent and temporary; reversible or not. Hence, one can notice that the term "service" has a vague meaning, encompassing a large number of terms and a definition is difficult to delimit.

## Service Importance

As the economies of the world make the transition from agriculture and manufacturing to services [Spohrer 08], services become more and more important. Market-based services, excluding those provided by the public sector (e.g., education, health care) represent 50% of the total value added in the OECD countries, and are becoming the main driver of productivity and economic growth, especially as the use of Information Technology (IT) services is growing [Spohrer 05]. Most of the past interest has focused on marketing or management of services. However, technology-enabled services are more and more profitable for companies [Demirkan 08]. This growth in services economy causes a growth in services research.

## Telecommunications Service Definition

Out of all the activities that are designated as services, we focus on telecommunications services. Telecommunication is the transmission of information over significant distances [1]. The term telecommunications service, as defined by [FCC 96], term 51, means "the offering of telecommunications for a fee directly to the public, or to such classes of users as to be effectively available directly to the public, regardless of the facilities used." The telecommunications industry delivers telephone (wired, cellular), television (cable, satellite), Internet (broadband, mobile), and other services to customers. Providing the primary means of communication to virtually all businesses, households, and individuals, telecommunications firms supply an essential service to the economy [2].

We notice that the definition is large, including any content of telecommunications, from voice (i.e. telephone), to synchronized video, text, file transfer, etc between two or more participants. These services can be offered through several types of facilities, either using classical switch-based network, or through Internet or other kinds of networks. Furthermore, there is no restriction as to whom can offer such telecommunications services, although key actors have created niches and standards.

1. http://en.wikipedia.org/wiki/Telecommunication , accessed on 06.11.2011
2. http://www.bls.gov/oco/cg/cgs020.htm , accessed on 06.11.2011

Constructing a telecommunications service involves several points of view: that of the End User, who determines its value and who uses it, and those of the various actors involved in the value network that provides the service [Basole 08]. As the End User is the one that determines the value of the service, his needs and expectations are very important for the providers of that service. Integrating the End User's needs into the construction cycle of a service and ensuring that they are fulfilled becomes more important than in the past. The functionalities (ensured by a software and hardware system) of a service, although still essential, are not enough to meet the End User's expectations; the business model and quality properties (e.g., cost, quality of service) are very important as well.

## Complexity of the Service Value Network

However, the service (in general, not only telecommunications) value is created in an ecosystem - a service value network. The structure and dynamics of the value network increase the complexity of the services ecosystem. The Information and Communications Technology (ICT) plays a central role in reducing complexity for End Users, by providing greater levels of value network integration, information visibility, and means to manage and anticipate change [Basole 08]. Subsequently **main services research is focused on Information and Communications Technology**.

In the context of telecommunications, the complexity of the service ecosystem comes from the complexity of the services themselves, the numerous applications that interact with each other (e.g. the feature interaction problem [Bowen 89]), the functional dependencies between them and their expected properties (e.g., quality of service, performance, reliability, availability). All of these are difficult to meet.

In conclusion, services are an important part of the economy of developed countries and are becoming an ever more important part in the economies of developing countries. Among these, telecommunications services are essential to the economy, as they represent the primary means of communication between different economic entities. As the End Users are the ones who decide the value of a service, their role is increasingly more important in the service creation cycle. This requirement of increased involvement of the End User, in conjunction with important changes in the Information and Communication Technology, and with market deregulation are the catalyst of revolution in the telecommunications industry.

# TELECOMMUNICATIONS SERVICE CONSTRUCTION CHALLENGES IN A SOFTWARE WORLD

The history [3] of electrical telecommunications starts with the electrical telegraph (1838, Samuel F. B. Morse); the next major development is the telephone (1876, Alexander G. Bell) network build with (at first mechanical and later) electric switches and copper wires - this is called a **circuit switched network**. After sound transmission (i.e. telephone),

3. after `http://en.wikipedia.org/wiki/History_of_telecommunication` and `http://bnrg.eecs.berkeley.edu/~randy/Courses/CS39C.S97/index.html` , accessed on 06.11.2011

in 1927, Philo T. Farnsworth demonstrates an early version of all-electronic transmission of silhouette images (later called television). Wireless telecommunications are developing in parallel: in 1893, Nikola Tesla describes and demonstrates the principles of wireless telegraphy; in 1896, Guglielmo Marconi is awarded the patent for radio. In 1947, Bell Labs proposes a cellular radio telephone network.

A major revolution is introduced by the development of **packet** (versus the existing circuit) switching (e.g. early 1960s, Paul Baran), which enables much more flexible and optimized use of network resources. The first computer network based on this technology, ARPANET, is established in 1969 between four USA universities. Following the merge of ARPANET with other networks, in 1981, the TCP/IP protocol is introduced, enabling the Internet. Protocols for e-mail, SMTP, and for hyper-linked Internet, HTTP, are introduced in 1982, and 1996 respectively. Internet access becomes widespread late in the 20th century, converging with the old telephone and television networks. With its multi-layered architecture, Internet enables the **separation** between (software) services and (hardware) networks on which services are deployed. This separation/layering enables actors to focus only on certain parts of the telecommunications service creation and also allows them to specialize and lower costs.

From a political point of view, telecommunications (including mail/post) is an extreme case of nationalization almost all over the world. Starting in Europe from the 1880s (e.g. in France, in 1889, the National Assembly decides to transfer the management of telephone to the State), nationalization spreads across the globe, leaving telecommunications under private ownership in only a few countries, most notably the USA. This changes drastically, in the 1980s, when the telecommunications industry experiences radical degrees of liberalization. Deregulation of the telecommunications market [FCC 96], [Lal 01], [Umali 02] dramatically increases competitiveness all over the globe.

Ex-national telecommunications providers have to face serious competition in the context of the convergence of telecommunications with the Internet, of the change from stove pipe ”silo” to multi-layered IP-based architectures [Arnold 10], and of market liberalization. They have to adapt to the new Internet standards, while integrating their traditional fix and mobile telephone networks with the Internet. To compete with actors from the Internet world (e.g., Skype™, Google®), they must adopt a new construction process for services, inspired from software development methods. They must increase their productivity, while the quality of the offered services must be maintained at least at the same level.

## Research Questions

The research problem that drives this thesis deals with reducing the construction time by orders of magnitude, (e.g.; from months or weeks to days, even hours [Pan 08]; or from 18 to 3 months in other appreciations [Bo 10b]), while affecting non-negatively other parameters (e.g., cost, quality of service - QoS, quality of experience - QoE, designer creativity) of the new telecommunications services offered by telecommunications providers. To tackle this broad theme, we divide it into three research questions:

1. *RQ 1 Construction process*: What construction process should telecommunications providers follow to meet the challenges introduced by the convergence with the Internet, the change from stove pipe ”silo” to multi-layered IP-based architec-

tures, the increased involvement of End Users in the construction process?

2. *RQ 2 Software tools*: Many stakeholders/actors play a part in a telecommunications service's life-cycle. Among these stakeholders, we focus on Service Providers and Developers. After having proposed a construction process to answer the previous *RQ 1 Construction process*, the next question is what software tools should Service Providers and Developers employ to implement/follow this process? A high degree of automation, together with support for team work and testing/verification will reduce the integration time, increase software quality, therefore ultimately reducing the construction time and increasing overall quality.
3. *RQ 3 Tool building process*: However, software tools may be expensive and may take a long time to develop. Also, they must be adapted to the specific needs of Service Providers and Developers, and make use to the maximum of their specific capabilities. That is why it is important to determine what tool building process Tool Vendors should use to provide software environments to Service Providers and Developers?

## Contributions

In this thesis we address the previous research questions and propose answers to them. Concerning the *RQ 1 Construction process*, we propose a waterfall-like process, in which a stakeholder converses only with the stakeholders immediately upstream and immediately downstream from them in the telecommunications service life-cycle. Among the numerous responsibilities of a stakeholder, we focus on four: modeling, testing, collaborating and inter-operating. Each stakeholder/role describes the telecommunications service from their viewpoint, according to their concerns. Each view models the telecommunications service using a dedicated Modeling Language. This model is then tested through simulation. Of course, several professionals work in each role. They interact with each other and with professionals from other roles. They also import/integrate their models. The most important challenge introduced by this process is the interoperability of models developed in different views.

To enable stakeholders to follow this process, we build on Enterprise Architecture frameworks and Modeling Languages. These offer a unified vision of an enterprise and its product, necessary to stakeholders. We profile an Enterprise Architecture Modeling Language with domain specific models for Service Providers and Developers. For this profile we provide a graphical editor, code generation, and translation towards a network simulator - essential *RQ 2 Software tools*. Moreover, we offer a Modeling Language for capturing decisions and the arguments behind them. Similarly, Domain Specific Modeling Languages and connected software tools for other roles can be offered. In conclusion, among the numerous software tools needed by telecommunications service stakeholders, we focus on tools for the four activities that form the construction process. We also focus on two of these stakeholders, the Service Provider and the Service Developer.

The *RQ 3 Tool building process* we use is based on Model Driven Engineering, a subfield of Software Engineering. It provides an approach for defining Modeling Languages that enables automatic generation of the graphical editor, translation towards other Modeling Languages and code generation towards programming languages. This tool building process is focused on the software tools that we advance for the four activities that form

our telecommunications service construction process. Model Driven Engineering is based on models, therefore, quality is improved and software tool construction time is reduced.

## ORGANIZATION

We begin by presenting the state of art in telecommunications service creation, the current approaches to answering the research questions (cf. Section Research Questions), in Chapter 1. We present the service life-cycle and the actors involved in it. These actors have various needs related to the software which they use to describe the telecommunications service from their point of view. This software is generally called a Service Creation Environment. Additionally to offering dedicated tools to each actor, this environment should provide an integration/interoperability mechanism between the tools. We present, categorize and compare existing approaches to answer the requirements of two of the actors (Service Providers and Service Developers), and we indicate their limits.

In order to offer a solution for these limitations, we investigate two fields, that of Enterprise Architecture, in Chapter 2, and that of Model Driven Engineering, in Chapter 3. One of the main requirements of Service Providers, regarding the Service Creation Environment, is to have an overall graphical model of service construction which integrates all business, management, and technical activities (cf. *Req 1 An overall model*). This type of representation is one of the goals set for Enterprise Architecture frameworks and modeling languages (Chapter 2). We present and discuss the main frameworks and Modeling Languages of Enterprise Architecture, from which we select TOGAF and ArchiMate as the most appropriate for our purpose. One of the main features of Enterprise Architecture frameworks is tackling complexity through viewpoints. While this reduces complexity of one view to a manageable size, it introduces interoperability issues between views and their dedicated software. Both advantages and limitations of applying Enterprise Architecture to telecommunications service creation are discussed.

Another main concern of both Service Providers and Service Developers is related to having software tools specialized for their tasks and domain (cf. *Req 2 Domain specificity*). One manner of producing these tools is through the Model Driven Engineering approach, presented in Chapter 3. Moreover, Model Driven Engineering is based on models and generative techniques; hence it enables one to rapidly obtain prototypes of telecommunications services. The chapter ends with a general discussion on the advantages and limitations of using Model Driven Engineering for telecommunications.

We continue by presenting our proposals to answer the identified research questions and limits of existing approaches. As an answer to *RQ 1 Construction process*, in Chapter 4 we introduce a telecommunications service construction process inspired from the software waterfall development cycle. This waterfall-like telecommunications service creation process reflects current practices in the telecommunications industry, where a telecommunications service is defined using a stove pipe ”silo” approach, without going back to a previous phase, and in isolation from other telecommunications services. As such, this process should be easily accepted by practitioners. It consists of each role doing the same activities, after which the model goes downstream, to the next role, who will perform the same set of activities.

To build the dedicated tools for each role, we propose a model-driven tool building process, in Chapter 5. This is based on models and offers a high automation degree; there-

fore, it allows Tool Vendors to build the software tools more rapidly. It is also noteworthy that the same process can be used to build the tools for all and any of the stakeholders. In fact, this process can be used to define a family of Domain Specific Modeling Languages and associated tools.

Tools used by designers in this construction process are presented in Chapter 6. We focus on defining Domain Specific Modeling Languages and associated tools for the Service Provider and Service Developer roles. Based on our choices of TOGAF and ArchiMate, we define domain specific profiles for ArchiMate by using the building process proposed in the previous chapter. These profiles are dedicated to Service Providers and Service Developers. We also build software tools for these profiles, tools that constitute parts of the Service Creation Environment for Service Providers and Service Developers. We finish by discussing the advantages and disadvantages of the software tools we propose.

To better show the place of the proposed tools in the Service Creation Environment and demonstrate their capabilities, we provide an application of the telecommunications service construction process and tools to a complete multimedia conferencing service case study, in Chapter 7. The conferencing service has been the subject of significant amount of research (cf. e.g. the related work mentioned in Section 7.5). This shows that our proposals work towards reducing construction by orders of magnitude, while maintaining at least the same level of quality and not increasing the cost.

# PART I : STATE OF THE ART

To answer the research questions identified in Section Research Questions, we first review existing approaches for telecommunications service creation, in Chapter 1. We identify limits of these approaches and investigate two fields, that of Enterprise Architecture, in Chapter 2, and that of Model Driven Engineering, in Chapter 3, that can provide solutions to overcoming them.

TELECOM Bretagne

# 1 Telecommunications Service Creation

”With all the commotion Bilgames disturbed Huwawa in his lair, 66
he launched against him his auras of terror.
Bilgames was overcome [with a stupor], as if in a sleep,
Enkidu was beset [with torpor], as if in a daze.”

Bilgames and Huwawa: ’The lord to the Living One’s Mountain’, in
*The Epic of Gilgamesh. A new translation*, Andrew George, Penguin Books, 2000

As we have presented earlier, defining a telecommunications service involves several points of view: that of the end consumer, who determines its value, and those of the various actors involved in the value network that provides the service. In this chapter we present the latter and their place in this network in more detail. To perform their specific tasks, each actor needs dedicated software tools. We analyze existing approaches and indicate limitations for two of these actors, the Service Provider and the Service Developer, which operate in the Construction phase of the telecommunications service life-cycle.

## 1.1 Telecommunications Service Life-cycle

To understand the value network which provides a telecommunications service and the place different stakeholders occupy in it, it is necessary to first have a global understanding of the phases through which a telecommunications service passes, from its conception, to its withdrawal. For this purpose, we investigate in this section what is commonly called the ”life-cycle” of a telecommunications service. All services go through a life-cycle, which contains descriptions of activities, in the form of an ordered collection of processes or steps, that are required to support the development, the operation, and the maintenance of a service [Demestichas 99].

### 1.1.1 The ”Traditional” Life-cycle

Many proposals were made for the description of the telecommunications service life-cycle. For example, the life-cycles in one category share many similarities and are called ”traditional” life-cycles (e.g. by [Pan 08]).

For example, a product life-cycle management process [Georgalas 09] defines five steps: Concept, Design and Develop, Deploy, Operate, Retire. The TINA-C service life-cycle [TINA-C 97] defines five stages: Need, Construction, Deployment, Utilization, Withdrawal.

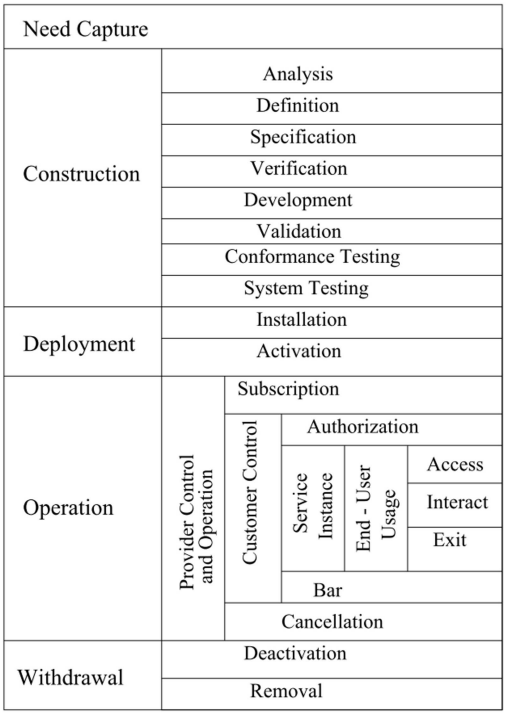


Figure 1.1 : Telecommunications service life-cycle, from [Berndt 94].

Even if they propose slightly different names for the five phases, these life-cycles are very similar. We present here the telecommunications service life-cycle proposed by [Berndt 94], on which the TINA-C life-cycle [TINA-C 97] is based, Figure 1.1, and adopt their terminology. The five phases are:

- The *Need Capture* phase: includes activities for eliciting and analyzing requirements of the End User.
- The *Construction* phase: includes all the off-line activities required in designing and developing the software and any special hardware associated with a service. It can be further broken down into sub-phases that resemble those of a traditional software development life-cycle (cf. Section 1.1.3).
- The *Deployment* phase: is responsible for introducing a pre-defined service into an environment. It encompasses the installation of its constituent parts, and concludes with the activation.
- The *Operation* phase: encompasses aspects of service management, such as context negotiation, and service transaction. It does not, however, encompass deployment and withdrawal. It is broken down into service provider activities, customer organization activities, and end-user activities.
- The *Withdrawal* phase: is responsible for removing a service from an environment without negatively impacting other live and dormant services. This activity encompasses planning, de-activation, de-installation and/or de-commissioning of its constituent parts, and testing for adverse consequences. At the conclusion of the activity, the service is no longer available to the service provider.

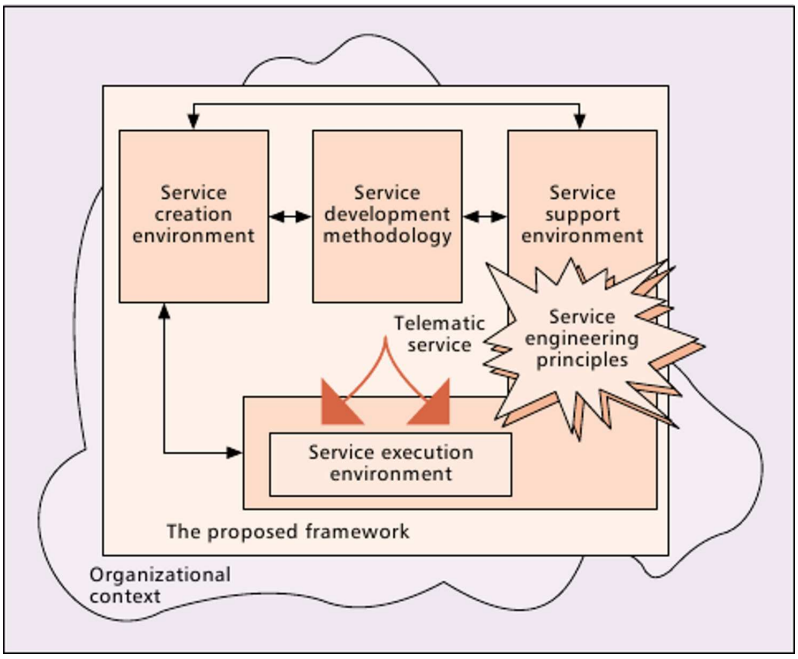


Figure 1.2 : Telecommunications service engineering framework, from [Adamopoulos 02].

In what follows, we use this terminology when discussing the telecommunications service life-cycle.

### 1.1.2 The Telecommunications Service Engineering Framework

To enable the telecommunications service life-cycle, a framework, a suite of software tools, is needed. One generic proposal is the framework (cf. Figure 1.2) from [Adamopoulos 02]. It consists of:

- A *service development method*: According to the telecommunications service life-cycle we adopted, it is a service creation/construction (not only the *Development* sub-phase) method (for definitions of *methodology*, *method* and *process* adopted in this thesis, cf. Section 4.1). It guides the stakeholders in a systematic and structured way.
- A *service creation environment (SCE)* (referred to in this thesis as Service Creation Environment): A collection of software tools used according to the service creation process. Its aim is to automate and simplify as much as possible the service creation process, and facilitate consistency and verification checks. It corresponds mainly to the *Construction* phase.
- A *service support environment*: In the initial proposal, it consisted of two main components:
  - *Service engineering principles*: These are concepts, guidelines, practices applicable to service engineering activities.
  - *A service execution environment* (referred to in this thesis as Service Logic Ex-

ecution Environment): Encompasses the computing and network infrastructure and the appropriate software needed for and during the execution of services. It enables the *Deployment* phase; this is where services are deployed.

It was extended to support the *Operation* phase as well (cf. Business Support System (BSS) and Operations Support System (OSS) [TM Forum 10]). Considering this extension, the SCE should also enable the construction of (or at least the connection with) components that support the *Operation* phase.

Figure 1.2 also presents the framework in an organizational context. The connections between the service and the larger organization should also be described.

In what follows, we will focus on the *Construction* phase and the Service Creation Environment.

### 1.1.3 The Construction Phase of the Life-cycle



The scope of this thesis is focused on the Construction phase from a telecommunications service life-cycle. The reasons are twofold. Firstly, there is a particular personal interest of the author of this thesis in design, architecture and implementation. Secondly, this thesis was begun in collaboration with a major telecommunications operator, which plays the roles of Service Provider and Service Developer, roles which operate (cf. Section 1.2) in the Construction phase.

Similarly with the entire ”traditional” telecommunications service life-cycle, many proposals were made to describe the Construction phase as well. An enriched variation of the TINA-C service life-cycle model is presented by [Adamopoulos 00], focusing on the Construction phase. This enriched variation is further refined for the Design/Specification phase by [Adamopoulos 09]. A ”traditional” Construction phase, according to [Pan 08], generally consists of 5 phases: Concept and Definition, Design, Development, Testing and Deployment. Whereas there are differences (e.g. [Pan 08] considers Deployment as part of the Construction phase), the sub-phases of the Construction phase from different proposals are again quite similar. We continue presenting (Figure 1.1) and adopting the sub-phases proposed in [Berndt 94], based on [Chapman 93]:

- The *Analysis* sub-phase: describes in a structured way the relevant requirements, obligations, and synchronization policies from the different stakeholders that are involved.
- The *Definition* sub-phase: provides a description of the desired effect of the service as seen by various stakeholders. The description specifies what the service will do and not how it will be implemented. It describes the semantics of the problem domain for which the service is designed. It identifies the objects that compose a service, the different types they belong to and the relations between them.
- The *Specification* sub-phase: provides a formal and unambiguous description of the service. It should specify how the service can be achieved (implemented). Computational modeling takes place in this sub-phase. The service is described in terms of computational objects interacting with each other.
- The *Verification* sub-phase: analyzes the correspondence between the service specification (Specification sub-phase) and the initial requirements (Analysis/Definition sub-phases).

- The *Development* sub-phase: comprises the implementation of the software and hardware modules needed for the specified service. The pieces of the service software (computational objects) are defined and implemented in an object-oriented programming language (e.g. Java).
- The *Validation* sub-phase: the implemented service (software and possible hardware modules) is subjected to a variety of tests in order to ensure it operates correctly and reliably.
- The *Conformance Testing* sub-phase: comprises checking the implementation for conformance to architectural rules and standards expressed in the design. Thus it checks the correspondence between the outcome of the Development sub-phase versus Definition sub-phase versus Specification sub-phase.
- The *System Testing* sub-phase: comprises the testing of software and hardware modules in an (possible) operational environment.

In this thesis, we focus especially on the Definition, Specification, Verification and Development sub-phases.

### 1.1.4 Other Life-cycles

Other life-cycles, that extend or modify the ”traditional” one, have been proposed. For example, an extended telecommunications service life-cycle that is template-based [Pan 08], proposes 4 phases: Service Maturity Modeling, Immature Service Creation, Maturing Service Promotion or ”Templatization”, and Mature Service Creation. The Immature Service Creation phase corresponds to the ”traditional” life-cycle. Other proposals build on iterative software development methods.

In this thesis we will follow the ”traditional” life-cycle model, and especially its Construction phase, as it is closer to the current practices of the telecommunications industry. Based on it, we propose a process for telecommunications service construction (cf. Chapter 4). Other iterative telecommunications service construction processes, based on iterative life-cycles, are discussed as perspectives (cf. Section 7.5.4).

## 1.2 Roles in Telecommunications Service Construction

Different actors/stakeholders participate in the various phases that form the longtelecommunications service life-cycle. Each stakeholder has a different perspective and different objectives. To simplify this situation, roles have been identified to allow separation of the different perspectives of the concerned organizations/stakeholders. There are, for example, roles that focus on business modeling, like the ones proposed by [TINA-C 97]: Consumer, Broker, Retailer, Third-party service provider, Connectivity provider. However, there are proposals that cover the entire life-cycle, like [Hållstrand 94]:

- *End User*: the actual user of a service; primarily interested in service functionality, quality and cost.
- *Service Subscriber*: the person or organization who contracts and pays for the service. It may need to customize and change services.
- *Service Provider*: the organization that provides value-added services. The traditional Operator is not the only one acting as Service Provider; separate (network-

independent) organizations do this as well. The Service Provider is primarily interested in considerations related to the End User (e.g. anticipating and determining requirements for new services, promotion of services in the marketplace) and deployment (e.g. validation of services against requirements, testing of services in the target network, the interaction between new and existing services in the network).
- *Service Developer*: the organization that performs all the actions necessary to create a new service. They are interested in rapid prototyping and customization of services.
- *Network Provider*: the organization that operates, administers and maintains telecommunications networks; provides bearer and management services.
- *Manufacturer*: the traditional telecommunications manufacturer which provides a platform upon which services can execute. They are interested in providing reliable, open and flexible platforms.
- *Tool Vendor*: the organization that produces tools which are used by the various actors in the service industry. They are interested in producing tools that cover specific tasks but are adaptable to the needs of each organization/role.

We adopt this terminology regarding the roles involved in the telecommunications service life-cycle in our thesis.

These roles intervene in different phases of the telecommunications service life-cycle [Combes 99]. The End User, the Service Subscriber and the Service Provider may be involved in the Need Capture phase. The main roles involved in the Construction phase are the Service Subscriber, the Service Provider, the Service Developer, the Network Provider and to some extent, the Manufacturer. The Deployment and Withdrawal phases involve mainly the Service Provider and the Network Provider. The Operation phase involves the Service Subscriber, the Service Provider, the Network Provider and sometimes the End User. The Tool Vendor role is a transversal one, offering tools to all other roles, all through the life-cycle.

In this thesis, we focus on the Service Provider and Service Developer roles. The main reason for this is is that this thesis was begun in collaboration with a major telecommunications operator (BT) that plays these roles. We also focus on the Construction phase from the telecommunications service life-cycle; therefore, we can refine the scope of this thesis even more, to the level of the sub-phases of the Construction phase. In this phase, the Service Provider is active in the Definition, Specification and Verification sub-phases, and the Service Developer mainly in the Development sub-phase. The Tool Vendor is also, of course, present. These are therefore the sub-phases and roles that we focus on especially in this thesis. We will also sometimes generalize solutions/approaches to the entire life-cycle and other/all roles.

## 1.3 Requirements for Service Creation Environment

Having focused on the Construction phase from the telecommunications service life-cycle and the Service Provider and Service Developer roles, in this section, we want to detail their requirements/needs in this phase. Requirements are an important manner to capture the needs of stakeholders. However, by their nature, they may fail to capture the most essential and important needs of stakeholders. Reasons for this include the fact that in some cases, stakeholders themselves do not know what they need. In an attempt of tackling these inherent limitations of requirements, we have compiled the following list

from different literary sources. Moreover, we have tried to have as many requirements as possible sustained by several literary sources. These requirements have been grouped in the need for a computer aided tool, generically called a Service Creation Environment. Essential requirements for this Service Creation Environment comprise:

1. *Req 1 An overall model*: Service Providers are interested in selling and marketing their services. A graphical model describing the service creation process from a marketing standpoint is crucial [Kosmas 97]. This model should include all business planning and economic analysis activities. In addition, network planning should also be represented through a separate model. These models should be integrated with the model of the technical activities involved in development so that an overall model of service creation taking in all business, management, and technical activities is obtained [Hållstrand 94]. The management of new telecommunications services requires a total mind shift, from traditionally separated stove pipe management island (i.e. the flow of information is restricted through rigid lines of control) towards unified, enterprise-wide management systems [Blum 09b]. This mind shift consists in a change from a focus on just creating a service, towards the unification of the creation process with business processes and management processes. This is a wider vision. To support it, the capability of describing a service at the business level, and together with its management, is needed. The unification of service creation processes, business processes and management processes is needed [Blum 09b].
2. *Req 2 Domain specificity*: Service Providers want to reduce the concept-to-market time, as the market windows are decreasing from years to months. One way to reduce it is by using tools specialized for the task and domain, e.g. for specifying service functionality [Hållstrand 94].
3. *Req 3 Rapid prototyping*: Both Service Providers and Service Developers need a rapid prototype of a service [Pavan 07], in order to assess its potential market success [Hållstrand 94], [Khlifi 08]. This can be achieved using a high degree of automation [Kosmas 97]. Rapid prototyping is also needed to integrate feedback from users.
4. *Req 4 Collaborative support*: Both Service Providers and Service Developers need collaborative team support, so that their personnel can communicate [Kosmas 97].
5. *Req 5 Early verification/simulation*: Service Developers need early verification, incompatibility detection, error detection, simulation [Pavan 07] as early as possible, to guarantee that the operation of the new service will not produce unwanted side effects within a distributed telecommunications context [Kosmas 97]. It is noteworthy that service execution within a ”model” environment is in some cases totally different from the one produced within a real environment which includes competition for resources or conflicting interests. Hence, different aspects must be considered during the Verification, Validation and System Testing sub-phases.
6. *Req 6 Integration*: All roles need integration/interoperability with existing standard and domain specific tools. The Service Creation Environment is basically a platform that uses a library of components and established ”plug-in” methods [Kosmas 97].
7. *Req 7 Reuse*: A library of components, or high level APIs, must be established [Pavan 07]. A new service must be developed as part of classes of services (thus reusing parts of other services), not standalone [Kosmas 97].
8. *Req 8 Wide range of services*: Service Providers must support services for different types of networks [Pavan 07], both circuit and packet switched. In this way, they can

provide a single solution platform [Khlifi 08].

9. *Req 9 Easy evolution of services*: Service Providers must be able to change the service logic rapidly and efficiently, to answer user expectations [Yelmo 08]. Services should be defined independently of a specific network technology. The current vertically-integrated networks need migration to horizontally-layered structures offering open and standard interfaces [Falcarin 08] (i.e. a service platform, based on shared services and network enablers, which can be easily composed in a loosely coupled manner [Pollet 06]). This layering enables support for multi-player (open) environments [Khlifi 08].

It is important to make the distinction between the Service Creation Environment, which is intended for the Construction phase of a telecommunications service, and the Service Logic Execution Environment, which is intended for the Operation phase. A Service Creation Environment is generally a graphical user interface (with constraints to ensure correctness) for developing services using predefined components, also called building blocks. It consists of tools used (normally) at design time to create the executable telecommunications services. A Service Logic Execution Environment is the environment (e.g. platform, enablers) in which the services are actually executed. It provides a set of common functions required for the execution of the services. Many of the concepts used (at design time) in a Service Creation Environment relate to the execution platform, enablers, (i.e. the Service Logic Execution Environment), therefore the two are related, but clearly separated. After Development of the telecommunications service, using the Service Creation Environment of course, all validations, verifications and testing can be done on a simulated/virtual environment.

The interested reader can consult other requirements, specific to the Network Provider, in [Ohnishi 07]. The software tool dedicated to the Network Provider is usually named Service Delivery Platform.

In what follows, we will investigate proposed Service Creation Environments, and analyze how they answer these requirements.

## 1.4 Major Initiatives for Service Creation

The Service Creation Environment proposals can be classified according to widely accepted standards/major initiatives. These major initiatives introduce several levels of abstraction on top of the network, with the ultimate goal that the same Service Creation Environment be used for all types of networks.

### 1.4.1 Major Telecommunications Initiatives for Service Creation

Major Telecommunications initiatives have introduced several abstraction layers, to separate the service from the underlying network/platform specificity.

The Intelligent Network (IN) introduced in the early 1990s two layers, a separation between the service and the network [Study Group XVIII 10]. The service is described in terms of Service Independent Building blocks (SIBs) [ITU 93]. The Service Independent Building blocks encapsulate the underlying network capabilities, thus making the service independent from it. They can be composed in many ways to form new services. This means that even if the hiding of the supporting platform reduces the possibilities of using that

platform, the increased composition potentiality offered by these abstracted components increase the number of new possible services. The Intelligent Network also introduces the concepts of Service Creation Environment and Service Logic Execution Environment, to ease and speed up the development and respectively deployment of services.

Following the Intelligent Network, the Next Generation Network (NGN) is a standardization wave that further refines the abstraction layers. It is a broad term, used to describe key architectural evolutions. The technical vision of Next Generation Network is to reunite different networks into a single network, built around the Internet Protocol (IP). This network is a generic one, independent of services and thus enabling all kinds of services. A Next Generation Network architecture consists of four layers: access, transport, control and application. The Service Creation Environment is at the application layer. This layered architecture allows telecommunications services to be independent of control, transport and access mechanisms [Yelmo 08].

The IP Multimedia Subsystem (IMS) [3GPP 06] is a standardized Next Generation Network architecture that offers unified access to Internet, Public switched telephone network and mobile access networks, defined by the European Telecommunications Standards Institute (ETSI) and the 3rd Generation Partnership Project (3GPP). The IP Multimedia Subsystem is more like a real implementation of the Next Generation Network, whereas the Next Generation Network is more like a "philosophy" of how to design a network. The IP Multimedia Subsystem enables the delivery of a completely new range of Web/Telecommunications service compositions, independent from the utilized access network [Blum 09b]. The main protocol used by the IP Multimedia Subsystem is Session Initiation Protocol (SIP) [Rosenberg 02]. To describe the composition of Session Initiation Protocol based services, the SIP application composition mechanism [Cosmadopoulos 08] has been proposed.

The IP Multimedia Subsystem architecture (Figure 1.3) is a collection of functions linked by standardized interfaces. Many vendors follow the IP Multimedia Subsystem architecture closely and implement each function into a single node [Camarillo 08]. On the left side of Figure 1.3 are terminals (mobile, computer), typically referred to as the *User Equipment*. The remainder of the figure shows the nodes from the IP Multimedia Subsystem Core Network Subsystem:

- one or more user databases, called *HSSs (Home Subscriber Servers)* and *SLFs (Subscriber Location Functions)*.
- one or more SIP servers, collectively known as *CSCFs (Call/Session Control Functions)*. The CSCF is an essential node in IP Multimedia Subsystem. There are three types of CSCF:
  - *P-CSCF (Proxy-CSCF)*: the first point of contact in the signaling plane between the terminal and the network. All the requests initiated by the terminal or destined to the terminal traverse the P-CSCF.
  - *I-CSCF (Interrogating-CSCF)*: a Session Initiation Protocol proxy located at the edge of an administrative domain. When a SIP server follows SIP procedures to find the next SIP hop for a particular message, the SIP server obtains the address of an I-CSCF of the destination domain.
  - *S-CSCF (Serving-CSCF)*: the central node of the signaling plane. It performs session control and also acts as a SIP registrar (it maintains a binding between the user location and the user's SIP address of record).

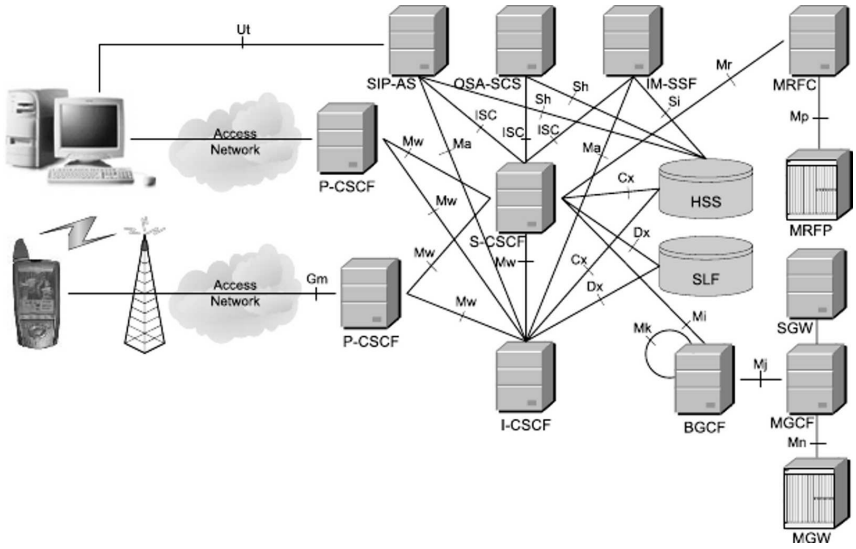


Figure 1.3 : The IP Multimedia Subsystem architecture overview, from [Camarillo 08].

- one or more *ASes (Application Servers)*: the entities that host and execute services.
- one or more *MRFs (Media Resource Functions)*, each one further divided into *MRFCs (Media Resource Function Controllers)* and *MRFPs (Media Resource Function Processors)*.
- one or more *BGCFs (Breakout Gateway Control Functions)*.
- one or more PSTN gateways, each one decomposed into an *SGW (Signaling Gateway)*, an *MGCF (Media Gateway Controller Function)*, and an *MGW (Media Gateway)*. PSTN gateways provide an interface toward a circuit-switched network, allowing IMS terminals to make and receive calls to and from the PSTN (or any other circuit-switched network).

IP Multimedia Subsystem offers access to Internet and mobile networks through the P-CSCF, and to PSTNs through the PSTN gateway. In this way, it reunites different networks in one.

Currently, Next Generation Network is widely used by various Service Providers. For example, BT uses it under the name 21st Century Network (21CN). According to BT [BT 10], 55% of the UK is covered by exchanges which have been enabled for 21CN. It is also used by AT&T [Taylor 08], under the name of AT&T U-verse, through which AT&T offers Internet access, Television, and telephone services in various parts of the USA [AT&T News 09]. In the Netherlands, KPN is developing a Next Generation Network in a transformation program called all-IP [KPN 11]. While the Intelligent Network is still used, the Next Generation Network is currently far more widely accepted.

The Open Mobile Alliance (OMA) [OMA 11] standardized service capabilities in the form of enablers. An OMA enabler is a management object around a service, a generic framework, consisting of requirements documents, architecture documents, technical specifications and test specifications. The same approach has been taken by the 3rd Generation Partnership Project (3GPP) and the International Telecommunication Union (ITU-T). They call OMA service enablers respectively service capabilities, service support capabil-

ities.

### 1.4.2 Major Web Initiatives for Service Creation

In parallel with the development of major initiatives that abstract the telecommunications networks and try to converge them with the Web, there have been proposals to converge the Web to the Next Generation Network and propose Service Creation Environments for these Web-based major initiatives.

The Web technologies which seem the most promising for telecommunications are the Web services and associated technologies (e.g. Service Oriented Architectures - SOA, Web Service Description Language - WSDL). Web services are software systems designed to support machine-to-machine interaction over a network [Haas 04]. They are independent of the programming language and of the execution platform. They can be considered as functions (services) described in a pivot language (Web Service Description Language) which enables their interaction. An important issue for Web services is composition/orchestration. One widely-used solution involves the orchestration language Web Services Business Process Execution Language (WS-BPEL) [Jordan 07]. There have been proposals to define telecommunications-specific Web services, like OneAPI (formerly ParlayX [ETSI 06]). ParlayX evolved the Parlay OSA API [ETSI 07] (cf. Section 1.5.1) into a simplified Web services interface for telecommunications network functionality.



## 1.5 Service Creation Environments

In this section, we present and classify some approaches for Service Creation Environments, based on the standards/major initiatives from the previous section. Our intention is to discern certain directions, not to provide a systematic review [Kitchenham 04].

### 1.5.1 Service Creation Environments for the Next Generation Network

Although there are Service Creation Environments developed for the Intelligent Network, like [Aubonnet 01], we focus in this section only on Service Creation Environments for Next Generation Network, because it is more widely used. In Next Generation Network IP Multimedia Subsystem, the Service Creation Environments are intended for the creation of telecommunications services in the IP Multimedia Subsystem Application Server component (cf. Section 1.4.1).

We classify Service Creation Environments according to their relation to Session Initiation Protocol, into the following categories:

#### 1.5.1.1 Session Initiation Protocol dependent

Examples of *Cat 1 SIP - dependent* Service Creation Environments include:

– The Common Gateway Interface (CGI) for Session Initiation Protocol [Lennox 01] is a mechanism through which a Session Initiation Protocol server can delegate the creation of telecommunications services to a standalone application. It is independent of the programming language in which the standalone application is written. Being protocol-dependent, it offers full access to protocol headers, thus it has the

potential of creating a wide range of services. Although at the time of introduction it was considered to be a Service Creation Environment, in the meantime the level of abstraction attached to Service Creation Environments has risen. Consequently, at present it is commonly considered as a mechanism for Service Logic Execution Environment.

– The Session Initiation Protocol servlet API [Cosmadopoulos 08] is a standard extension to the Java platform that defines the relationships between Session Initiation Protocol servlets and servlet containers. The servlet is a Java-based component that generates content. Therefore, the Session Initiation Protocol servlet API is used to generate dynamic content.
– The OMA Service Environment (OSE) [OMA 07] is a common abstraction layer for IP Multimedia Subsystem, specifying several service enablers and a general access function for $3^{rd}$ party service access.

#### 1.5.1.2 Session Initiation Protocol independent



Examples of *Cat 2 SIP - independent* Service Creation Environments include:

– Call Processing Language (CPL) [Lennox 04] is a language which describes and controls Internet telephony services. It is designed to be implementable on either network servers or user agents. Initially focused on Session Initiation Protocol, the latest version is meant to be independent of the operating system or the signaling protocol.
– Specification and Description Language (SDL), the current standardized version - SDL-2000 [ITU-T 07], although a revised version called DSL-2010 has been submitted for approval to ITU-T [Reed 11], is a language intended for unambiguous specification (i.e. the description of the required behavior) and description (i.e. the description of the actual behavior, the implementation) of telecommunications systems. It is part of a larger family of languages that include also: Message Sequence Chart (MSC) [ITU-T 11a], a trace language for the specification and description of the communication behavior of system components and their environment by means of message interchange; User Requirements Notation (URN) [ITU-T 08], intended for the elicitation, analysis, specification and validation of requirements; Testing and Test Control Notation (TTCN) [ITU-T 11b], intended for specification of test suites that are independent of platforms, test methods, protocol layers and protocols. SDL describes the structure and behavior of systems [Belina 89] [Braek 96]. SDL has been used especially in the 'Specification' sub-phase to develop libraries of reusable components that can be further specialized into telecommunications services [Sinnott 99] [Kolberg 99]. The use and impact of SDL in other phases: 'Need Capture', 'Construction' and 'Deployment', has been investigated in e.g. [Mullery 99]. [Mullery 99] concludes that SDL has advantages especially in terms of: formality, which enables subsequent validation and mapping to implementation technologies of telecommunications services described in SDL; tool support (current tools include IBM®Rational®SDL Suite™[IBM 11] - the ex-Telelogic SDL tool, for a history cf. [Reed 11]; PragmaDev Real Time Developer Studio [PragmaDev 11]; Cinderella SDL [Cinderella 11]). However, one of the main problems of producing SDL models is their level of abstraction [Mullery 99]. Modeling systems at a very low level of

abstraction (e.g. of Corba IDL objects) results in very complex models, for which is difficult to e.g. manage changes, visualize. This further impacts the complexity and scalability of tools (e.g. state space explosion for checking behavior).

- Parlay API/OSA [ETSI 07] is a specification of basic functions which enable the creation of services independent of the underlying networking technology and of the programming language. There are several sets of realizations for these specifications (e.g., for CORBA IDL, Java, Web Services - Parlay X). There are two categories of APIs in Parlay:
  - Framework: includes interfaces for trust and security management, event notification, service discovery, service registration, service subscription, integrity management.
  - Service: exposes the actual network capabilities. It includes interfaces for call control, user interaction, generic messaging, mobility, terminal capabilities, connectivity management, account management, charging, data session control, presence and availability management.

  Many proposals were made for the further development of Service Creation Environments on top of the Parlay API. An example of such a proposal is a higher-level Service Creation Environment [Glitho 03], in which high-level components that use Parlay API are defined and then combined.
- The JAIN SLEE [Ferry 08] is a specification for the internal architecture of a (Java) Service Logic Execution Environment, on top of which a Service Creation Environment can be build. A Jain SLEE service is a collection of reusable object-oriented components - service building blocks - running inside the Service Logic Execution Environment. The Service Creation Environment will specify the composition rules of the building blocks. There are several implementations of the JAIN SLEE, and even scalability and performance evaluations [Femminella 09]. An example of Service Creation Environment for composition based on BPEL is [Lehmann 09]; BPEL is translated into Java, and then the service is executed in an environment based on JAIN SLEE.

#### 1.5.1.3 Composition

In parallel with the Service Creation Environments for Web (cf. next section, Section 1.5.2), which focus mainly on the composition of (web-)services, proposals were made, i.e. *Cat 3 NGN composition*, to compose (Session Initiation Protocol) based services:

- A Session Initiation Protocol Servlet application composition mechanism [Cosmadopoulos 08] [Cheung 07] proposes a component that selects and invokes the next service in the composition - the application router. Some suggest [Ren 08] enhancing this mechanism with business rules.
- The Service Capability Interaction Manager (SCIM) [Gouya 06] proposes a mechanism for managing the interactions and the incompatibilities that may occur between several service capability invocations in IP Multimedia Subsystem. A BT - Alcatel Lucent proof-of-concept [Abhayawardhana 07] has been implemented for the SCIM.
- Telecommunications mash-ups: Mash-ups are small applications made through the composition of two or more services and contents [Yelmo 08]. Examples include:

– currently Ribbit API [1] (formerly BT Web21C). BT Web21C allowed external developers and businesses to build their own applications using a small set of telecommunications services, e.g. messaging, voice call and conference call. However, the creators must be technically skilled, because the services are composed through Java and .NET APIs. Other examples include: BlueVia [2] (formerly Telefonica's Open movilforum), Vodafone's Betavine [3], Orange Partner [4] and AOL (America OnLine) Developer Network [5]. A number of mash-ups are compared in e.g. [Yelmo 11].

Service Creation Environments for service composition are usually thought to be intended for End Users (cf. a survey [Laga 08]).

## 1.5.2 Service Creation Environments for the Web

These Service Creation Environments are based on the assumption that (basic) telecommunications services have already been defined. Value-added, composed services can be defined from these. This is why they focus on static composition/orchestration/-choreography as design mechanisms, which they apply to Web services or/and to Next Generation Network services. Sometimes, the Service Creation Environments claim to have a high enough ease of use to target directly the End Users. In conclusion, Web-based Service Creation Environments target rather the End User than the Service Provider and/or Service Developer:

### 1.5.2.1 Parlay X

Examples of *Cat 4 Parlay X* Service Creation Environments include:

– An approach to enable eXtreme model driven design of Parlay X-based communications services [Blum 09a].
– An automated method for Parlay X service orchestration [Kraemer 09].
– An integrated Service Creation Environment, with a mash-up toolkit, a simulation-based validation, a run-time adaptation tool, a personalized service provisioning environment, on top of Parlay X API [Kim 08].
– An approach for WS-BPEL choreography of Web services [Jie 09b]. In the Developer IDE, a list of Parlay X services are displayed to be composed.

### 1.5.2.2 Web Mash-ups

Like Telecommunications mash-ups, they are small applications made through the composition of two or more services and contents. However, the services are web services. Examples of *Cat 5 Web mash-up* Service Creation Environments include:

– [Braga 08] focuses on the mash-up of search services that offer the ability to carry out complex queries over the web.

1. `http://www.ribbit.com` , accessed on 06.11.2011
2. `http://bluevia.com/` , accessed on 06.11.2011
3. `http://www.betavine.net` , accessed on 06.11.2011
4. `http://www.orangepartner.com` , accessed on 06.11.2011
5. `http://dev.aol.com/` , accessed on 06.11.2011

- [Rosenberg 08] provides a lightweight language for RESTful service mash-ups.
- A Domain Specific Language solution for the service mash-up is proposed in [Maximilien 08], which provides facilities to help share and reuse mash-up components.
- WMSL (Web Mashup Scripting Language) introduced in [Sabbouh 07] enables the End User to build mash-ups with web page scripting.
- [Jung 11] uses knowledge support in terms of user experiences and activity-aware functional semantics to assist End Users in finding relevant open APIs.

The assumption that basic telecommunications services have already been defined is integrated both in the Parlay X and Web mash-up approaches. Therefore, they focus on composing these services.

### 1.5.3 Hybrid Service Creation Environments

Similarly to the web Service Creation Environments from the previous section, the hybrid Service Creation Environments, i.e. *Cat 6 Hybrid*, deal usually only with composing existing telecommunications services. It is assumed that the telecommunications services have already been constructed. Examples include:

- A Web-Telecommunications hybrid services orchestration and execution middle-ware [Bo 10b] based on the Web-Telecommunications based hybrid service bus (WTSB). WSTB contains two kinds of service engines: one is BPEL-based and the other follows the JAIN SLEE specifications.
- StarSLEE extends JAIN-SLEE. StarSCE is the Service Creation Environment on top of StarSLEE. StarSCE, however, helps developers transform a JAIN-SLEE service into a web-service [Falcarin 08].
- An approach [Niemöller 09] introduces concepts of aspect-oriented programming to service composition, to describe service features that are separated from its business core functionality. While the approach is applicable to both Session Initiation Protocol and Web Services, the authors advise that in practice SIP is less suitable because of the end-to-end sessions.
- OPUCE (the Open Platform for User-centric service Creation and Execution) [Yelmo 07] [Yelmo 08] [Sienel 09] proposes an architecture that allows the interoperation of telecommunications and IT applications. It proposes a Service Creation Environment which targets End Users, and presents test results to sustain its claims [Yelmo 11].
- [Jie 09a] applies the recommender system theory and technologies to assist the mash-up creator create hybrid Web-Telecommunications services.
- SPICE [Droegehorn 08] is a tool chain for service creation that also gives End Users the ability to create their own converged services.
- A service broker that mediates between $3^{rd}$ party applications and Next Generation Network service enablers [Blum 10].

Composition mechanisms for hybrid services include e.g. [de Vrieze 11], a proposal to extend End User programming with mash-ups for enterprise use, to support business processes.

### 1.5.4 Synthesis

In this section, we have broadly identified six categories of approaches for Service Creation Environment:

- three for NGN:
  - *Cat 1 SIP - dependent*;
  - *Cat 2 SIP - independent*;
  - *Cat 3 NGN composition*;
- two for Web:
  - *Cat 4 Parlay X*;
  - *Cat 5 Web mash-up*;
- *Cat 6 Hybrid*.

It is important to specify that the inclusion in one category or another does not always fully characterize the individual solutions. Many individual solutions combine several characteristics and may be included in several categories; the choice to include them in one category or another tries to reflect their main characteristics (as perceived from their presentations).



## 1.6 Comparison of Service Creation Environments

We compare the categories of Service Creation Environments identified in the previous section, based on the requirements identified in Section 1.3:

- *Req 1 An overall model*: None of the categories tackle it.
- *Req 2 Domain specificity*: NGN approaches have a high degree of domain specificity, with *Cat 1 SIP - dependent* being the most specific one. Web approaches are general, independent of domain, web services can deal with information from any domain; *Cat 4 Parlay X* is still specific to telecommunications though. Hybrid approaches still maintain a connection (e.g. by translation, bus, broker) with the domain.
- *Req 3 Rapid prototyping*: Basic (non-composed) services take a long time to define, and this is done usually with NGN approaches (*Cat 1 SIP - dependent*, *Cat 2 SIP - independent*). Composed services (*Cat 3 NGN composition*, *Cat 4 Parlay X*, *Cat 5 Web mash-up*,*Cat 6 Hybrid*) are usually defined much faster, with *Cat 5 Web mash-up* being probably the champion.
- *Req 4 Collaborative support*, *Req 5 Early verification/simulation*, *Req 6 Integration*: These aspects are independent of, but complementary to the service creation approach. They do not characterize categories, but rather individual approaches.
- *Req 7 Reuse*: As is the case for *Req 3 Rapid prototyping*, reuse is low when defining basic services (*Cat 1 SIP - dependent*, *Cat 2 SIP - independent*) and high when composing them (*Cat 3 NGN composition*, *Cat 4 Parlay X*, *Cat 5 Web mash-up*, *Cat 6 Hybrid*).
- *Req 8 Wide range of services*: Of course, NGN solutions are limited to NGN, whereas Web (provided that telecommunications services have been exposed as web services) and Hybrid address both platforms.
- *Req 9 Easy evolution of services*: Evolution is related to the effort of defining a service, the amount of reuse, and the range of platforms it covers. It will be more difficult to evolve NGN services. Composed services should be easier to evolve, as

| R \ A | NGN | | | Web | | *Cat 6 Hybrid* |
|---|---|---|---|---|---|---|
| | SIP (IMS) | | *Cat 3 NGN composition* | *Cat 4 Parlay X* | *Cat 5 Web mash-up* | |
| | *Cat 1 SIP - dependent* | *Cat 2 SIP - independent* | | | | |
| *Req 1 An overall model* | NO | NO | NO | NO | NO | NO |
| *Req 2 Domain specificity* | Very high | High | High | Medium | None | Medium |
| *Req 3 Rapid prototyping* | Very slow | Slow | Fast | Fast | Very fast | Fast |
| *Req 4 Collaborative support* | N/A | N/A | N/A | N/A | N/A | N/A |
| *Req 5 Early verification/simulation* | N/A | N/A | N/A | N/A | N/A | N/A |
| *Req 6 Integration* | N/A | N/A | N/A | N/A | N/A | N/A |
| *Req 7 Reuse* | Very low | Low | High | High | Very high | Very high |
| *Req 8 Wide range of services* | Limited to SIP | Limited to NGN | Limited to NGN | NGN (if exposed as web services) + Web | NGN (if exposed as web services) + Web | NGN + Web |
| *Req 9 Easy evolution of services* | Very difficult | Difficult | Easy (inside NGN) | Easy | Very easy | Easy |

Table 1.1 : Comparison of approaches for Service Creation Environments

only the composition of basic services evolves. However, compositions depend on the degree of integration of composing services - changes in one of the composing services of a tightly-integrated composed service may involve changes in other composing services as well.

The comparison is synthesized in Table 1.1.

## 1.7 Discussion

There are a number of conclusions that can be drawn from the analysis presented in the previous section on approaches to Service Creation Environments:

- *Req 1 An overall model* is not addressed by any of them. To tackle it, we investigate Enterprise Architecture in the next chapter.
- *Req 4 Collaborative support*, *Req 5 Early verification/simulation* and *Req 6 Integration* are related to the architecture of the Service Creation Environment. To ensure *Req 6 Integration*, an architecture focused on integrating several components (e.g. around a bus, based on plug-ins) would be highly recommended. Moreover, it would allow integrating components, tools (either existing - Off the Shelf components -, or new ones) for *Req 4 Collaborative support* and *Req 5 Early verification/simulation*.
- Approaches for Web, in contrast with those for Next Generation Network, have a high degree of *Req 7 Reuse*, a possible *Req 8 Wide range of services*, *Req 9 Easy evolution of services* and *Req 3 Rapid prototyping*. These qualities are highly desirable.
- Approaches for Web, in contrast with those for Next Generation Network, have a low degree of *Req 2 Domain specificity*. This is a major disadvantage. In light of the desirable qualities identified in the previous point, Web (in general IT) approaches, would be very beneficial, provided a means to include domain specificity is found. One such way is through Domain Specific Modeling Languages.
- Approaches for Web focus on composing basic services. Such IT solutions should be extended to the creation of the basic services as well.

In this chapter we have presented the telecommunications service life-cycle. We have focused on the construction phase and presented the actors involved in it. To construct the software part of a telecommunications service, we have identified requirements for a Service Creation Environment, a software tool that, ideally, offers to each actor the tools he/she needs for his/her specific tasks. Among the actors involved in the telecommunications service life-cycle, we have focused on the Service Provider and the Service Developer. We have discussed existing approaches to building the Service Creation Environment. While these approaches each have its benefits, none of them completely fulfills the identified requirements and there is one requirement which is not addressed by any of them. To answer these limitations, in the next two chapters, we investigate the fields of Enterprise Architecture and Model Driven Engineering, which we use in our proposals. We advance a Service Creation Environment, in Chapter 6. However, before building this Service Creation Environment, we need a tool-building process, which we propose in Chapter 5. Both the Service Creation Environment and its building process depend on the activities of the Service Provider and the Service Developer, therefore a telecommunications service creation process is proposed in Chapter 4. Finally, to show the applicability of our Service

Creation Environment, we use it in Chapter 7 to model a multimedia conferencing service.

# 2 Enterprise Architecture for Telecommunications Services

”He [Bilgames] took in his hands thirty shekels of oil and rubbed his chest, 86
he stood on the Great Earth like an ox [on all fours],
he bent his neck downwards and gave voice:
...
’Should I behave as if awestruck on the knees of Ninsun, the mother who bore me?’ ”

Bilgames and Huwawa: ’The lord to the Living One’s Mountain’, in
*The Epic of Gilgamesh. A new translation*, Andrew George, Penguin Books, 2000

As in the previous chapter we have focused on Service Providers and Service Developers, in this chapter we continue by investigating how some of their requirements, especially *Req 1 An overall model*, can be fulfilled. Enterprise Architecture strives to achieve overall representation; therefore, we will present this field in more detail. An important initiative in the Enterprise Architecture field is the definition of frameworks, which aim at structuring concepts and activities necessary for designing and building a system. One telecommunications-specific framework (which is not an Enterprise Architecture framework) for organizing, specifying and developing new generation management systems is Frameworx. Recent work has established rich synergies between Frameworx and an Enterprise Architecture framework, TOGAF. Hence, TOGAF seems to be a suitable Enterprise Architecture framework for telecommunications; it is envisaged as a general framework which can be extended and complemented with telecommunications specific concepts and activities derived from Frameworx. However, a framework is just theoretical, to be applied to modeling systems, it needs Modeling Languages, called for Enterprise Architecture, Enterprise Architecture Modeling Languages. The Enterprise Architecture Modeling Language closest to TOGAF is ArchiMate. As domain (e.g. telecommunications) specific information is envisaged to be introduced in TOGAF, our proposal is to extend ArchiMate with domain (e.g. telecommunications) specific concepts. The resulted Modeling Language, defined as an Enterprise Architecture Modeling Language (ArchiMate) extension, will thus benefit from the advantages Enterprise Architecture offers.

## 2.1 Enterprise Architecture Frameworks

An *enterprise* is one or more organizations which have a definite mission, goals and objectives to offer an output such as a product or a service [ISO 00]. This includes the extended enterprise (integration of suppliers and customers) and virtual enterprise (oriented

to interoperability of dynamic networked enterprises). Enterprises are diverse, made up of multiple interconnected elements, both technical and social. The development of large and complex organizations or systems involves many people, stakeholders, each with their own viewpoint.

To document, understand and master the complexity of an enterprise Information Systems, architectures are an important means [Troche 06]. Here, we adopt the IEEE Standard 1471-2000 for the term *architecture*. This defines it as "the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principle guiding its design and evolution" [IEEE Comp. Soc. 00].

Another manner of doing this is by separation of concerns, specification of multiple viewpoints. In [ISO/IEC 07], a *viewpoint* is defined as a "work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns". A *view* is defined as a "work product expressing the architecture of a system from the perspective of specific system concerns". A *concern* is an "interest in a system relevant to one or more stakeholders". A *stakeholder* is an "individual, team, organization, or classes thereof, having an interest in a system".



To describe the architecture of an enterprise, Enterprise Architecture is useful. There are several definitions of Enterprise Architecture, e.g., [Chen 08], [Jonkers 06]. Related to the variety and difference of definitions in the Enterprise Architecture discipline, [Schoenherr 09] considers that there is a lack of theoretical foundation, definitions or a common understanding within the authors who publish in this context. To avoid contributing to this problem, we adopt definitions for the main concepts. For *Enterprise Architecture (EA)*, [Jonkers 06] defines it as "a coherent whole of principles, methods and models that are used in the design and realization of the enterprise's organizational structure, business processes, information systems, and infrastructure". *Enterprise Modeling (EM)* describes the Enterprise Architecture from various viewpoints in detail to allow the system's specification and implementation [Chen 08]. Thus, Enterprise Modeling combines two means of mastering complexity, architecture and separation of concerns.

Enterprise Architecture benefits (e.g., [Chen 08] [Jonkers 06] [Jonkers 04]) have been reviewed and categorized by [Niemi 06] into hard, intangible, indirect and strategic. They include:

- strategic:
  - increasing the insight and overview required to successfully align the business and technology platforms;
  - improved change management;
  - providing enterprise stability and agility;
  - promoting common understanding across the enterprise;
- indirect:
  - understanding the wealth of relations between an enterprise and its customers and other partners;
  - improved risk management;
  - managing system complexity;
- intangible:
  - providing a holistic view of the enterprise;
  - helping decision making;
- hard:

- improved interoperability and integration of enterprise constituting systems;
- reduced costs;
- shortened cycle time;

The evidence of the contribution of Enterprise Architecture to the achievement of organizational goals is reviewed by [Boucharas 10]. To achieve these goals, alignment between viewpoints and enterprise strategy is needed [Simonin 11].

According to IFAC-IFIP Task Force [IFAC IFIP Task Force 99] and ISO 15704 [ISO 00] there are two types of enterprise architectures: *System architectures* (also called "Type 1") that deal with the design of a system, *Enterprise-reference projects* (also called "Type 2") that deal with the organization of the development and implementation of a project such as an enterprise integration or other enterprise development program. Type 1 architectures represent a system (e.g. a system of the Information System of an enterprise) in terms of its structure and behavior. Type 2 architectures are actually frameworks that aim at structuring concepts and activities/tasks necessary to design and build an enterprise system. The system design (Type 1) must be coherent with that of other enterprise systems and especially be aligned with the enterprise strategy (Type 2). According to a survey of Enterprise Architectures [Chen 08], most of these are Type 2 architectures, frameworks [1].

An *Enterprise Architecture framework* is a structure expressed in terms of diagrams, text and formal rules that relates the components of a conceptual entity to each other [ISO 06]. It defines and organizes the generic concepts that are required to enable the creation of enterprise models for industrial businesses. Its main purpose is to provide an organizing mechanism so that concepts, problems and knowledge on enterprise interoperability (both inter and intra) [Guedria 08] can be represented in a more structured way. Enterprise Architecture frameworks have been reviewed these last years for example by [Mahmood 06]. They have been compared for example by [Leist 06] [Urbaczewski 06]. Notable examples include: the Computer Integrated Manufacturing Open System Architecture (CIMOSA), the Purdue Enterprise-Reference Architecture (PERA), the Zachman Framework, the Generalized Enterprise-Reference Architecture and Methodology (GERAM), The Open Group Architecture Framework (TOGAF), the Command Control Communication and Computer Intelligence Surveillance and Reconnaissance (C4ISR) (also known as Department of Defense (DoD) Architecture Framework - DoDAF), the Integrated Architecture Framework (IAF).

Applying Enterprise Architecture frameworks to create Type 1 architectures (i.e. unambiguously specifying and describing system components and their relations) requires coherent Enterprise Architecture Modeling Languages [Jonkers 04]. A *Modeling Language (ML)* is "a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system" [Booch 05]. An *Enterprise Architecture Modeling Language (EAML)* is a language with a high level of abstraction, which aims at representing enterprise architectural structure, characteristics and properties at an early stage of design [Chen 08]. Although general purpose modeling languages like UML have been applied to model Enterprise Architectures, e.g. [Fatolahi 06], they are not particularly adapted to this task. The most notable example of an Enterprise Architecture Modeling Language is ArchiMate [The Open Group 09a].

1. http://en.wikipedia.org/wiki/Enterprise_Architecture_Framework#Types_of_enterprise_architecture_framework, , accessed on 06.11.2011

## 2.2 Frameworks for Telecommunications

Frameworx [TM Forum 10] is a telecommunications-specific framework developed by the TM Forum to organize, specify, design and develop new generation management systems. It provides a standard method, common terminology and harmonized framework for the telecommunications industry value chain. It consists of four frameworks or content models which represent the cornerstones of a Service Provider Enterprise Architecture:

1. *Business Process Framework enhanced Telecom Operations Map (eTOM)*: It defines most of the common business processes of a Service Provider environment based on the current state of technology. It provides a standard framework and a common language for the definition and classification of most business processes within a Service Provider's environment.
2. *Enterprise-wide Information Framework Shared Information and Data (SID) Model*: It provides a comprehensive common information and data model covering a wide set (though not all) of business concepts relevant within a Service Provider's environment. It also offers a common language that software developers and integrators can use to describe information and data.
3. *Application Framework Telecom Application Map (TAM)*: a common reference map and language to navigate the complex application landscape.
4. *Integration Framework Technology-Neutral Architecture (TNA)*: It identifies the dependencies and unifies the TM Forum eTOM, SID, TAM and TM Forum Interface Program (TIP) in a Service-Oriented Architecture (SOA) context.The key to the Integration Framework is a growing set of re-usable building blocks, known as "business services" (also known previously as NGOSS contracts).

These documents are currently under development; hence changes could appear in the future.

Currently, an industry liaison group between TOGAF and TM Forum is exploring synergies between TOGAF and Frameworx. They have publicized a white paper on the the mapping of TOGAF and Frameworx solutions eTOM, SID and TAM [TOGAF and TM Forum 11] (cf. also Section 2.4). Due to this connection and mapping between the two, we will investigate TOGAF in more detail in the next section.

## 2.3 TOGAF

Among Enterprise Architecture frameworks, the most promising to date for telecommunications is The Open Group Architecture Framework (TOGAF) [The Open Group 09b]. A mapping [TOGAF and TM Forum 11] between TOGAF and Frameworx has identified important synergies between them.

TOGAF provides methods of assisting in the acceptance, production, use and maintenance of an Enterprise Architecture. At the core of TOGAF is the Architecture Development Method (ADM). ADM provides an iterative process of continuous architecture development. It provides one of the most complete [Sessions 07] guiding step-by-step process for creating an Enterprise Architecture. ADM (cf. left part of Figure 2.1) consists of eight phases:

– The *Preliminary Phase* describes the preparation and initiation activities required in order to meet the business directive for a new enterprise architecture.

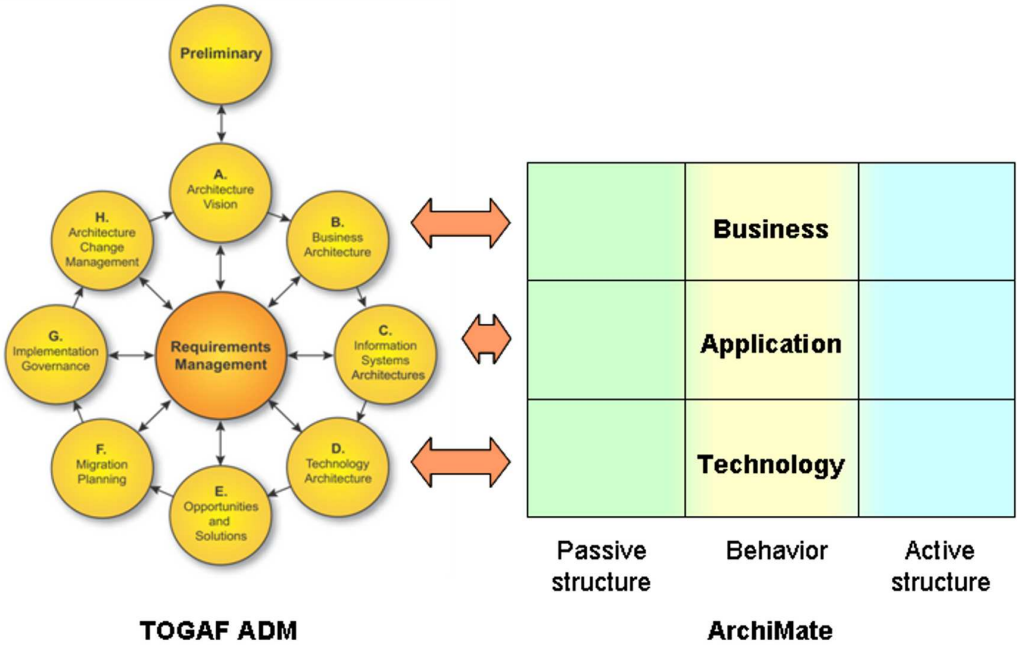


Figure 2.1 : Correspondence between TOGAF and ArchiMate, from [The Open Group 09a].

- *Phase A: Architecture Vision* includes information regarding scope definition, stakeholder identification, Architecture Vision creation and approval obtainment.
- *Phase B: Business Architecture* has as objective the development of a Target Business Architecture, describing the product/service strategy, and the organizational, functional, process, information, and geographic aspects of the business environment, based on the business principles, business goals, and strategic drivers.
- *Phase C: Information Systems Architectures* focuses on identifying and defining the applications and data considerations that support an enterprise's Business Architecture. It actually has two sub-phases:
  - *Data Architecture* has as objective the definition of the major types and sources of data necessary to support the business. This has to be done in a way that is understandable by stakeholders, complete, consistent and stable.
  - *Application Architecture* has as objective the definition of the major kinds of application systems necessary to process the data and support the business.
- *Phase D: Technology Architecture* seeks to map application components defined in the Application Architecture phase into a set of technology components. These represent software and hardware components, available from the market or configured within the organization into technology platforms.
- *Phase E: Opportunities and Solutions* conducts initial implementation planning and the identification of delivery vehicles for the architecture defined in the previous phases.
- *Phase F: Migration Planning* addresses the formulation of a set of detailed sequences

of transition architectures with a supporting Implementation and Migration Plan.
- *Phase G: Implementation Governance* provides an architectural oversight of the implementation.
- *Phase H: Architecture Change Management* establishes procedures for managing change brought to the new architecture.
- *Requirements Management* examines the process of managing architecture requirements throughout the ADM. It is central to the ADM and shows how requirements should be traced during the entire process.

TOGAF ADM is iterative, over the whole process, between phases, and within phases; [The Open Group 09b] suggests iteration cycles. This enables the evolution of Enterprise Architectures modeled with TOGAF ADM.

A comparison between TOGAF and several architecture initiatives (e.g., RM-ODP, Zachman Framework) is provided by [The Open Group 07]. TOGAF is considered [Urbaczewski 06] to be one of the best frameworks concerning business and technical layers, as it provides much detail for these.

## 2.4 Mapping of TOGAF and Frameworx

We focus here only on the mapping of TOGAF ADM and Frameworx, cf. Fig. 2.2, from [TOGAF and TM Forum 11]. The three Frameworx solutions map mainly into Phases A, B and C of the TOGAF ADM:

- The ADM *Preliminary Phase* does not explicitly exist in Frameworx and is consequently to be considered as an add-on to Frameworx when an enterprise architecture approach is to be applied.
- The ADM *Phase A: Architecture Vision* also represents a new part for Frameworx, as this is only covered partially by the latter in the form of definitions related to the formulation of the enterprise' strategy.
- The *Business Process Framework enhanced Telecom Operations Map (eTOM)* describes the foundational business processes of a telecommunications Service Provider and therefore maps to *Phase B: Business Architecture* of the TOGAF ADM.
- The *Enterprise-wide Information Framework Shared Information and Data (SID) Model* provides a common comprehensive information and data model for a telecommunications Service Provider and it maps to sub-phase *Data Architecture* of *Phase C: Information Systems Architectures* of the TOGAF ADM.
- Service Providers normally talk about BSS/OSS Systems (Business/Operations Support Systems); therefore, the *Application Framework Telecom Application Map (TAM)* maps to sub-phase *Application Architecture* of *Phase C: Information Systems Architectures* of the TOGAF ADM. "The mapping of the Frameworx TAM with the TOGAF Information Systems (Application Architecture) appears to be a rich area in terms of synergies between the two models" [TOGAF and TM Forum 11].

In conclusion, among current Enterprise Architecture frameworks, TOGAF seems to be the most likely to be applied to Telecommunications.

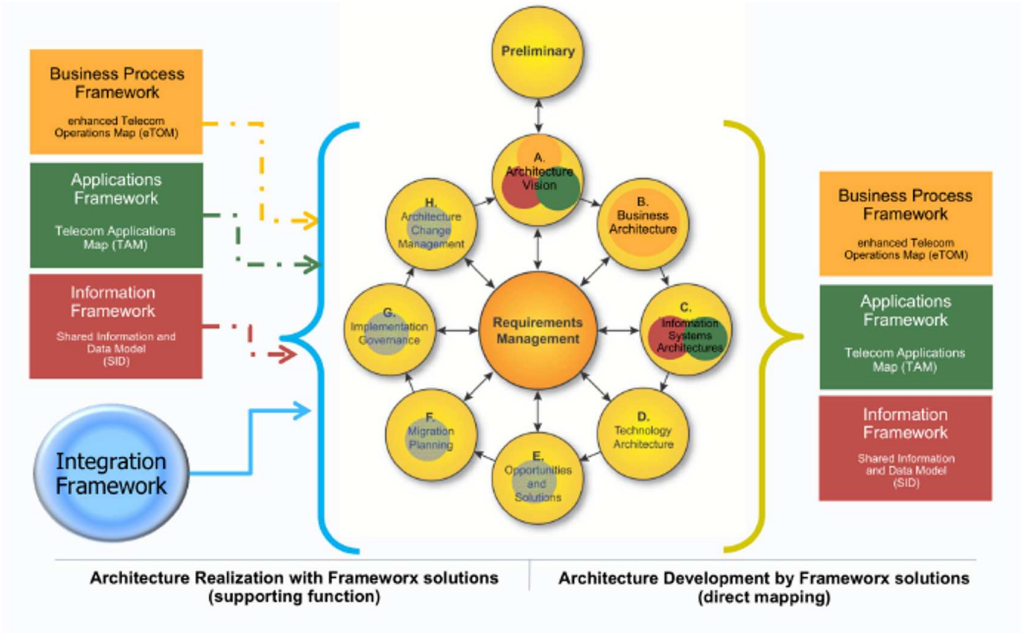


Figure 2.2 : Mapping between TOGAF ADM and Frameworx, from [TOGAF and TM Forum 11].

## 2.5 ArchiMate

One Enterprise Architecture Modeling Language is ArchiMate [The Open Group 09a]. It shares [Berrisford 09] important concepts with TOGAF, and thus "the two are usable together". It models three phases of TOGAF ADM (cf. Figure 2.1) into three layers: Business, Application and Technology. It regulates interoperability between them by defining possible dependencies.

### 2.5.1 The Abstract Syntax

Each of the three ArchiMate layers specifies a Meta-Model (cf. the Meta-modeling approach for language definition, Section 3.3, for the relation between Modeling Languages and Meta-Models). The ArchiMate Business layer Meta-Model [The Open Group 09a], presented in Figure 2.3, captures the relations between structural concepts:

- Business Actor: an organizational entity capable of (actively) performing behavior.
- Business Role: a named specific behavior of a business actor participating in a particular context.
- Business Collaboration: a (temporary) configuration of two or more business roles resulting in specific collective behavior in a particular context.
- Business Interface: a declaration of how a business role can connect with its environment.

between behavioral concepts:

- Business Process: a unit of internal behavior or collection of causally-related units of internal behavior intended to produce a defined set of products and services.

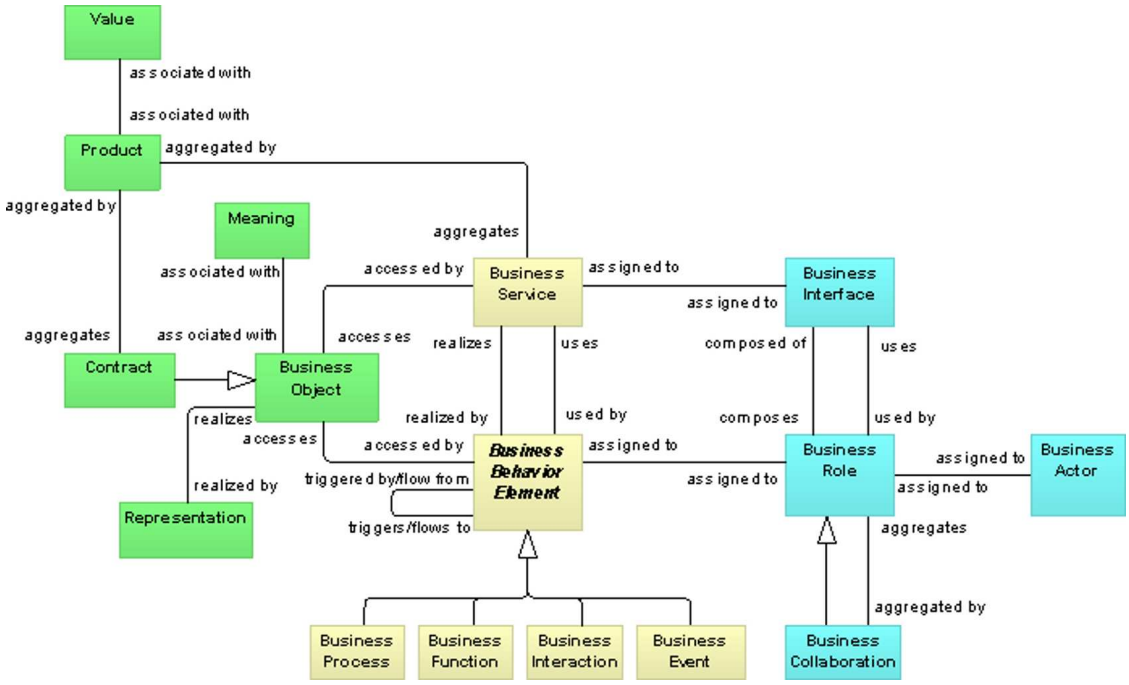


Figure 2.3 : The ArchiMate Business layer Meta-Model, from [The Open Group 09a].

- Business Function: a unit of internal behavior that groups behavior according for example, to required skills, knowledge, resources, etc., and is performed by a single role within the organization.
- Business Interaction: a unit of behavior performed as a collaboration between two or more business roles.
- Business Event: an occurrence (internal or external) that influences behavior (business process, business function, business interaction).
- Business Service: the externally visible ("logical") functionality, which is meaningful to the environment and is realized by business behavior (business process, business function, or business interaction).

and between informational concepts:

- Business Object: a unit of information that has relevance from a business perspective.
- Representation: the perceptible form of the information carried by a business object.
- Meaning: the knowledge or expertise that is to be found in the representation of a business object, given a particular context.
- Value: that which makes a party appreciate a service or a product, possibly in relation to providing it, but more typically to acquiring it.
- Product: a coherent collection of services, accompanied by a contract/set of agreements, which is offered as a whole to (internal or external) customers.
- Contract: a formal or informal specification of an agreement that specifies the rights and obligations associated with a product.

The ArchiMate Application layer Meta-Model [The Open Group 09a], presented in Figure 2.4, captures the relations between structural concepts:

- Application Component: a modular, deployable, and replaceable part of a system

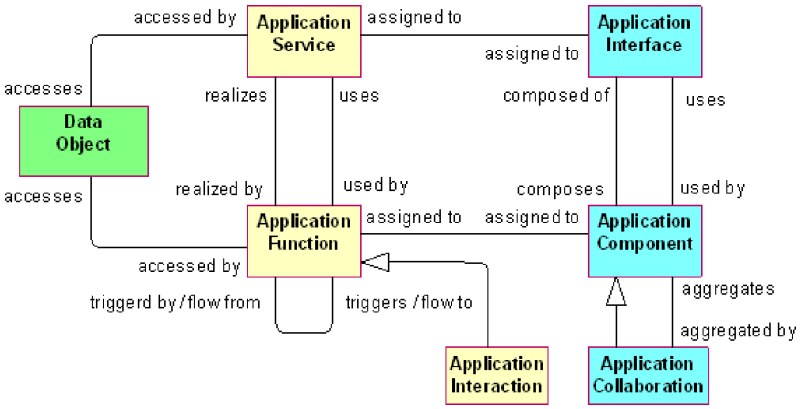


Figure 2.4 : The ArchiMate Application layer Meta-Model, from [The Open Group 09a].

that encapsulates its contents and exposes its functionality through a set of interfaces.

- Application Collaboration: a (temporary) configuration of two or more components that co-operate to jointly perform application interactions.
- Application Interface: a declaration of how a component can connect with its environment.

between behavioral concepts:

- Application Function: a representation of a coherent group of internal behavior of an application component.
- Application Interaction: a unit of behavior performed by collaboration between two or more components.
- Application Service: an externally visible unit of functionality, provided by one or more components, exposed through well-defined interfaces, and meaningful to the environment.

and between informational concepts:

- Data Object: a coherent, self-contained piece of information suitable for automated processing.

The ArchiMate Technology layer Meta-Model [The Open Group 09a], presented in Figure 2.5, captures the relations between structural concepts:

- Node: a computational resource upon which artifacts may be deployed for execution.
- Device: a physical computational resource upon which artifacts may be deployed for execution.
- Infrastructure Interface: a point of access where the functionality offered by a node can be accessed by other nodes and application components.
- Network: a physical communication medium between two or more devices.
- Communication path: a link between two or more nodes, through which these nodes can exchange information.

between behavioral concepts:

- Infrastructure Service: an externally visible unit of functionality, provided by one or more nodes, exposed through well-defined interfaces, and meaningful to the environment.

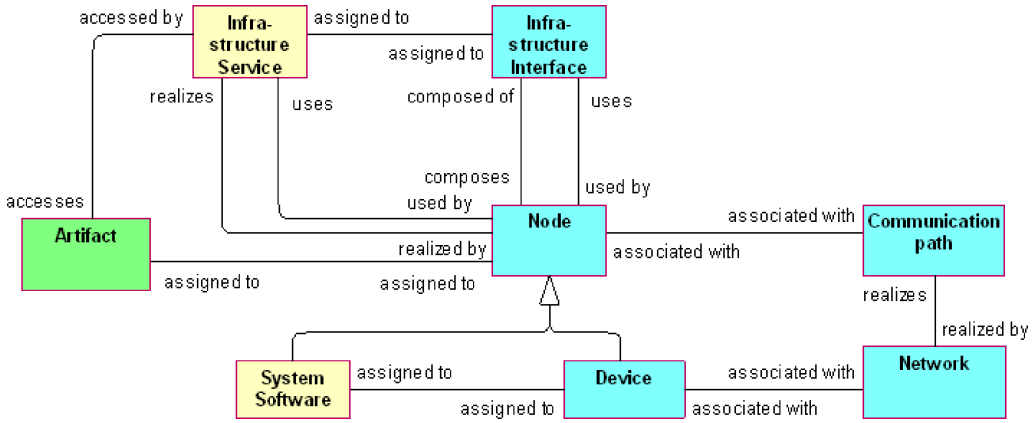


Figure 2.5 : The ArchiMate Technology layer Meta-Model, from [The Open Group 09a].

– System Software: a software environment for specific types of components and objects that are deployed on it in the form of artifacts.

and between informational concepts:

– Artifact: a physical piece of information that is used or produced in a software development process, or by the deployment and operation of a system.

The relations that are used in the ArchiMate Meta-Models are:

– Structural:
  – Composition: indicates that an object consists of a number of other objects.
  – Aggregation: indicates that a concept groups a number of other concepts.
  – Assignment: links active elements (e.g., business roles or application components) with units of behavior that are performed by them, or business actors with business roles that are fulfilled by them.
  – Realization: links a logical entity with a more concrete entity that realizes it.
  – Used by: models the use of services by processes, functions, or interactions and the access to interfaces by roles, components, or collaborations.
  – Access: models the access of behavioral concepts to business or data objects.
  – Association: models a relationship between objects that is not covered by another, more specific relationship.
– Dynamic:
  – Triggering: describes the temporal or causal relations between processes, functions, interactions, and events.
  – Flow: describes the exchange or transfer of, for example, information or value between processes, function, interactions, and events.
– Other:
  – Grouping: indicates that objects belong together based on some common characteristic.
  – Junction: is used to connect dynamic relationships of the same type.
  – Specialization: indicates that an object is a specialization of another object.

Of course, each of these concepts has (at least one) graphical representation.


Figure 2.6 : The Business ArchiMate concrete syntax, from [The Open Group 09a].


Figure 2.7 : The Application ArchiMate concrete syntax, from [The Open Group 09a].

### 2.5.2 The Concrete Syntax

The graphical representations of the ArchiMate concepts (cf. the abstract syntax, Figure 2.3, Figure 2.4, Figure 2.5) and relations are presented respectively in Figure 2.6, Figure 2.7, Figure 2.8, Figure 2.9. For example, the *Business Actor* (cf. Figure 2.3) is represented as a human sketch (cf. first image of Figure 2.6. Relations have a graphical representation as well (cf., e.g. Composition, the first image of Figure 2.9).

### 2.5.3 The Semantics

A description in natural language (English) of the meaning of ArchiMate concepts and relations is given in Section 2.5.1. This can be considered as a (informal) means of describing language semantics.

### 2.5.4 Interoperability between ArchiMate Layers

A central issue in Enterprise Architecture is business-IT alignment: how to match the layers? This is part of the more general issue of view alignment. ArchiMate defines the possible relations between concepts of two adjacent layers.

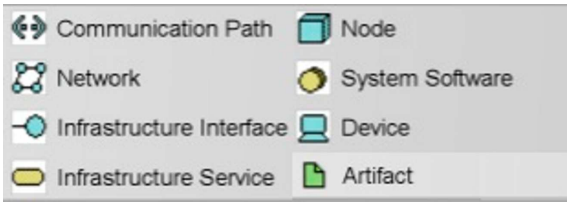


Figure 2.8 : The Technology ArchiMate concrete syntax, from [The Open Group 09a].

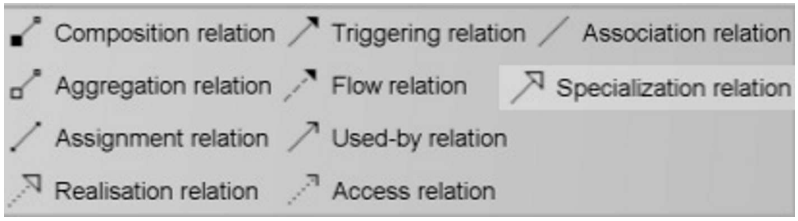


Figure 2.9 : The ArchiMate relations concrete syntax, from [The Open Group 09a].

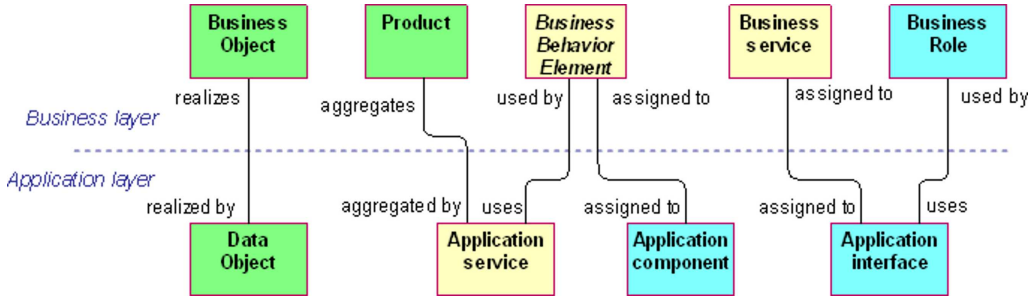


Figure 2.10 : The ArchiMate Business-Application alignment, from [The Open Group 09a].

Figure 2.10 shows the relationships between business layer and application layer concepts. There are three main types of relationships between these layers:

- *Used by* relationships, between application service and the different types of business behavior elements, and between application interface and business role. These relationships represent the behavioral and structural aspects of the support of the business by applications.
- A *realization* relationship from a data object to a business object, to indicate that the data object is a digital representation of the corresponding business object.
- *Assignment* relationships, between the application component and the different types of business behavior elements, and between the application interface and business service, to indicate for example, that business processes or business services are completely automated.

Figure 2.11 shows the relationships between application layer and technology layer concepts. There are two types of relationships between these layers:

- *Used by* relationships, between infrastructure service and the different types of application behavior elements, and between infrastructure interface and application component. These relationships represent the behavioral and structural aspects of the use of technical infrastructure by applications.
- A *realization* relationship from artifact to data object, to indicate that the data object is realized for example, by a physical data file, and from artifact to application component, to indicate that a physical data file is executable, realizing an application or part of an application.

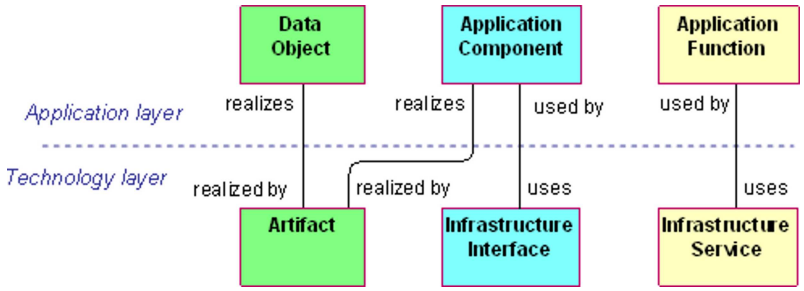


Figure 2.11 : The ArchiMate Application-Technology alignment, from [The Open Group 09a].

## 2.6 Introducing Domain Specificity in Enterprise Architectures

An Enterprise Architecture Modeling Language offers the advantage of a unified language, capable of describing a wide range of domains. It allows the creation of integrated models of the enterprise, which can be understood by all stakeholders. While this is very useful at the business layer, at the technical layer, where more detail is needed to describe a system, this unified language lacks the required semantic strength. This means that the concepts present in the language are too abstract and they need refinement and specification. To cover this lack, an Enterprise Architecture Modeling Language development method [Khoury 07] proposes the use of a unified modeling language at the business level, while at more detailed levels, it suggests the use of Domain Specific Modeling Languages and domain specific methods.

ArchiMate was actually designed as a Modeling Language at an abstraction level between very/too generic concepts, and domain-specific concepts (cf. Figure 2.12). Its concepts provide a common basis and an integration layer for the more specific concepts in the bottom layer. ArchiMate connects architectural domains, while acknowledging the need for specialized languages for different architectural domains (e.g. UML, BPMN [Steen 04], [Lankhorst 09]). Such Modeling Languages can be thought of as e.g. describing the structure of houses and office buildings, whereas ArchiMate describes the structure of cities. ArchiMate models the architecture of an enterprise (all of the business, information and technology systems in an organization), whereas e.g. UML models the structure and behavior of a software-based (i.e. part of the information) system.

Although the ArchiMate notation intentionally resembles that of UML [The Open Group 09a], their semantics are different. Moreover, many ArchiMate concepts are derived from or related to UML (especially for application and infrastructure), and BPMN (at the business layer) [Lankhorst 05]. A partial mapping (of common concepts) between ArchiMate and UML, and between ArchiMate and BPMN, is offered in [Steen 04].

The domain-specific concepts can be ”further specialized or composed to form concepts tailored towards a more specific context” [Steen 04]. The TM Forum has produced a technical report [TM Forum 09] which recommends that the tooling framework should be oriented towards the use of Domain Specific Languages in order to better address the requirement specific to each individual modeling activity and simplify the modeling

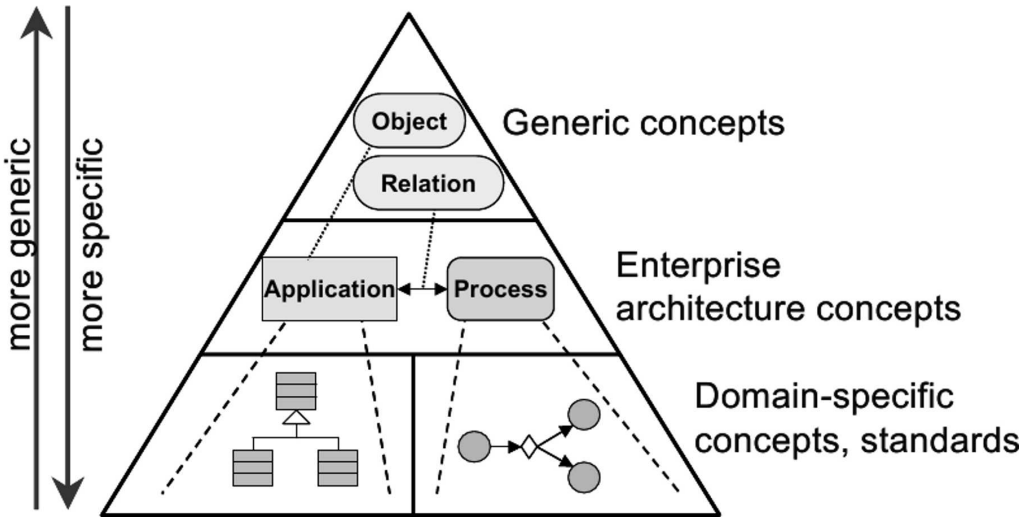


Figure 2.12 : Modeling Languages at different levels of specificity, from [Steen 04].



process for end-users.

In accordance with the Enterprise Architecture Modeling Language development method proposed by [Khoury 07], and following the TM Forum recommendations [TM Forum 09], we propose the definition of a telecommunications Domain Specific Modeling Language which extends ArchiMate.

## 2.7 Enterprise Architectures for Telecommunications

Applying Enterprise Architecture for telecommunications service development includes, for example, the Enterprise Architecture for Unified Process (EA4UP) [Simonin 08] telecommunications service development method. [Simonin 08] also reports on the results of applying EA4UP on eleven industrial projects and on obtaining notable advantages for component re-usability, but increased costs as well. [Moghaddam 08] reports on using the Enterprise Architecture framework GERAM together with axiomatic design at the Iran Telecommunication Research Center to develop a reference architecture; they indicate as a major benefit the tracking and change management of entities. [Bertin 09b] argues that business processes are essential in achieving service convergence. [Zoric 11], although does not apply Enterprise Architecture, it does introduce a scenario-driven approach for techno-business modeling and analysis.

Automatic (formal) measures are useful to the alignment of the functional and business views of the telecommunications services. A proposal for such measures, using Model Driven Engineering, is presented in [Simonin 07], [Simonin 10].

Specific advantages provided by Enterprise Architecture in telecommunications include, for example [Arnold 10]:

– Managing complexity with alternative technology choices: telecommunications innovations have to be prepared to accommodate not only rapid and uncertain developments in the application domain, but also the technological uncertainty from the control and access layers throughout the user interface.

– Reducing implementation uncertainty due to the distance between the strategic and the operational level.
– Synchronizing innovation processes.

TM Forum has recently published a survey [Manuel (ed.) 11] on the adoption and benefits of the telecommunications-specific framework Frameworx by Service Providers. According to the report, the survey targeted 86 of the world's top 100 Service Providers. From the respondents, 83% said they already use eTOM, SID and TAM. 80% of respondents cited the simplification of IT architecture as the biggest reason for using Frameworx, with 60% reporting benefits in this area. The second biggest adoption driver, according to 59% of respondents, is Frameworx as an important enabler for new service roll-out. For example, MIMO Tech Co. Ltd. of Thailand reduced operating expenditure by 20 to 30% and continues to see at least a 50% improvement in efficiencies in the way it offers new value-added services. According to 72% of respondents, Frameworx is being used more widely to reduce the cost and risks involved in integration. Frameworx also enables massive savings. For example, China Unicom estimates that the single tier, centralized platform has saved more than $7.86 million on system implementation and $ 3.93 million on maintenance costs annually. Another example is offered by the U.S. multi-service operator Charter Communications, which reduced its annual IT spend by nearly 60% by introducing greater IT governance, using Frameworx as a common reference across IT operations and planning.



## 2.8 Discussion

We have seen examples of Enterprise Architecture applied to telecommunications service development. The main reasons for using it are related to managing complexity and increasing component re-usability. More specifically, there is a current effort to map the Telecommunications framework, Frameworx, to an Enterprise Architecture framework, TOGAF. Taking into consideration the requirements of Service Providers and Service Developers for Service Creation Environment, Enterprise Architecture is expected to contribute towards:

– *Req 1 An overall model*: Enterprise Architecture presents a unified view of the enterprise, from different viewpoints, at different layers, so that one can obtain an integrated, overall model of service creation, which takes in business, management, and technical activities.
– *Req 6 Integration*: The specific Enterprise Architecture Modeling Language of ArchiMate defines the possible relations between its layers and therefore contributes towards the integration/interoperability of models produced at different layers. However, interoperability between viewpoints remains an issue.
– *Req 7 Reuse*: By introducing a layered architecture of the enterprise, components at a lower layer of abstraction may be re-used by several components at a higher layer of abstraction.
– *Req 8 Wide range of services*: The layered architecture enables the separation of business and functional telecommunications service descriptions from the platform/network.
– *Req 9 Easy evolution of services*: The specific Enterprise Architecture framework TOGAF addresses the evolution of Enterprise Architectures.

In conclusion, Enterprise Architecture contributes towards meeting a number of requirements for Service Creation Environment. To benefit from this, we propose using Enterprise Architecture frameworks and Enterprise Architecture Modeling Languages. Due to its closeness to Frameworx, we choose TOGAF and ArchiMate. To introduce telecommunications specificity in the Enterprise Architecture framework, we propose extending the Enterprise Architecture Modeling Language. However, this does not address all of the requirements. To address especially *Req 2 Domain specificity*, *Req 3 Rapid prototyping* and *Req 5 Early verification/simulation*, we investigate in the next chapter the use of Model Driven Engineering for telecommunications.

In this chapter we presented Enterprise Architectures, which provides a set of models to describe the structure and functions of an enterprise. It thus contributes to meeting Service Providers' requirements of a unified global view concerning the construction process, requirements of the integration, the reuse, the wide range and the easy evolution of telecommunications services. Due to its synergy with the telecommunications specific framework Frameworx, we chose to focus on the Enterprise Architecture framework TOGAF and the Enterprise Architecture Modeling Language usable with it, ArchiMate. To provide the semantic strength required at lower abstraction layers, an Enterprise Architecture Modeling Language should include concepts specific to the domain. Adding domain specific concepts to a language can be done through the mechanism of profiling, extending a base language. This mechanism, together with a model-driven approach for defining modeling languages, are presented in the next chapter. The model-driven approach also ensures the generation of dedicated software tools for these languages, reducing even more the effort, time and final cost of constructing telecommunications services.

# 3 Model Driven Engineering for Telecommunications Services

” ’Set to, 0 Enkidu! Two men together will not die: that lashed to a boat cannot sink, 106
no man can cut a three-ply rope,
a flood cannot sweep a man off a wall,
fire in a reed hut cannot be extinguished!
You join with me, I will join with you, what can anyone do to us then?’ ”

Bilgames and Huwawa: ’The lord to the Living One’s Mountain’, in
*The Epic of Gilgamesh. A new translation*, Andrew George, Penguin Books, 2000

In the previous chapter, we have seen how Enterprise Architecture provides a unified representation of an enterprise. We have chosen an Enterprise Architecture framework, TOGAF, and a modeling language to apply it, ArchiMate. However, ArchiMate is a language for modeling any Enterprise Architecture, whereas, to increase performance, Service Providers would benefit from the inclusion of telecommunications concepts directly in the language. One manner of defining Modeling Languages is advanced by Model Driven Engineering. This approach is intended for dedicated, graphical languages, which offer powerful expressiveness. Putting models at the center of the approach, Model Driven Engineering offers powerful generative advantages for language-associated tools (e.g. graphical editors). Models also introduce a high degree of formality, enabling a wide range of operations, collectively known as Model Transformations. We present major initiatives and tools for Model Driven Engineering. We discuss advantages and limitations of using Model Driven Engineering for telecommunications service creation.

## 3.1 Model Driven Engineering Challenges

*Model Driven Engineering (MDE)* is a software development method which focuses on creating and exploiting domain models [1]. It allows the exploitation of models to simulate, estimate, understand, communicate and produce code [Gherbi 09]. *Models* are representations of the reality for a given purpose; they are abstractions of reality in the sense that they cannot represent all aspects of reality, allowing people to deal with the world in a simplified manner, avoiding the complexity, danger and irreversibility of reality [Rothenberg 89]. In Model Driven Engineering, models are conformant to *Meta-Models (MM)*, much like a program is conformant to the grammar of the programming language (cf. $M_1$ and $M_2$

1. `http://en.wikipedia.org/wiki/Model-driven_engineering`, , accessed on 06.11.2011

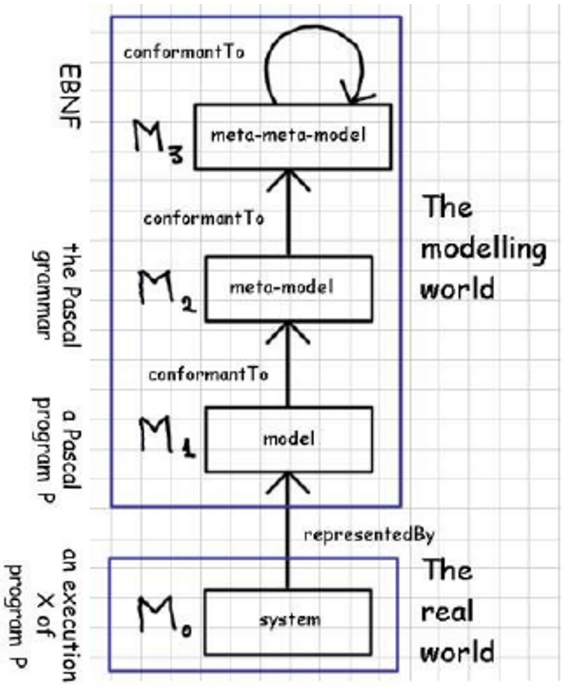


Figure 3.1 : Relation between Meta-Modeling and Language Theory, after [Bezivin 04].

levels from Figure 3.1, after [Bezivin 04]). Meta-Models are models as well, but they are more abstract and general than regular models. To describe Meta-Models, a third abstraction level is introduced, the Meta-Meta-Models. This corresponds to formalisms/languages used to describe grammars of programming language.

The main Model Driven Engineering challenges are [France 07]:

- Model manipulation and management challenges: associated with
  - defining, analyzing, and using Model Transformations;
  - maintaining traceability links among model elements in order to support model evolution and round-trip engineering;
  - maintaining consistency among viewpoints;
  - tracking versions;
  - using models during runtime.
- Modeling Language challenges: related to providing support for creating and using problem-level abstractions in modeling languages, and rigorously analyzing models.
- Separation of concerns challenges: associated with modeling systems using multiple, overlapping viewpoints that utilize possibly heterogeneous languages.

In the next sections, among these challenges, we will focus on Model Transformations, Modeling Languages and separation of concerns.

## 3.2 Model Transformations

A Model Transformation (MT) [Kleppe 03] is the automatic generation of target model(s) from source model(s), according to a set of transformation rules. A transforma-

tion rule is a description of how one or more constructs in the source language can be transformed into one or more constructs in the target language. Model Transformations have been classified according to multiple dimensions [Czarnecki 06], [Mens 06], [France 07]. Three of these dimensions are:

1. Number of source and target models. Usually the number of sources is indicated first, that of targets, second. There are: *one-to-one* (one source model is transformed into one target model), *one-to-many*, *many-to-one*, and *many-to-many*.
2. Level of abstraction. It is related to the amount of details included in the models (e.g. a UML model vs. a Java model/program). If the source model(s) are on the same level of abstraction as the target model(s), the Model Transformation is called *horizontal*, if they are on different levels, the Model Transformation is called *vertical*.
3. Technical space. It refers to the Meta-Model/language to which models are conformant. If the source and target technical spaces are the same, the Model Transformation is called *endogenous*, if they are different, the Model Transformation is called *exogenous*.

The most common types of Model Transformations are:

1. Refinement. Also called model-to-text, synthesis or code generation. It is a one-to-one, vertical, exogenous Model Transformation. Usually used to implement the operational semantics of languages, or render models executable (e.g. from UML to Java).
2. Abstraction. Also called text-to-model or reverse engineering. It is an operation complementary to Refinement, and like it, is a one-to-one, vertical, exogenous Model Transformation.
3. Translation. Also called migration. It is a one-to-one, horizontal, exogenous Model Transformation (e.g., from Java to C++, from UML to Petri Nets).
4. In-place. It is a Model Transformation for which the source model is the same with the target model. Examples comprise optimization, refactoring, extension. It is a one-to-one, horizontal, endogenous Model Transformation.
5. Decomposition. One source model is transformed into several target models. It is a one-to-many, horizontal, endogenous Model Transformation.
6. Synchronization. Changes from one source models are transmitted to several target models. It is a one-to-many, horizontal, endogenous Model Transformation.
7. Composition. Also called merging. Several source models are brought together into one target model. It is a many-to-one, horizontal, endogenous Model Transformation.

Model Transformations are models themselves, and they are written in Model Transformation languages and they usually have Meta-Models as inputs and outputs. Because they are models, they can serve as inputs and/or outputs to other Model Transformations. Model Transformations which have other Model Transformations as input and/or as output are called *Higher Order model Transformations (HOTs)* [Tisi 09].

## 3.3 The Meta-modeling approach for language definition

There are several methods for defining a (programming/modeling) language. Compiler theory [Aho 72] uses: abstract syntax, concrete syntax, and semantics. It provides (meta-)tools (e.g., lex, yacc) that generate language-specific tools (e.g., lexical, syntacti-

cal parsers). However, it deals with textual languages. A *Modeling Language (ML)* is "a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system" [Booch 05].

A denotational, frequently used approach for defining graphical Modeling Languages is the Meta-modeling approach for language definition [Clark 01], [Kurtev 06]. It defines a Modeling Language as a set of five components (cf. Figure 3.2, without the big red rectangle):

- Concrete syntax : a human-centric representation of the syntax domain. The syntax domain defines the symbols used to represent the concepts in the language. A Modeling Language may have different concrete syntaxes. Each one is defined by a "display surface" Meta-Model.
- Abstract syntax : a computer-centric representation of the syntax domain. It expresses the notions specific to the domain of the language. In Model Driven Engineering, the domain is modeled through the use of a Meta-Model. Therefore, the Meta-Model describing the domain represents the abstract syntax of the Modeling Language. Meta-Models play the same role for Modeling Languages as grammars do for programming languages [Kleppe 07].
- The semantic domain: the meaning of the concepts in the language. Formal semantic description is significant for the design, reasoning and standardization of programming languages, ensuring their final unambiguous execution. The semantic domain is the most difficult to define. It may be defined through the semantic mapping towards the precise semantics of an existing programming language [Kurtev 06]. Dynamic semantics (e.g. control structures) may be described through operational, denotational or axiomatic frameworks [Zhang 04]. Static semantics (e.g. data types) may be described through ontologies [Ciocoiu 00].
- The display mapping: links the abstract syntax to the concrete syntax. It can be defined as a Model Transformation [Kurtev 06].
- The semantic mapping: links the abstract syntax to the semantic domain. It can be defined as a Model Transformation [Kurtev 06]; the most promising candidate is the Refinement (code generation) Model Transformation type.

An important re-use mechanism of language definition and associated tools is language extension. Extension mechanisms allow refining the reference language in a strictly additive manner, so that they cannot contradict standard semantics. This means that profile tools, defined as extensions to existing tools for the reference language, only have to process the additions. A profile [Alhir 02] is a generic extension mechanism for customizing reference languages with constructs that are specific to particular domains, platforms. Although profiles are usually associated with UML, they can be generalized to other languages as well. The term profile is used in this thesis in its general meaning, and not in its UML-bound sense. The initial cost of Domain Specific Modeling Language tool implementation as profile is lower than that for a standalone Domain Specific Modeling Language. Also, interoperability between several Domain Specific Modeling Languages, defined as profiles, is facilitated by the reference language. The constructs common between the reference language and each Domain Specific Modeling Language respectively, and the connections between the constructs from the reference language, facilitate defining connections (interoperability) between profiles. The Meta-modeling approach for language definition is rendered in Fig. 3.2, with the big red rectangle. It consists in extending the initial meta-

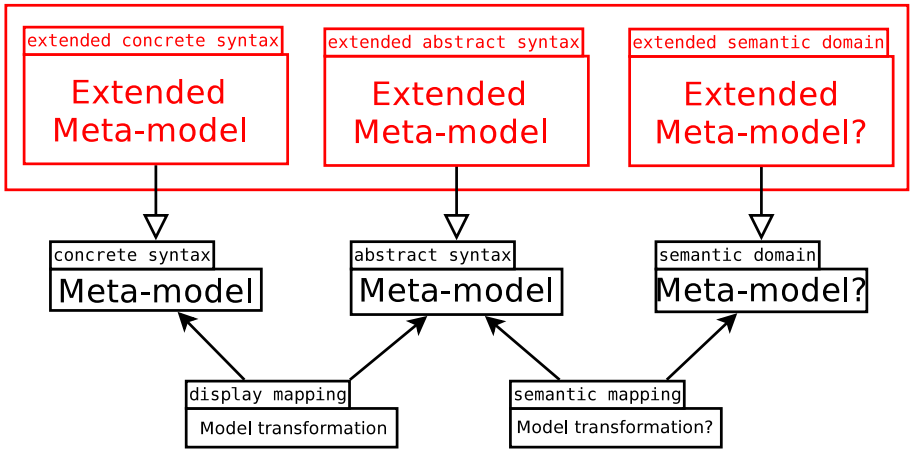


Figure 3.2 : The Meta-modeling approach for language definition, after [Clark 01]. In red, the Meta-modeling language extension approach.

models describing the abstract and concrete syntaxes, and extending the semantics.

The main advantage of the Meta-modeling approach for language definition is related to the use of Meta Tools. The general accepted definition of a Meta Tool is that it is a tool which allows the specification and generation of another tool. Meta Tools reduce significantly the time, effort and cost of constructing language-specific tools (e.g., textual or graphical syntax editors, pretty-printers, consistency checkers, analysis tools, interpreters or compilers, debuggers). When language definition evolves, it impacts the Meta-Models describing the syntax and the semantics. Due to Meta Tools, language-specific tools can be re-generated to include the new language features in a fast and inexpensive way and with limited modifications.

## 3.4 Separation of Concerns

One way to address this issue is through the *Aspect Oriented Modeling (AOM)* approach [France 04]. It provides support for the modeling of concern solutions in isolation, as modeling views called aspects (e.g. security [Woodside 09]). These aspect models are synthesized in an integrated model by composing aspect and primary model views. The main issue brought up by this approach is the model composition of the aspect and primary models. One way to describe model composition rules/directives, is through the use of Model Transformation languages. Approaches for Aspect Oriented Design Modeling have been surveyed for example by [Wimmer 11].

## 3.5 Major Model Driven Engineering Initiatives

*Model Driven Architecture (MDA$^{TM}$)* [OMG 03] is an approach to system development that supports the Model Driven Engineering of systems. The requirements for the system are modeled in a *Computation Independent Model (CIM)*, a model which describes the system in the environment inside of which it will operate. The system itself is described by a *Platform Independent Model (PIM)*, a model that may contain enterprise, informa-

tion and computational specifications, but not details of the system's use of its platform. One or several platforms that enable implementation of the system are chosen. The model produced by the transformation of the Platform Independent Model for a specific platform and which specifies how the system uses the chosen platform, represents the *Platform Specific Model (PSM)*. Each of these three models may contain several models from different viewpoints.

To describe these three types of models, OMG proposes to use UML. *Unified Modeling Language (UML$^{TM}$)* [OMG 11b] is a language which provides system architects, software engineers, and software developers with tools for analysis, design, and implementation of software-based systems, as well as for modeling business and similar processes. It has a very (too) broad scope that covers a large and diverse set of application domains.

The *Object Constraint Language (OCL)* [OMG 10b] is a formal, declarative language used to describe expressions, rules on UML models. These expressions typically specify invariant conditions that must hold for the system that is modeled, or queries over objects described in a model.

To describe Model Transformations, OMG proposes the *Query/View/Transformation (QVT)* [OMG 11a], a hybrid declarative/imperative language, with the declarative part divided into a two-level architecture. The two declarative layers are: a user-friendly *Relations* language that supports complex object pattern matching, and a *Core* language, which is just as powerful, but simpler and more verbose than the *Relations* language. The QVT *Operational Mapping* language allows either to define transformations using a complete imperative approach (operational transformations) or to complement relational transformations with imperative operations which implement the relations (hybrid approach).

The OMG family of Model Driven Architecture languages (e.g. UML, OCL, QVT) are defined using the *Meta Object Facility (MOF)* [OMG 10a]. MOF is an extensible model-driven integration framework for defining, manipulating and integrating meta-data and data for Model Driven Architecturein a platform independent manner. It represents the Meta-Meta-Model (cf. Section 3.1).

There are numerous tools [2] for Model Driven Engineering, with different degrees of maturity and addressing different challenges. A survey of open source tools is done for example by [Hussey 10]. Another survey of Model Driven Engineering tools, for user interfaces, is done for example by [Pérez-Medina 07].

We focus on the Eclipse Modeling Project [3], which we appreciate is rapidly becoming the de facto open source standard implementation of the Model Driven Engineering vision. It contains all the official projects of the Eclipse Foundation focusing on model-based development technologies. Among them, we mention:

- Eclipse Modeling Framework (EMF): a modeling framework and code generation facility for building tools and other applications based on a structured data model;
- Graphical Modeling Project (GMP) : provides generative components and runtime infrastructures for developing graphical editors based on EMF and the Graphical Editing Framework (GEF). GEF provides technologies to create rich graphical editors and views for the Eclipse Workbench User Interface.

2. `http://mdwhatever.free.fr/index.php/2010/06/model-driven-tools-the-big-list/` , accessed on 06.11.2011

3. `http://eclipse.org/modeling/` , accessed on 06.11.2011

- UML2 Tools: a set of GMF-based editors (GMF is a project of GMP) for viewing and editing UML models.
- Atlas Transformation Language (ATL): a model transformation language and toolkit for model-to-model transformations.
- Xpand: a statically-typed template language featuring polymorphic template invocation, for model-to-text transformations (refinement).

While still not completely mature, many of these tools are powerful and have been used successfully in different modeling projects (cf. e.g. ATL Use cases [4]).

## 3.6 Model Driven Engineering for Telecommunications

We find it useful to think at Model Driven Engineering usage for telecommunications at two levels: the use of model-driven techniques by the Service Providers and Service Developers, and its use by the Tool Vendors. Proposals to use Model Driven Engineering by the Service Providers and Service Developers comprise:

- Managing the entire telecommunications service life-cycle [Azmoodeh 05], sometimes by providing "vertical slices" [Carroll 06]. Modeling business capabilities, constraints and context, roughly corresponds to the *Need Capture* phase and the *Analysis* and *Definition* sub-phases of the *Construction* phase of the telecommunications service life-cycle (cf. Section 1.1). It is suggested that the result of these activities be a Computation Independent Model. Modeling system capabilities, constraints and context, roughly corresponds to the *Specification* and *Verification* sub-phases of the *Construction* phase of the telecommunications service life-cycle (cf. Section 1.1). It is proposed that the result of these activities be a Platform Independent Model. Implementation capabilities, constraints and context, roughly correspond to the remaining sub-phases of the *Construction* phase of the telecommunications service life-cycle (cf. Section 1.1). It is suggested that they be done as a Platform Specific Model. Deployment capabilities, constraints and context, roughly correspond to the *Deployment* phase of the telecommunications service life-cycle (cf. Section 1.1) and are proposed to be done as a running system. A similar approach of modeling the telecommunications service life-cycle using Model Driven Engineering, but by slicing it according to concerns, is proposed by [Carroll 06]. One such "vertical slice" offers a cross-section of the telecommunications service definition aspects for a specific scenario.
- Modeling business capabilities [Ahuja 10]. One manner of describing the business and functional architecture of new services is as a sequence of tasks. This roughly corresponds to the *Definition* sub-phase of the telecommunications service life-cycle (cf. Section 1.1).
- Service discovery [Yang 06a], [Yang 06b]. One manner of creating new telecommunications services (cf. Section 1.5) is by combining already existing ones. This roughly corresponds to the *Specification* sub-phase of the telecommunications service life-cycle (cf. Section 1.1). However, before doing this, the fitting telecommunications services must be found, discovered, among the numerous existing ones. One approach is the model-based service discovery [Yang 06a], according to which service descriptions are automatically generated from models of services.

4. `http://www.eclipse.org/m2m/atl/usecases/`, , accessed on 06.11.2011

- Early *validation* (cf. *Validation* phase of telecommunications service life-cycle, Section 1.1) [Achilleos 08b]. The validation of a telecommunications service before implementation brings important benefits (e.g. reduced number of errors). There are a number of formalisms that can be used for the validation/testing of the telecommunications service specification (e.g. timed automata). Model Driven Engineering can be used in this context to transform the telecommunications service specification into code specific to that formalism (e.g. into Petri Nets [Achilleos 08b]).
- Bridging the gap between different levels of abstraction [Yang 05], [Yang 07]. Model Driven Engineering's iterative refinement capabilities, described for example in a policy-based Domain Specific Language [Yang 05], can be used to *specify* and *implement* (cf the *Construction* phase of telecommunications service life-cycle) telecommunications services (cf. Section 1.1).
- Automatic generation of context-aware services [Ou 06], [Georgalas 07]. During the *Operation* phase of the telecommunications service life-cycle (cf. Section 1.1), same services (e.g. pervasive ones), may depend on contextual information (e.g. finding restaurants around the current position of the user). One way to model this information is as an ontology. Generally, when using ontologies for this purpose, one describes a meta-ontology from which one generates context-specific ontologies, using Model Driven Engineering.



Indeed, Model Driven Engineering can be used all along the telecommunications service life-cycle, to describe and automate (sub-)phases, and to automate transitions between (sub-)phases.

Proposals to use Model Driven Engineering by the Tool Vendors suggest:

- Building tools that automate the passage of telecommunications services from one (sub-)phase of their life-cycle to the next one [Azmoodeh 05].
- Building a framework that allows (to Tool Vendors) to easily build domain-specific tools. Such approaches are not telecommunications-specific, but rather independent of the domain. For example, [Achilleos 07], [Achilleos 08a] proposes integrating existing Eclipse based tools (EMF, GMF, ATL, oAW) into such a framework, called *IEME - Integrated Eclipse Model driven Environment*. Another such framework is the *Java Application Building Center (jABC)* [Steffen 07], an Eclipse based framework for which plug-ins providing functionalities like animation, analysis, simulation, verification, execution, compilation can be developed. Users of jABC can develop services and applications by composing reusable building blocks into hierarchical (flow-)graph structures. jABC is a general-purpose framework; therefore, it has been used in very different projects and not only in the telecommunications domain. The specificity of jABC for telecommunications consists in the integration of Parlay X (cf. Section 1.4.2) services [Blum 09a].

Actually, if Tool Vendors use Model Driven Engineering tools, these are in fact Meta Tools (cf. Section 3.3). Tool Vendors may use Model Driven Engineering tools to build telecommunications specific tools, thus the tools they use are indeed Meta Tools. Of course, Tool Vendors may build these Meta Tools using other technologies, not only Model Driven Engineering.

Expected benefits of Model Driven Engineering for Telecommunications comprise [Azmoodeh 05]:

- More precise behavior specification in models;

– Potential reuse of models and their transformations;
– Reduced cost by defining transformations from model to model/code only once;
– Ability to adapt the languages to domain-specific needs.

Caveats of Model Driven Engineering forTelecommunications comprise:

– Maturity degree of Model Driven Engineering tools [Carroll 06]. If general-purpose Model Driven Engineering Meta Tools are used by Tool Vendors, these may not yet be mature enough. Of course, Tool Vendors have the option of building telecommunications specific tools using other technologies than Model Driven Engineering.

In conclusion, there are proposals to use Model Driven Engineering along the entire telecommunications service life-cycle, and to build Meta Tools that generate telecommunications service specific tools. These proposals present some of the important advantages of applying Model Driven Engineering for telecommunications.

## 3.7 Discussion

We have presented examples of the use of Model Driven Engineering in all phases of the telecommunications service life-cycle. Consequently, Model Driven Engineering should help telecommunications service creation as well. From the requirements for Service Creation Environment, Model Driven Engineering is expected to contribute towards:

– *Req 2 Domain specificity*: through the Meta-modeling approach for language definition, which can be used to define telecommunications specific Modeling Languages.
– *Req 3 Rapid prototyping*: through the high automation degree coming from the use of Model Transformations between models at different abstraction levels.
– *Req 5 Early verification/simulation*: either by using model-based testing/verification tools (reviewed for example by [Shafique 10]), or by integrating domain specific testing tools with Model Transformations;
– *Req 6 Integration*: it is to be expected that several Modeling Languages and associated tools will be used; using the Meta-modeling approach for language definition enables the use of Model Transformations to describe (syntactic) translations between them;
– *Req 7 Reuse*: due to many concepts, constructs, specific to telecommunications service construction are captured by the Modeling Languages; however, more powerful mechanisms (e.g. feature models) are needed to describe commonalities and variations between different telecommunications services (e.g. Software Product Lines);
– *Req 8 Wide range of services*: independence from the platform is achieved through the separation of the Computation Independent Model, the Platform Independent Model and the Platform Specific Model; hence, many (all) types of different networks can be supported;
– *Req 9 Easy evolution of services*: due to the high automation degree and automatic connections between models at different abstraction levels (i.e. Model Transformations), changes in requirements (e.g. from End Users) can be easily propagated in the telecommunications service specification, implementation, etc.

We find it useful to think at Model Driven Engineering at two levels: when used by Service Providers and Service Developers (i.e. the designers of a product/service), and when

used by Tool Vendors (i.e. those that provide software tools to the designers of a product/service). For the first group, Model Driven Engineering offers advantages of increased formality (with positive results like reduced number of errors), higher automation (with positive results like reduced time and cost, increased reuse, release of designers for more creative tasks), better interoperability. All these contribute to reduce the construction time for a new product/service. For the second group, the Tool Vendors, through Model Transformations, Model Driven Engineering increases the re-usability rate, allowing extensions and combinations of existing languages, generation of language specific software tools, definition and generation of bridges for language and tool interoperability. However, the cost of tools remains important, even when using a Model Driven Engineering approach, therefore, a uniform process that provides tools to all stakeholders would greatly benefit Tool Vendors.

# PART II : CONTRIBUTIONS

We propose, in Chapter 4, a telecommunications service construction process, inspired from the software waterfall development cycle. To build the dedicated tools used in this construction process, we propose a model-driven tool building process, in Chapter 5. The tools used by the Service Provider and Service Developer roles in the construction process, and which we built according to the tool building process, are presented in Chapter 6. An application of the telecommunications service construction process and tools to a complete case study is provided in Chapter 7.

TELECOM Bretagne

# 4 A Process for Telecommunications Service Construction

" "Fright is countered with fright, cunning with cunning!" " 117

Bilgames and Huwawa: 'The lord to the Living One's Mountain', in *The Epic of Gilgamesh. A new translation*, Andrew George, Penguin Books, 2000

In this chapter we present our proposal to *RQ 1 Construction process*, a telecommunications service creation process, inspired from the waterfall software development cycle. Among the numerous responsibilities of a stakeholder, we focus on four main activities he/she performs. After that, the model goes downstream, to the next stakeholder, which will perform a similar set of activities. These main activities are: *MA 1 Model* the telecommunications service from her point of view, *MA 2 Test* the model, *MA 3 Collaborate* with other designers that share her viewpoint and *MA 4 Inter-operate* with software tools of stakeholders from other viewpoints. To support the activity of modeling, we propose designers use Domain Specific Modeling Languages. To support testing, we provide the possibility of simulating models. To support collaboration, we focus on Design Rationale and propose designers use a Domain Specific Modeling Language for capturing it. To support interoperability between tools, we propose an automatic approach.

## 4.1 A Telecommunications Service Construction Process

We start by defining what a process is, and built upon definitions from software development. A software development *method* can be loosely defined as "a recommended collection of phases, procedures, rules, techniques, tools, documentation, management, and training used to develop a system" [Avison 03]. Another definition [Ramsin 08] describes it as consisting of two main parts:

- a set of modeling conventions (a Modeling Language);
- a *process*, which:
  - provides guidance as to the order of the activities;
  - specifies what artifacts should be developed using the Modeling Language;
  - directs the tasks of individual developers and the team as a whole;
  - offers criteria for monitoring and measuring a project's products and activities.

The process determines what activities should be carried out to develop the system, in what order, and how. In its most abstract form, a process is a sequence of steps to guide its

users in applying the Modeling Language for accomplishing a set of software development tasks. The process acts as the dynamic, behavioral component of the method.

The term *methodology* is frequently used with the same sense as the term *method*, therefore, we clarify here that we consider methodology to be the study of methods, as it is lexically correct [Graham 01].

After a survey of existing software and systems development processes, we choose a waterfall-like process to describe the activities done by telecommunications roles (cf. Section 1.2). This process reflects current practices in the telecommunications industry; therefore, it should be (easily) accepted by practitioners. The process for each stakeholder follows the same pattern, which consists of four main activities.

Among many software development processes, reviewed for example by [Ramsin 08], the most common ones are: waterfall, spiral, iterative, agile. The waterfall model is a sequential development approach, according to which development flows downwards through the phases of requirement analysis, design, implementation, testing (validation), and maintenance.

We choose a waterfall-like process, in which a role converses only with the roles immediately upstream and immediately downstream it in the telecommunications service lifecycle. In this process, each role performs the same activities, after which the model goes downstream, to the next role, which will perform a similar set of activities. This waterfall-like telecommunications service creation process reflects current practices in the telecommunications industry, according to which a telecommunications service is defined by using a stove pipe ”silo” architecture, without going back to a previous phase, and in isolation from other telecommunications services. As such, this process should be (easily) accepted by practitioners.

Teams of modelers are working for each role. Modelers are considered here to be: system architects (i.e. the high-level designer of a system to be implemented), system designers (i.e. somebody working under under the direction of a systems architect, designing, developing and implementing the clearly defined requirements of a new information system), software architects (i.e. the high-level, coarse grain, designer of a software to be implemented), software designers (i.e. usually an engineer doing low-level, fine grain component and algorithm implementation). The responsibilities of a modeler [Kruchten 08] are:

- *Define the architecture/design of the system.* A modeler abstracts the complexity of a system into a manageable model that describes the essence of a system by exposing important details and significant constraints. This comprises technical activities like: extracting architecturally significant requirements, translating business strategy into technical vision, exploring alternatives, making choices (e.g. technology selection), synthesizing a solution, validating it, defining the major modules of the system.
- *Maintain the architectural integrity of the system.* This is done through regular reviews, writing guidelines, and presenting the architecture/design/model to various parties, at different levels of abstraction and technical depth.
- *Participate in project planning.* This is done by assessing technical risks and working out mitigation strategies/approaches, or by suggesting the order and the content of development activities.
- *Discuss with implementation, integration, product marketing teams.* Due to their technical expertise, modelers are involved into problem-solving activities that go beyond strictly solving modeling issues. They have insights into what is feasible,

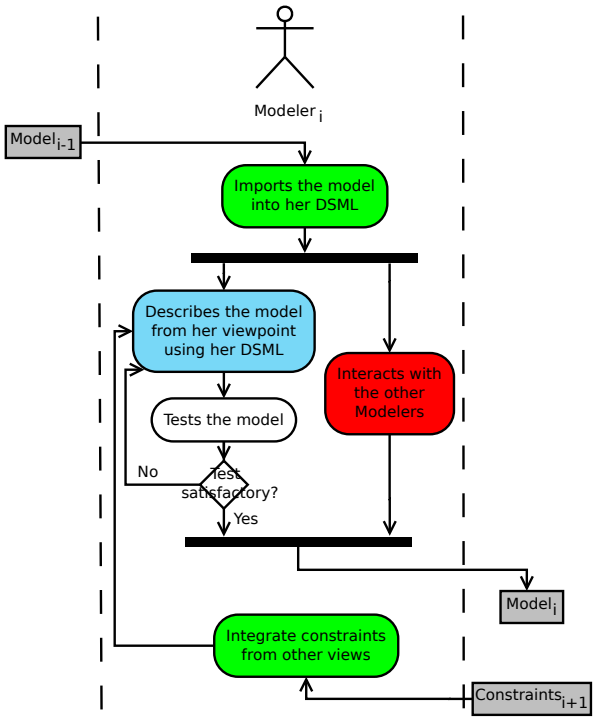


Figure 4.1 : The telecommunications service construction process from the Service Provider point of view.



and influence business strategy accordingly.

Some of these responsibilities are engineering, others are related to management, organizational politics, business strategy. Out of all modeler responsibilities, we focus on the following modeling activities, for modelers of all roles (cf. Figure 4.1):

1. *MA 1 Model*: modeling the service from their point of view. Each stakeholder/role describes the telecommunications service from his/her point of view, according to his/her concerns (cf. the 'Describes ...' activity in Figure 4.1).
2. *MA 2 Test*: testing/verifying the service modeled from their point of view. Early validation, verification or simulation can identify errors (more costly to repair later) or performance/dimensioning issues (cf. the 'Tests ...' activity in Figure 4.1).
3. *MA 3 Collaborate*: collaborating with other modelers from the same role. Of course, several professionals are working in each role. They interact with each other and with professionals from other roles (cf. the 'Interacts ...' activity in Figure 4.1).
4. *MA 4 Inter-operate*: inter-operating with the software used by modelers from other roles. Professionals also import/integrate/compose their models. The most important challenge introduced by this process is the interoperability of models developed in different views (cf. the 'Imports ...' and 'Integrate ...' activities in Figure 4.1).

Each activity will be presented in more detail in the next sections of this chapter.

The modeler begins by importing an existing model (if there is one). He/She adds information to this model according to his/her concerns, from his/her point of view. His/Her modeling may be influenced by constraints coming from other viewpoints downstream in the process. These constraints may either result from previous projects and so be inte-

grated in the tool he/she is using, or may result from (in)formal discussions with modelers from other roles. After developing a first version of the model, he/she may check it for conformance or simulate to find e.g. performance issues. According to the obtained results, he/she may change the model, then test it again. In conclusion, there are a number of iterations between these two activities. The modeler is not the only one describing and testing the model. Most often, he/she is part of a team, and as such collaborates with other modelers during the entire duration of the description and test activities. When the model reaches a satisfactory state (according to the modeler's experience), it is transfered to modelers of the next role downstream.

The same kind of activities are performed by the modelers of all roles directly involved in telecommunications service construction (i.e. all roles except the Tool Vendor). The four activities can be viewed as a pattern, which is repeated for each role (cf. Figure 4.2). The entire process is started by the End User, whose needs are captured in a first model. At the end of the process, the new telecommunications service has been constructed and is ready to enter the deployment phase.

The proposed telecommunications service construction process (Figure 4.2) is of course quite abstract and general, and therefore a number of details are ignored. It is at an abstraction level that intermediates between the very abstract level of the life-cycle (cf. Figure 1.1) and the concrete level of the usage of tools (cf. e.g. Figure 6.14). The life-cycle is a waterfall one, in the sense that e.g. sub-phase Specification is followed by sub-phase Verification, which in turn is followed by sub-phase Development, and so on. This is in regard to the main information being processed, of course there can be some information that goes upstream, from e.g. Verification to Specification (e.g. results and recommendations obtained from Verification), but this are much less quantitatively. The concrete usage of tools implies a much closer interaction between roles and a less clear separation between their activities. At this level, there is not a one-to-one relationship between roles and phases from the life-cycle or activities from the construction process.

The advantage of the suggested process is that the same activities are proposed to each role, which means that the Tool Vendor can re-use the same solution to *RQ 3 Tool building process*, for all roles (cf. Chapter 5). The disadvantage is that, being based on a waterfall model, the process lacks the flexibility of iterative processes for example. The quantity of information going upstream is rather small; once a role has finished its activities, there is no going back.

## 4.2 Modeling a Viewpoint

Modeling is the process of generating abstract, conceptual, graphical and/or mathematical models [1]. Models have been defined in Section 3.1. This *MA 1 Model* comprises technical activities that correspond to the *Define the architecture/design of the system* responsibility of the modeler. Modeling Languages are an important means of supporting these activities. A Modeling Language is, "a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system" [Booch 05].

In contrast with programming languages, which are mainly textual, Modeling Languages are mainly graphical. While textual languages can use any text editor and have a

1. `http://en.wikipedia.org/wiki/Scientific_modelling` , accessed on 06.11.2011

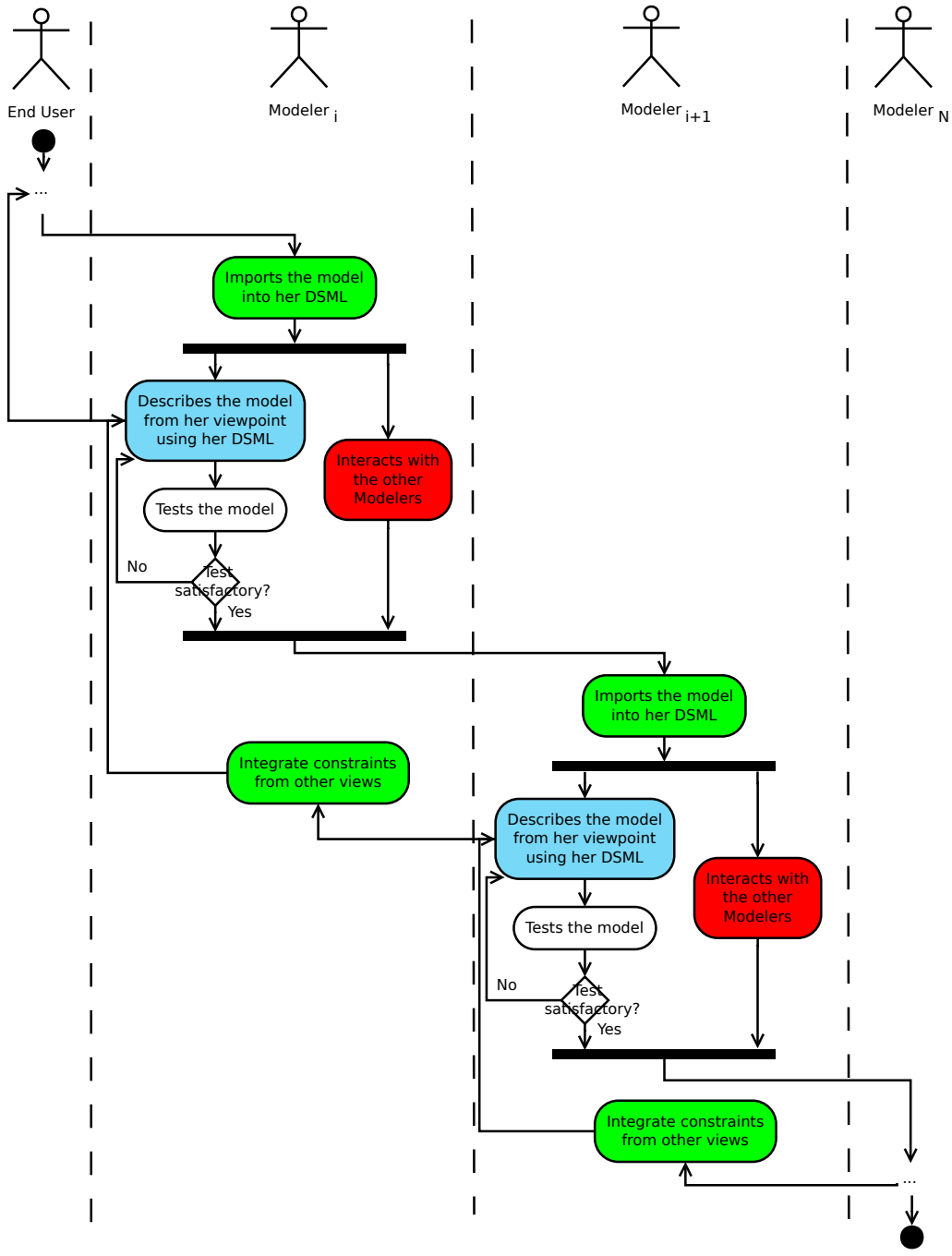


Figure 4.2 : The telecommunications service construction process from multiple points of view.

longer tradition of tool implementation, they are usually more difficult to understand and to use than graphical ones, which are more intuitive. Also, textual languages seem to be more appropriate for conveying a large number of details, while graphical ones are more appropriate for conveying the general image, the overall structure. Thus, graphical lan-

guages naturally impose themselves as dominant languages for modeling activities, which contain a lot of abstracting from details.

Another important aspect of languages is their degree of flexibility. A high degree of formality enables verifications and assistance, thus reducing the number of errors. However, in the exploratory phases of modeling, it is important that techniques and associated tools allow modelers (collaborative) creative freedom. Therefore, a compromise between the creative freedom and the degree of formality is needed [Chiprianov 10].

An equally notable aspect of Modeling Languages is their generality. General Purpose Languages (e.g. UML) are intended to be used in any domain. They can promote common understanding across different domains. However, because of their generality, the solution to a specific problem may be unnecessarily difficult to express and hard to understand. The concepts acquire several semantics from each domain in which they are used. Instead of promoting common understanding, they contribute to misunderstandings. A *Domain Specific Language (DSL)* is "a language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain" [Deursen 00]. Domain Specific Languages promise productivity increase [Gray 03] based on the conciseness of produced models. They are also easier to use, directly by domain experts [Deursen 00], they enable a high degree of reuse [Biggerstaff 98]. The main disadvantage of Domain Specific Languages is the additional cost of designing, implementing and maintaining it. Domain Specific Language development is hard, requiring both domain and language development expertise, which few people have [Mernik 05]. Therefore, it is important to balance the return, the benefits a Domain Specific Language generates, with the investment in its development, before deciding to create it. Empirical studies [Kosar 10] show that programmers' success rate is around 15% better for Domain Specific Language concerning learning, perception and evolution.

A *Domain Specific Modeling Language (DSML)* is taken in this thesis to be a graphical language that offers, through appropriate notations and abstractions, expressive power focused on a particular problem domain, to visualize, specify, construct and document the artifacts of a software-intensive system. Domain Specific Modeling Languages allow experts in general, and Service Providers in our case, to express, validate, modify solutions and achieve tasks specific to their domain. A Domain Specific Modeling Language requires less cognitive [Green 96] effort from experts than a more General Purpose Language, as it offers a higher closeness of mapping between the problem world and the solution world. Empirical studies conform that Domain Specific Languages are superior to General Purpose Languages in all cognitive dimensions [Kosar 10]. Thus quality [Baker 05], productivity [Kieburtz 96] [Baker 05], reliability [Kieburtz 96] [Spinellis 01], maintainability [van Deursen 98] [Baker 05], re-usability [Ladd 94] [Krueger 92] and flexibility [Kieburtz 96] can be enhanced.

Domain Specific Modeling Languages, by their definition, have reduced dimensions. Therefore, when they describe large domains, or different viewpoints for a domain, it is natural to define several Domain Specific Modeling Languages. While they are similar, they do differ in terms of syntax and semantics. A *family of languages* is "a set of related languages that offer similar functionality at an equivalent level of abstraction" [Ober 08]. Their main advantage is the systematic reuse of languages that are members of the family in defining new members [Evans 02]. Their main issue is their interoperability. As we propose to define a Domain Specific Modeling Language for each role, we will also have a

family of languages.

Our proposal for the modeling activity is to integrate Domain Specific Modeling Languages, for each role, into the process of telecommunications service creation. In this way, all roles can model while enjoying the benefits of Domain Specific Modeling Languages. More specifically, this meets Service Providers and Service Developers requirements for the Service Creation Environment:

- *Req 2 Domain specificity* - obviously, through the very definition of the Domain Specific Modeling Language;
- *Req 3 Rapid prototyping* - Domain Specific Modeling Languages, are graphical, and have a high abstraction degree, hence they are quite appropriate for prototyping. Also, their code generation capabilities reduce the creation time of a new service;
- *Req 8 Wide range of services* - Domain Specific Modeling Languages have a high degree of abstraction and the models produced with them are independent of lower level details, so they can describe services across all types of technical platforms;
- *Req 9 Easy evolution of services* - again, Domain Specific Modeling Languages are independent of technical platforms, and have code generation capabilities.

## 4.3 Testing of Telecom Services Models through Simulation



Testing is the process of executing a software or system with the intent of finding errors [Myers 04]. This *MA 2 Test* involves any activity aimed at evaluating an attribute or capability of a system and determining whether it meets its required results [Hetzel 91]. One means of evaluating the capability of a system is through simulation. System simulation is a set of techniques for using computers to imitate the operations of various kinds of real-world facilities or processes [Law 07].

We propose to effectuate the testing activity after modeling and before implementation. This way, any necessary changes that are detected can be handled early in the Specification sub-phase of the telecommunications service life-cycle. Related approaches for early testing of telecommunications services include for example the use of Petri Nets [Achilleos 08b], checking various static constraints defined in the language semantics [Combes 99], automatic test generation [Cavalli 96], observation during simulation [Combes 99], multilayer and multimedia traffic modeling [Ince 06], [Golaup 06] and the use of network simulators [Kubinidze 06].

Network simulators [Ince 07] (i.e. software programs that imitate the operation of a computer network) have become popular in telecommunications [Sarkar 11]. They offer sophisticated and powerful simulation packages, as well as high flexibility in model construction and validation. Ten simulators are compared by [Sarkar 11], who finds that the most frequently used network simulators in scientific articles are NS-2 (the latest version is NS-3 [2]) and OPNET [3].

As such powerful software tools already exist for telecommunications service simulation, we leverage them by ensuring the interoperability between the Domain Specific Modeling Languages used for the *MA 1 Model* and a network simulator, which is used

2. `http://www.nsnam.org/` , accessed on 06.11.2011
3. `http://www.opnet.com/` , accessed on 06.11.2011

for the *MA 2 Test*. In this way, the models produced using the Domain Specific Modeling Languages are fed as input to simulation tools. The interoperability/translation from Domain Specific Modeling Languages and the internal format of the network simulator is of course automatized (e.g. using Model Transformation approaches - see Section 3.2). The simulation results are then used in the modeling activity to make adjustments.

More specifically, this meets Service Providers and Service Developers requirements for the Service Creation Environment:

- *Req 2 Domain specificity* - through the re-use of existing telecommunications-dedicated network simulators;
- *Req 3 Rapid prototyping* - as the testing through simulation activity is right after the modeling activity and before implementation, performance and dimensioning issues can be tackled rapidly;
- *Req 5 Early verification/simulation* - effectuating the testing activity after modeling and before implementation;
- *Req 9 Easy evolution of services* -as the transformation from Domain Specific Modeling Languages and the network simulator is automatic.

However, the use of network simulators brings in other issues, such as:

- the level of details to be included in the model (which is by definition at a high level of abstraction) so as to ensure that the simulator has enough information;
- the balancing of different concerns and the choice among them (e.g. increasing throughput - beneficial - may mean adding extra hardware, so increased costs - disadvantageous). Decision aid systems may be considered to help tackle this issue;
- round-trip testing/simulation, consisting in (partially) automatizing the reflection of the *MA 2 Test* results into the *MA 1 Model*.



## 4.4 Collaboration inside a Role

Collaboration is a process where two or more stakeholders (persons or organizations) work jointly to create or achieve the same thing/aim. They do this by sharing knowledge, learning and building consensus [Merriam-Webster 11], [Learner 11]. It addresses the *MA 3 Collaborate*. It is a vast subject. A number of articles survey it, for example [Li 06a], [Li 06b], [Shen 08], [Wang 02], [Renger 08a].

However, an important and common issue of collaboration is information seeking, which consumes much of software engineers' time (e.g. 31,90% according to [Goncalves 09]). Collocated software development teams lack mostly information regarding design and program behavior [Ko 07]. *Design Rationale (DR)* is the justification behind decisions, the reasoning that goes into determining the design of the artifact [Dutoit 06]. Hence, capturing Design Rationale and enabling its retrieval (i.e. capitalizing on the experience) would significantly reduce the time alloted to seeking information, and would increase the performance of collaboration.

Design Rationale supports collaboration also by [Dutoit 06]:

- promoting coordination in design teams,
- exposing differing points of view,
- facilitating participation and collaboration,
- building consensus.

The main activities involving Design Rationale are:

1. *Capture*: the process of eliciting rationale from designers (e.g., directly from designers themselves, or through an automatic manner) and recording it. Designers may be quite reluctant about capturing rationale [Dutoit 06]. One of the most important factors of this resistance seems to be the intrusiveness of the capture process (e.g., the actual work required for capture, the disruption in designers' thinking, the possibility of later using the information against the very designers that captured it). To solve the capture problem, one direction explores reducing its intrusiveness. Proposals include finding when rationale is elicited naturally as part of design communication and capturing at that moment, and integrating rationale artifacts with the system models [Wolf 07], [Wolf 08]. This last proposal ingrains the communication channels in the system models, thus reducing the participants' reluctance to communicate and to capture raising issues.
2. *Formalization*: the process of transforming rationale into the desired representation form. The most commonly used way of formalizing Design Rationale is as a type of argumentation that is structured according to a given schema. There are many ways in which Design Rationale argumentation may be structured. One branch of thought uses a group of Design Rationale schemas having Issue-Based Information System (IBIS) [Kunz 70], Questions Options Criteria (QOC) [MacLean 91], and Decision Representation Language (DRL) [Lee 91] among its most prominent members.
3. *Retrieval*: the process of getting recorded rationale to the people who need it and providing access to it. Knowledge is thus capitalized upon and transmitted between projects. The most common approach to accessing Design Rationale is through use of a system that lets users browse a hyper document containing the rationale. Conventional information retrieval search techniques can also be used.



To address the Design Rationale capture problem, we propose formalizing the rationale with a Domain Specific Modeling Language which is integrated with the telecommunications service modeling Domain Specific Modeling Languages. This Design Rationale Domain Specific Modeling Language is transversal to all Domain Specific Modeling Languages intended for roles, is independent of them, and in general, of the domain. It can be used not only for telecommunications, but for any domain that implies decisions and requires their capture. The Design Rationale Domain Specific Modeling Language is based on argumentative formalization of Design Rationale, and contains concepts and constructs from the representation schema. One or a combination of schemas like IBIS, QOC, DRL can be used. Modelers use it during the *MA 1 Model*, at the moment which is natural to them. This is expected to reduce the Design Rationale capture intrusiveness. For Design Rationale retrieval, existing navigation or query-based systems can be integrated with the Domain Specific Modeling Languages framework.

More specifically, this meets Service Providers and Service Developers requirements for the Service Creation Environment:

- *Req 4 Collaborative support* - through the Domain Specific Modeling Language that formalizes Design Rationale. This information is essential, sought after and much needed in software engineering projects;
- *Req 6 Integration* - as rationale between decisions is explicitly captured and can be retrieved by all modelers working on the same model, putting together their parts of the model is easier. Also, transfer of models between roles is facilitated; consequently role integration is enhanced;

– *Req 7 Reuse* - retrieving rationale from existing models can influence the quality of current models, can reduce decision time;
– *Req 9 Easy evolution of services* - it is necessary to understand the current versions before changing / evolving a model. Being able to retrieve the rationale behind decisions taken for the current version of the model speeds up the understanding process and avoids dead end situations or previously erroneous zones.

## 4.5 Towards Automatic Semantic Interoperability of Modeling Languages

Our process proposes to model a telecommunications service using Domain Specific Modeling Languages for the different viewpoints that characterize the roles involved. While this makes complexity manageable, it introduces the difficult issue of interoperability. The modelers from different roles exchange models in the waterfall-like cycle with the modelers from adjacent roles (i.e. just upstream and downstream of them). For these models to be exchanged, their format/schema must be understood by the tools of the adjacent role downstream. We propose to ensure this understanding, interoperability, in an automatic manner, so the only activity of the modeler be "Imports the model into her DSML" (cf. Figure 4.1). However, to ensure this, a few considerations on the definition and scope of interoperability must be made.

There are numerous definitions for interoperability in literature, depending on the domain. For our purposes, and following [Peristeras 06], we consider interoperability to be the ability of two or more *tools* to *exchange models* and to use them in order to operate effectively together. Considering this definition, to operate together, tools for adjacent role Domain Specific Modeling Languages need to exchange models. Models are conformant with Meta-Models and in a Meta-modeling approach for language definition, Meta-Models define (the syntax of) Domain Specific Modeling Languages. Therefore, the issue of tools exchanging models written in different Domain Specific Modeling Languages becomes an interoperability issue between Domain Specific Modeling Languages. Hence, to ensure interoperability between models, one must address interoperability between Domain Specific Modeling Languages.

Interoperability is a complex problem; hence there are numerous proposals of decomposing it into levels. One particularly suitable for our approach is the C4IF (Connection, Communication, Consolidation, Collaboration Interoperability Framework) [Peristeras 06]. This is due to its mapping between Information Systems Communication and Linguistics. Linguistics and the Meta-modeling approach for language definition share concepts (e.g., syntax, semantics), thus establishing the connection with C4IF. The C4IF defines four levels:

1. *Connection*: the ability of Information Systems to *exchange signals*.
2. *Communication* refers to the ability of Information Systems to *exchange data*. *Syntactic* communication includes data in commonly accepted data syntax/schemas.
3. *Consolidation* refers to the ability of Information Systems to *understand data*. The focus is on data meaning (i.e., *semantics*).
4. *Collaboration* refers to the ability of Information Systems to *act together*.

These levels of interoperability are usually defined in such a manner so as to ensure a (strict) linearity [Peristeras 06] between them - to reach an upper level of interoperability,

all the previous levels must have been successfully addressed.

In order to ensure interoperability between two Domain Specific Modeling Languages we ideally have to ensure all four levels of the C4IF. The Information Systems of C4IF, in our case, are the tools associated to Domain Specific Modeling Languages, and the data they exchange, are thus the models. We consider the C4IF *Connection* level as being implemented by existing communication and signaling media in computers.

The mapping proposed in [Peristeras 06] assigns the *Communication* interoperability level of Information Systems Communication to *Syntax* of Linguistics. So, communication, *syntactic interoperability*, between Domain Specific Modeling Language tools, is the level of interoperability between the syntaxes of Domain Specific Modeling Languages. Approaches to ensure syntactic interoperability between different Domain Specific Modeling Languages have been proposed, like combining them [Vallecillo 10]: extension, merge, embedding, weaving or hybrid approaches. These are powerful and satisfactory approaches.

The mapping proposed in [Peristeras 06] assigns the *Consolidation* interoperability level of Information Systems Communication to *Semantics* of Linguistics. So, consolidation, *semantic interoperability*, between Domain Specific Modeling Language tools, is the level of interoperability between the semantics of Domain Specific Modeling Languages. Semantic interoperability is still a research topic, and we propose a solution for ensuring it in Section 5.4. We do not yet address *Collaboration*, as we consider, in conformance with their (strict) linearity property, that the *Consolidation* level must be ensured first.

To conclude this section, we indicate how the process for assuring *MA 4 Interoperate* meets Service Providers and Service Developers requirements for the Service Creation Environment:

- *Req 1 An overall model* - by providing a means of (semi-)automatically ensuring interoperability between Domain Specific Modeling Languages of two adjacent roles, the models produced with the respective Domain Specific Modeling Languages have the possibility of accumulating descriptions from multiple viewpoints corresponding to the involved roles;
- *Req 3 Rapid prototyping* - due to the (semi-)automatic nature of the interoperability solution, based on Model Transformations and Higher Order model Transformations;
- *Req 6 Integration* - obviously, by determining the shared ontologies and using them to generate the Model Transformations between the two Domain Specific Modeling Languages;
- *Req 9 Easy evolution of services* - again, due to the (semi-)automatic nature of the interoperability solution.

## 4.6 Discussion

Although we propose a process involving four modeling activities, there are other activities of the modeler that have not been addressed in this chapter (e.g., project planning, consulting with other teams). These are not in the scope of this thesis.

Our proposal for *MA 1 Model*, defines a Domain Specific Modeling Language for each role, while promising increased expressive power and thus higher modeler performance. This is a difficult and costly task. It would be beneficial if some of these disadvantages

were reduced. One manner of doing this is by defining Domain Specific Modeling Languages as extensions to existing Modeling Languages.

We tackle *MA 2 Test* by suggesting integration with existing widely used telecommunications-specific software tools, network simulators. However, round-trip testing (i.e. integrating automatically the test results, simulation feedback, into the model) would be highly useful. Also, integration with many other testing tools, capable of analyzing different aspects of telecommunications models, is needed.

Like testing, *MA 3 Collaborate* is a vast subject. We have tackled one of the most important issues, the lack of rationale behind modeling decisions. We built on Design Rationale research to propose a Domain Specific Modeling Language based on schemas and integrated with the telecommunications Domain Specific Modeling Languages. This integration is expected to reduce the capture intrusiveness problem and determine modelers to use the Design Rationale Domain Specific Modeling Language. More collaboration software tools and devices (e.g.; multi-writer, real-time, distributed collaborative editors that also include chat, white-board, and screen sharing functionality[Prechelt 11]; multi-touch table; interactive white-boards [Renger 08b]) should be integrated.

To help the modeler in his/her *MA 4 Inter-operate*, we propose to ensure Domain Specific Modeling Language interoperability automatically, at the syntactic and semantic levels. However, this does not address the flow of information upstream, and it is the modeler's responsibility to integrate constraints that may come from other views, roles (cf. activity 'Integrate...' from Figure 4.1).

The telecommunications service creation process proposes a similar set of activities for modelers of all roles. The advantage is that the same activities are proposed to each role, which means that the Tool Vendor can re-use the same solution to *RQ 3 Tool building process*, for all roles (cf. Chapter 5). The disadvantage is that, being based on a waterfall model, the process lacks the flexibility of iterative and agile processes for example. Although there is some information that goes upstream, this is rather reduced; once a role has finished its activities, there is no going back upstream. The proposed process illustrates the integration of Domain Specific Modeling Languages into the software engineering process for creating telecommunications services, and the interaction of the *MA 1 Model* with other modelers' activities.

We presented in this chapter our proposal for a telecommunications service creation process. It reflects current practices in the telecommunications industry, according to which a telecommunications service is defined using a stove pipe 'silo' approach, without going back to the previous phase, and in isolation from other telecommunications services. As such, this process should be (easily) accepted by practitioners. Of course, in the future, one can envisage more flexible processes for integrating End User feedback more rapidly. These processes could be inspired from iterative software development methods; in this wider context, our proposed process is a necessary transitional step. Having introduced this process, we now need to introduce the Tool Vendor process for building the tools that stakeholders use to implement/follow it.

# 5 A Tool Building Process for Service Tool Vendors

" 'By the life of the [mother who bore you, the goddess Ninsun], and your father, [the pure Lugalbanda,] 93
may your god Enki-[Nudimmud inspire] your words:' "

Bilgames and Huwawa: 'Ho, hurrah!', in
*The Epic of Gilgamesh. A new translation*, Andrew George, Penguin Books, 2000

In this chapter we present our proposal to *RQ 3 Tool building process*. We focus on building the tools for the four main activities that form the template of our telecommunications service construction process. To define the modeling and collaboration Domain Specific Modeling Languages, build their software tools, generate configuration files to existing network simulation tools, as well as ensure automatic interoperability, we propose a tool building process, targeted at Tool Vendors. The tasks for defining Domain Specific Modeling Languages are inspired from the meta-modeling approach for modeling language definition. Enabling testing through simulation is based on generative approaches that take as input models. The integration of the collaboration Domain Specific Modeling Language and viewpoint specific Domain Specific Modeling Languages is based on model combination. Ensuring interoperability is done by lifting the Meta-Models into ontologies, followed by their alignment, and generation of Model Transformations.

Integrating Domain Specific Modeling Languages into software engineering processes, in addition to the language development process to be followed, introduces issues like: leveraging Off the Shelf (OTS) components, combining several Domain Specific Modeling Languages, ensuring interoperability between Domain Specific Modeling Language associated tools. These issues are tackled in this chapter.

Tool Vendors may have different areas of interest. A market segmentation study [Frank 07] uses a framework of four types of Tool Vendors, depending on two dimensions: focus on telecommunications and breath of operations, both of which can have values between low to high. The Tool Vendor types are:

- Specific software companies, with both a low breath of operations and a low focus on telecommunications;
- Generic software manufacturers, with a high breath of operations and a low focus on telecommunications;
- Generic telecommunications vendors, with both a high breath of operations and a high focus on telecommunications;

– Niche vendors, with a low breath of operations and a high focus on telecommunications.

The same study [Frank 07] reveals that most of the vendors have a high focus on telecommunications. Therefore, the process we propose is focused on them. Although it can be generalized to other domains as well, our process is adjusted to the needs of telecommunications Tool Vendors (generic telecommunications vendors and niche vendors).

## 5.1 The Meta-Modeling Domain Specific Modeling Language Definition Process for Tool Vendors

### 5.1.1 General Domain Specific Modeling Language Development Process

Domain Specific Modeling Language development approaches, for example [Ceh 11] (a developed version of the one presented in [Mernik 05]) and [Strembeck 09] (focused on Meta-Model-based Domain Specific Modeling Languages), propose processes with several phases. We present here the process from [Ceh 11], as the more straightforward and general one. It consists of seven phases:

1. *Decision*: Characteristics of problems deserving a Domain Specific Modeling Language approach are indicated for example by [Sprinkle 09]. These include: a well defined domain, a domain with repetitive elements (e.g. multiple products, features), a growing developer community, the existence of a clear process from requirements to the execution code which could be automatized, an already existing well-accepted representation, and intended use by a domain expert.
2. *Domain analysis*: The goal of this phase is to select and define the domain of focus and collect appropriate domain information and integrate it into a coherent domain model. Information sources for analysis are: technical documents, existing implementations, customer demands, discussions with domain experts, current and future requirements. Outputs of this phase include: domain specific terminology and semantics. Although various formal methodologies exist (e.g., Feature-Oriented Domain Analysis FODA [Kang 90] with its various extensions reviewed by [Schobbens 06], Organization Domain Modeling ODM [Simos 95], Family-Oriented Abstractions Specification and Translation FAST [Coplien 98], Ontology-based Domain Engineering ODE [de Almeida Falbo 02]), domain analysis is still done informally most of the time [Ceh 11]. A possible solution may be Semi-automated Domain Modeling [Iris 10], an approach which matches, merges and generalizes multiple application models in a domain, taking into consideration syntactic, semantic and structural aspects.
3. *Design*: In this phase, the syntax and semantics of the language are defined. The main definition methods [Mernik 05] are: inventing a new language from scratch and starting from an existing language. There are three main ways of re-using a base language: by piggybacking (partially reusing it), specializing (restricting it), extending (adding new features or constructs). The syntax and semantics can be described with a formal (e.g.; BNF, Meta-Models for syntax; grammars, denotational semantics for semantics) or informal method (e.g. natural language).
4. *Implementation*: Consists in building the language tools that describe the execution

semantics. Approaches include: compiler/interpreter, source-to-source transformation, embedding (new abstract data types and operators are defined on top of those of a base language), compiler generation, extensible compiler, etc. A comparison of ten such implementation approaches is presented by [Kosar 08]. It seems to show that the implementation effort and the End User effort of using the Domain Specific Modeling Language are almost inversely proportional (first approach according to one of the measures ranked penultimate according to the other). As low End User effort should be kept as low as possible, the implementation effort needs to be reduced. For this, we need to generate more model-based tools (e.g., interpreters, simulators, debuggers [Mannadiar 11], verifiers, unit testing frameworks [Wu 09], editors). This is what a number of approaches for defining (behavioral) semantics for Meta-Model-based Domain Specific Modeling Languages, surveyed by [Bryant 11], are trying to achieve in a formal, yet widely usable manner.

5. *Testing*: The Domain Specific Modeling Language is evaluated. Domain Specific Modeling Language success factors that should be tested, e.g. through questionnaires, have been identified e.g. by [Hermans 09]: learnability, usability, expressiveness, re-usability, development costs and reliability.
6. *Deployment*: Introducing a language into a community can engender technological, organizational and even social issues. Special attention should be given to integrating the new tools with legacy methods and infrastructure.
7. *Maintenance*: The Domain Specific Modeling Language is updated to reflect new requirements, but updates need to maintain consistency and continuity between versions of language and associated tools. A framework for the evolution of Modeling Languages is introduced by [Meyers 11]. It co-evolves the syntax, semantics and transformations through migration operators.

Most language developers seem to focus on the *Design* and *Implementation* phases. The least known and least examined phase is *Domain analysis* [Ceh 11]. *Testing* is often skipped or relaxed by language developers, mainly because of a widely accepted Domain Specific Language validation method [Gabriel 10].

The Domain Specific Modeling Language development method introduced by [Strembeck 09] defines a process with four activities (each of them further refined). It is focused on Meta-Model-based Domain Specific Modeling Languages, and so less general than the process of [Ceh 11]. The first activity, *Define DSL core language model* is roughly equivalent to the *Domain analysis* phase. There is no *Decision* task, rather the selection of the target domain is considered to be the triggering event. The second and third activities, *Define DSL behavior* and respectively *Define DSL concrete syntax*, are roughly equivalent to the *Design* phase. The fourth task, *Integrate with DSL platform*, is roughly equivalent to the *Implementation* and *Deployment* phases. All four tasks contain testing sub-tasks, which are roughly equivalent to a distributed *Testing* phase. There is no task for maintenance/evolution.

Concerning families of Domain Specific Modeling Languages, one promising method is to use Software Product Line (SPL) to model them [White 09]. Software Product Lines comprise software engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of

production [1]. Software Product Lines use feature models, which can describe commonalities and variabilities between Domain Specific Modeling Languages.

### 5.1.2 Domain Specific Modeling Language Definition Process for Telecommunications Tool Vendors

The process we propose to Tool Vendors is an instantiation of the general Domain Specific Modeling Language process presented in the previous section, focused on their practical needs. As our Domain Specific Modeling Languages are graphical, and because we want to take advantage of the Meta Tools (cf. Section 3.3), we focus on model driven approaches. We also concentrate on the *Design* and *Implementation* phases. These are the essential initial phases for defining a Domain Specific Modeling Language. We consider *Decision* more of an event born from a need, than a phase. In practice, *Domain analysis* is one of the great challenges of Domain Specific Modeling Language development, and is often done informally, through discussion with domain experts. *Testing* is also done informally, through presentations to the end-users. Concerning *Deployment*, the previous discussions with domain experts and end-users facilitate the introduction and adoption of the new Domain Specific Modeling Language in the community. We address technological integration as a separate issue and dedicate another process to it (cf. Section 5.4). As our process uses Model Driven Engineering, to ensure *Maintenance*, operators derived from model differencing, comparison, evolution (at present hot research topics) can be leveraged, as proposals like [Meyers 11] begin doing.

The process we propose for the *Decision* and *Implementation* phases of the Meta-Model-based Domain Specific Modeling Languages (cf. blue activities from Figure 5.1), is based on the Meta-modeling approach for language definition described for example by [Clark 01] and [Kurtev 06] (cf. also Section 3.3). It has at its center the Meta-Model used to describe abstract syntax (cf. activity "Define $MM_i$" from Figure 5.1). For the concepts present in this Meta-Model, another Meta-Model which describes concrete syntax is defined (cf. activity "Define display-surface MM" from Figure 5.1). The link between the two can be described with Model Transformations or it can be implemented in the programming language into which the Meta-Models are translated. The Meta-Model describing concrete syntax is used as input to a semi-automated process of generating the editor (cf. activity "Generate graphical editor"). Starting from the Meta-Model describing abstract syntax again (which justifies the claim that this Meta-Model is at the center of the process), Domain Specific Modeling Language semantics is described. We propose to implement Domain Specific Modeling Language semantics through template-based code generation (cf. Section 3.2) techniques (cf. activity "Generate code template-based" from Figure 5.1). This process can be employed for Domain Specific Modeling Languages of all roles, resulting in a family of Domain Specific Modeling Languages. This family has to be tested, validated, by end-users. This usually involves iterations (cf. task "Validate DSML family" and decision block "Ok?" from Figure 5.1).

1. `http://en.wikipedia.org/wiki/Software_product_line` , accessed on 06.11.2011

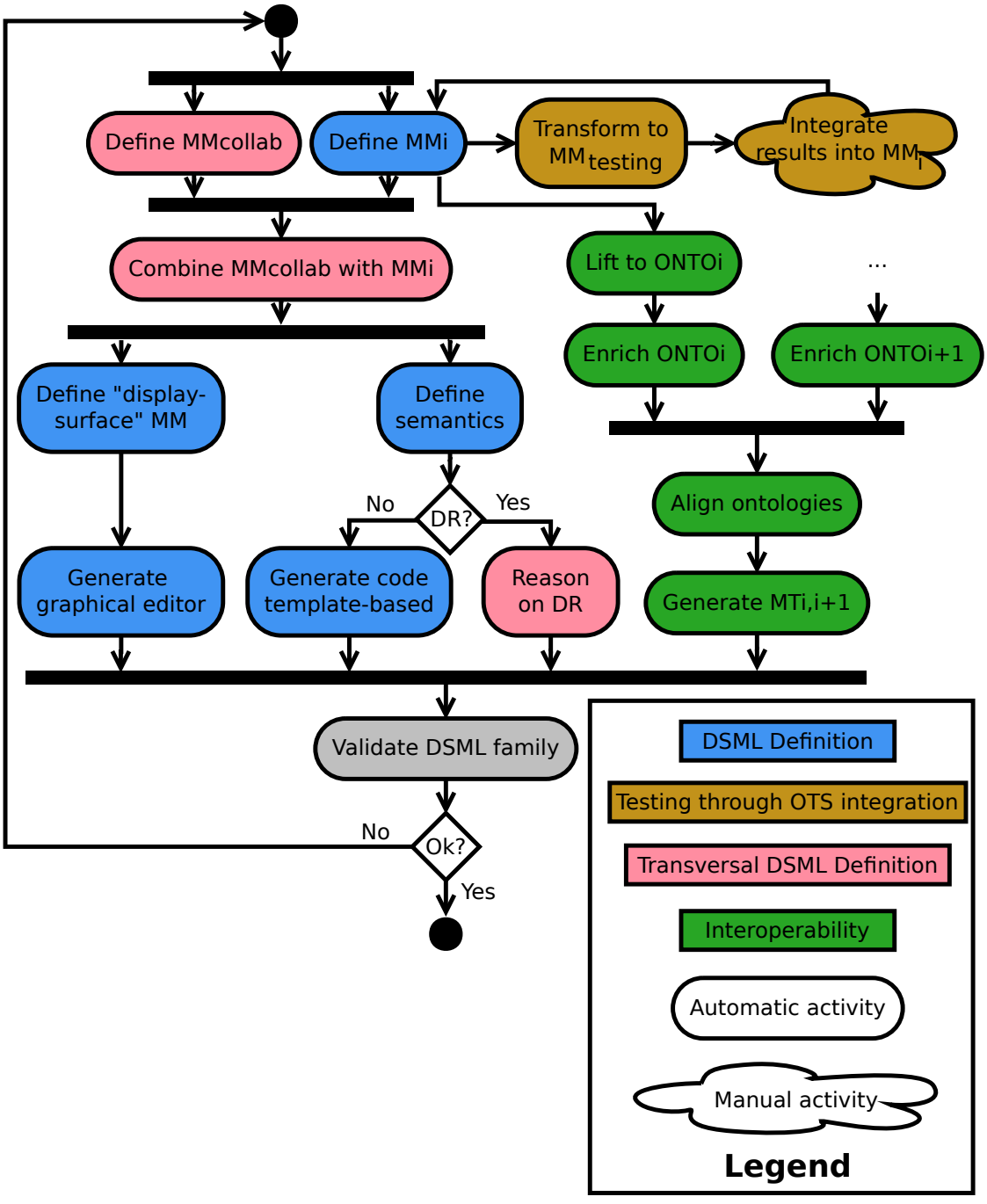


Figure 5.1 : The Tooling Approach for Service Tool Vendors.

## 5.2 Enabling Testing through Leverage of Off the Shelf Components

### 5.2.1 General Process for the Integration of Off the Shelf Components

The term Off the Shelf (OTS) component is very generic; it can refer to many different types and levels of software. We adopt the definition as stated by [Torchiano 04]: "a commercially available or open source piece of software that other software projects can reuse and integrate into their own products". It includes both components/services acquired for a fee (known as Commercial Off the Shelf COTS) and from Open Source communities (known as Open Source Software OSS).

Advantages of Off the Shelf components include:

1. Savings in development resources: reuse of Off the Shelf components means less implementation;
2. Increased potential for improved quality, enhanced reliability, availability, maintenance: the savings in development resources can be redirected towards these areas [Sedigh-Ali 01];
3. Reduced development costs: due to their large user base, Off the Shelf tools can be cheap;
4. Faster technology innovation: due to reuse;
5. Reduced time-to-market: due to faster technology innovation [Jansen 08], [Ncube 08];
6. Increased reliability: as they are used in numerous projects, they are activated and verified in potentially different situations [Land 09];
7. Functionally powerful: they offer many functions to answer to general needs.

However, there are also caveats when using Off the Shelf components:

1. Integration effort: Off the Shelf tools are build as stand-alone applications and make assumptions about their environment that do not hold when used as part of larger software systems [Egyed 01], [Li 08];
2. Inefficient debugging: while they may offer programmatic interfaces, they usually do not offer notification and data synchronization facilities needed for debugging [Li 08];
3. Difficulty of selection: choosing the right component in a systematic, explicit, objective and cost-efficient way is difficult. There are many processes and methods of doing this, reviewed for example by [Land 08], [Ayala 11];

The particular nature of Off the Shelf components requires a particular process of using them. One such process for vendors is proposed by [Morisio 02]. It consists of four phases, in which Off the Shelf specific activities are combined with non-Off the Shelf activities. The four phases, with the Off the Shelf specific activities, are:

1. *Requirements*: the tools/components/packages are identified/evaluated/selected. As underlined in the disadvantages, this is a major issue;
2. *Design*: the glue-ware and interfaces are identified;
3. *Coding*: the glue-ware and interfaces are written/implemented;
4. *Integration*: the components are integrated together and into the system. The system is tested. The acceptance system test is the final activity.

There are also two review activities, after the *Requirements* and the *Design* phases, and the customer should be involved in the *Requirements* phase.

### 5.2.2 Leveraging Network Simulators for Telecommunications Service Testing

Due to the fact that there are numerous mature simulation and testing tools for telecommunications, we propose leveraging them as Off the Shelf tools. We do not discuss the arduous activity of Off the Shelf selection, considering it the appanage of domain experts. We focus instead on the *Design* and *Coding* phases (cf. activity "Transform to $MM_{testing}$" from Figure 5.1). As a consequence of our proposal to use Model Transformations for the *Coding* phase, *Integration* of information from the main system to the Off the Shelf components is no longer be a problem. However, the flow backwards remains an issue (cf. activity "Integrate results into $MM_i$" from Figure 5.1).

The *Design* phase of our process consists in constructing a Meta-Model that captures the data used by the Off the Shelf tool that is to be integrated. The *Coding* phase consists in writing a Model Transformation between the Meta-Model of the Domain Specific Modeling Language and the Meta-Model of the Off the Shelf tool produced in the *Design* phase. In this way, the flow of information, represented by the models obtained with the Domain Specific Modeling Language and inputed in the Off the Shelf tool, is automated. This solves the *Integration* issue between the main system, the Domain Specific Modeling Language, and the Off the Shelf tools. However, as many of the results of the Off the Shelf tools have a visual nature, they can be accessed only by their end-users, not by programmatic means. Therefore, they should interpret the results and modify the model accordingly.

Leveraging Off the Shelf tools is possible for other modeling activities as well, not only for testing. For example, existing tools for communication (e.g. chat, VoIP) could be integrated using the process proposed here, helping the *MA 3 Collaborate*.

## 5.3 Towards Collaboration through Combination with a Design Rationale Domain Specific Modeling Language

From the large field of collaboration, we have decided to focus on Design Rationale and build a Domain Specific Modeling Language dedicated to it. The process of defining the Design Rationale Domain Specific Modeling Language is the same as the one used for the system Domain Specific Modeling Languages (cf. Section 5.1.1). The Meta-Model describing abstract syntax, based on schema-structured argumentation formalization approaches (e.g. IBIS, QOC, DRL), is defined (cf. activity "Define $MM_{collab}$" from Figure 5.1). However, the Design Rationale Domain Specific Modeling Language is transversal to system Domain Specific Modeling Languages. It has to be combined with each of the system Domain Specific Modeling Languages, in order to be used by modelers from each role. There are many Domain Specific Modeling Language combination approaches, reviewed for example by [Vallecillo 10]. As they are both based on Meta-Models, their combination becomes a combination of Meta-Models (through e.g. Meta-Model extension, Meta-Model merge).

For the Design Rationale Domain Specific Modeling Language, the concrete syntax (cf. activity "Define display-surface MM" from Figure 5.1) and graphical editor (cf. activity "Generate graphical editor") are defined and respectively generated, just like for any other Domain Specific Modeling Language. Of course, the semantics of the Design

Rationale Domain Specific Modeling Language is implemented starting again from the Meta-Model describing the abstract syntax. Here is the most specific point in defining this language compared with the system ones. The interest in Design Rationale is not to execute it, like with the system models, but to do inferences and queries on it, so that modelers can retrieve it at a later date (cf. activity "Reason on DR" from Figure 5.1). Existing tools for Design Rationale retrieval can be integrated using the Off the Shelf component integration process (cf. Section 5.2.1).

Other Domain Specific Modeling Languages transversal on system Domain Specific Modeling Languages (e.g. for model versioning, for Software Product Lines of models) can be conceived using the same process. The central issue of the proposed process is the Meta-Model combination operation. While it could be automatized using model (vs. Meta-Model) combination approaches, Meta-Models describing language syntax (should) have a low number of components. Accordingly, an expert-driven approach to extending Meta-Models is both practicable and preferable to an automatic one, which has a higher level of inaccuracy.

## 5.4 Using Ontologies for Ensuring Semantic Interoperability



As discussed in Section 4.5, we ensure automatic syntactic and semantic interoperability between Domain Specific Modeling Languages. However, this is dependent of Domain Specific Modeling Language implementation choices. As we chose a Meta-modeling approach for language definition, the (abstract) syntax of Domain Specific Modeling Languages is described by a Meta-Model. Therefore, ensuring *syntactic interoperability* between two Domain Specific Modeling Languages means describing the common concepts of the two Meta-Models. Approaches to ensure syntactic interoperability between different Domain Specific Modeling Languages have been proposed, like combining them [Vallecillo 10]: extension, merge, embedding, weaving or hybrid approaches. However, the most flexible way to describe relations between two Meta-Models, is through Model Transformations. Using Model Transformations, one can describe the similarities between two Meta-Models and capture the intersection between the concepts of their respective Domain Specific Modeling Languages. Nevertheless, Meta-Models describe only the syntaxes of Domain Specific Modeling Languages. Accordingly Model Transformations, or other combination approaches between Meta-Models, can describe interoperability only at a syntactic level. We focus in what follows on semantic interoperability.

### 5.4.1 On the use of Ontologies with Meta-Models

The common thread in defining ontology [Welty 03] is that it is a *formal description* of a domain, intended for *sharing* among different applications, and expressed in a language that can be used for reasoning.

To date, to the best of our knowledge, there is no common agreement in the scientific community as to the relationship between Meta-Models and ontologies. While many agree that Meta-Models and ontologies share many and "deep" characteristics, numerous differences are highlighted as well, and some consider that Meta-Models and ontologies are complementary [Terrasse 06]. Mostly, ontologies have been used with Meta-Models for:

- *Model checking*: using automated reasoning techniques for validation of models in formalized languages. Consequently, undesirable properties may be discovered (e.g. redundancy) [Parreiras 07].
- *Model enrichment*: Kramler et al. [Kramler 06] consider that an ontology explicitly expresses the semantics of the modeling concepts whose syntax is defined by a Meta-Model.
- *Semi-automatic identification of mappings between Meta-Models*: discovers mappings between Meta-Models. Meta-Model matching may be a complex, tedious and error-prone work when executed manually. Therefore, approaches that generate Meta-Model mappings have been proposed. Kappel et al. [Kappel 06] propose a process which semi-automatically *lifts* Meta-Models into ontologies, *refactors* (to unfold typically hidden concepts in Meta-Models that should better be represented as explicit concepts in an ontology) and *enriches* them, and then applies *ontology matching* on them. However, unlike in our approach, they do not use the discovered matchings to *generate* Model Transformations. They implement the lifting step by specifying a weaving model from which they generate ATL code.

### 5.4.2 Ensuring Semantic Interoperability between Static Semantics of Modeling Languages



We propose to use ontologies for: describing the static semantics of Domain Specific Modeling Languages (i.e., model enrichment) and discovering a common reference ontology (i.e., semi-automatic identification of mappings between Meta-Models). A common ontology will ensure semantic interoperability and coherence between two adjacent role Domain Specific Modeling Languages. It can be discovered by determining the mapping between two ontologies, each describing the semantics of one Domain Specific Modeling Language. For this, we advance the following process, with four phases:

1. *Lift*. It transforms each Meta-Model into an ontology (cf. activity "Lift to $ONTO_i$" from Figure 5.1). We implement it through a Model Transformation between the Meta-Meta-Model describing the modeling technical space (e.g., Ecore [2]) and the Meta-Meta-Model describing the ontology space (e.g., OWL DL [3]). OWL DL is particularly suited for our process, as its definition is already given in the form of a Meta-Model.
2. *Enrich*. The lifted Meta-Models are enriched by applying patterns (cf. activity "Enrich $ONTO_i$" and "Enrich $ONTO_{i+1}$" from Figure 5.1). This way, one can find correspondences between the relationships of different Meta-Models. One can use patterns similar to that of "Association Class Introduction" [Kappel 06] can be used. A new class is introduced in the ontology similarly to an association class in UML, thus transforming relationships from Meta-Models into concepts in ontologies. We implement it through an endogenous Model Transformation, which has both as input and output the Meta-Meta-Model describing the ontology space.
3. *Align*. In the ontology technical space we apply ontology-specific techniques [Choi 06] (e.g., alignment) on the lifted and enriched Meta-Models of two adjacent roles, thus

2. `http://www.eclipse.org/modeling/emf` , accessed on 06.11.2011
3. `http://www.omg.org/spec/ODM/1.0/` , accessed on 06.11.2011

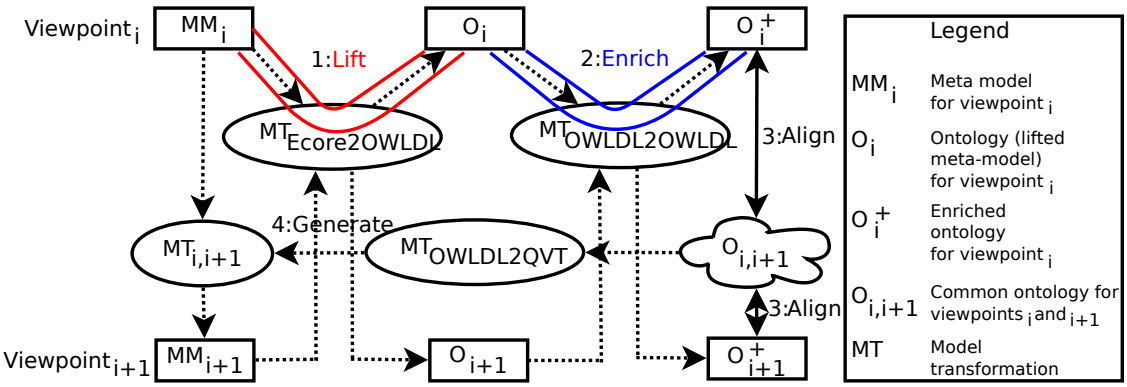


Figure 5.2 : Syntactic and semantic interoperability through Model Transformations and ontologies.

discovering their intersection (cf. activity "Align ontologies" from Figure 5.1). Because the lifted and enriched Meta-Models describe semantics of Domain Specific Modeling Languages, the discovered *shared ontologies* represent in fact the semantics of the Model Transformations between the original Meta-Models. We intend to rediscover these shared ontologies each time the (lifted and enriched) Meta-Models describing static semantics of Domain Specific Modeling Languages evolve. In this way, we ensure (static) semantic interoperability between two Domain Specific Modeling Languages.

4. *Generate*. Model Transformations which have as input and/or as output other Model Transformations are called Higher Order model Transformations (HOTs). We use shared ontologies as input for Higher Order model Transformations between the Meta-Meta-Model describing the ontology technical space and the Meta-Meta-Model describing the Model Transformation space (e.g., QVT [4]). These Higher Order model Transformations generate Model Transformations between the original Meta-Models (cf. activity "Generate $MT_{i,i+1}$" from Figure 5.1).

Consequently, we can automatically generate and evolve Model Transformations for a family of Domain Specific Modeling Languages, through their connections with shared ontologies, thus ensuring their syntactic and static semantic interoperability. The whole process can be automatized and consequently, it enables a high rate of reuse and faster iterations on evolving Meta-Models.

### 5.4.3 Example for two Adjacent Roles

Figure 5.2 exemplifies the proposed approach on two Meta-Models for adjacent roles (i.e., $MM_i$ and $MM_{i+1}$). Each Meta-Model describes a Domain Specific Modeling Language.

Each Meta-Model is *lifted* into an ontology (e.g., $O_i$ and $O_{i+1}$) by means of a Model Transformation (i.e., $MT_{Ecore2OWLDL}$). This Model Transformation is sufficient for lifting any Meta-Model (which conforms to the Meta-Meta-Model, in our case, Ecore) into an

4. `http://www.omg.org/spec/QVT/1.0/` , accessed on 06.11.2011

ontology, as it transforms concepts from Ecore, the language (Meta-Meta-Model) in which Meta-Models are written, into concepts from OWL DL, the language in which ontologies are written. To write this Model Transformation, we build on the mapping provided by [Kappel 06], updating it to the new versions of Ecore and OWL DL.

Patterns for refactoring, checking and *enriching* are applied on each lifted Meta-Model (e.g., $O_i$ and $O_{i+1}$) through a Model Transformation (i.e., $MT_{OWLDL2OWLDL}$). Similarly to lifting, this Model Transformation is sufficient for the enrichment of all lifted Meta-Models.

The enriched ontologies (e.g., $O_i^+$ and $O_{i+1}^+$) are *aligned*, resulting in a shared ontology (e.g., $O_{i,i+1}$).

With the shared ontology as input, $MT_{OWLDL2QVT}$ *generates* the Model Transformation between the initial Meta-Models (e.g., $MT_{i,i+1}$). Similarly to lifting and enrichment, this Model Transformation is sufficient for the generation of all Model Transformations between the initial Meta-Models.

The proposal presented in this section can also be found in our article [Chiprianov 11e].

## 5.5 Discussion

The tool building process proposed in this section (cf. Figure 5.1 in its totality) builds on processes for Domain Specific Modeling Language development, Off the Shelf component integration and Domain Specific Modeling Language combination approaches. The proposed Domain Specific Modeling Language development process has the following major advantages:

- clear separation between abstract syntax, concrete syntax, semantics;
- easy evolution of both language design and tools;
- easy composition of several languages (cf. Section 5.3);
- partial generation of language-associated tools from Meta-Models;
- semi-automatic interoperability between tools associated to different languages from the family (cf. Section 5.4);

but also disadvantages, which could limit its practical usability by Tool Vendors:

- even if done informally, domain analysis is long and difficult;
- the choice of concepts (abstract syntax) and of the representation of concepts (concrete syntax) requires several iterations and the direct involvement of domain experts and end-users.

The main advantage of the suggested process is the leverage of Off the Shelf components. This results in the massive re-use of existing formalisms and tools. The use of Model Transformations facilitates the building of the bridge between the Domain Specific Modeling Languages and the Off the Shelf tools and enables its model-based evolution. The difficulty of this process lies in constructing the Meta-Model describing the data used by the Off the Shelf tool. This is especially laborious when dealing with commercial or poorly-documented tools which do not provide many details about the format of the data files they expect as input.

This process reuses the Domain Specific Modeling Language definition process, may reuse model combination approaches transfered to Meta-Models, and may reuse the Off

the Shelf tool integration process. The Meta-Model combination effort is the only overhead activity for integrating the Design Rationale Domain Specific Modeling Language with a system Domain Specific Modeling Language. No extra effort is needed to integrate the concrete syntaxes or the semantics of the two languages.

Using a meta-modeling approach combined with ontologies has the advantage of co-evolving syntactic and semantic bridges that ensure interoperability between Domain Specific Modeling Languages. However, this co-evolution depends greatly on the shared ontology between viewpoints. If this were poor (small number of entities) or even empty, the interoperability bridge would be narrow. Consequently, in order for the proposed approach to be effective one should first make sure that the vocabularies for different viewpoints have a fair amount of concepts in common. This supports the idea that such an approach would be beneficial especially in the case of families of Modeling Languages.

The tool building process proposed in this section adjusts these general processes to the practical needs of telecommunications Tool Vendors. It also combines them into one overall process, showing the integration of the Domain Specific Modeling Language definition process into a larger vendor process which tackles not only modeling tools, but also testing, collaboration and interoperability. As part of this overall process, a new sub-process based on Model Transformations, Higher Order model Transformations and ontologies is proposed to ensure Domain Specific Modeling Language interoperability.

Noteworthy is the fact that the Meta-Model describing the abstract syntax of a Domain Specific Modeling Language is an input to all sub-process (cf. Figure 5.1). Its importance is therefore central, crucial, to the entire process. As it is the result of the *Domain analysis* phase, the importance of this phase becomes evident. Formal methods of carrying out this phase could be very helpful in evolving the Meta-Model.



We presented in this chapter our proposal for a process describing how to build the tools used by stakeholders in the telecommunications service creation process. It is of no consequence whether the telecommunications service creation process is waterfall-like or iterative, the tool building process we proposed is adapted to both. It is based on models and it offers a high automation degree, hence it allows Tool Vendors to build software tools more rapidly. It is also worth mentioning that the same process can be used to build the tools for all and any of the stakeholders. In fact, this is a process which defines a family of Domain Specific Modeling Languages and associated tools. Now we are ready to instantiate this tool building process for telecommunications.

# 6 Domain Specific Modeling Languages and Software Tools for Telecommunications Service Construction

”Huwawa spoke to Bilgames:
’0 deceitful warrior, ...’ ” 152 UrG

Bilgames and Huwawa: ’The lord to the Living One’s Mountain’, in
*The Epic of Gilgamesh. A new translation*, Andrew George, Penguin Books, 2000

For the four main activities of the telecommunications service creation process proposed in Chapter 4, we built software tools following the process proposed in Chapter 5. These tools are our answer to the *RQ 2 Software tools*. To enable the *MA 1 Model*, we defined Domain Specific Modeling Languages for two stakeholders, the Service Provider and the Service Developer. Domain Specific Modeling Language development consists in defining the abstract and concrete syntaxes, and the semantics (cf. Chapter 3). To reuse existing Modeling Language definitions and tools, we designed these Domain Specific Modeling Languages as profiles/extensions of an Enterprise Architecture Modeling Language selected previously, in Chapter 2, ArchiMate. To enable the *MA 2 Test*, we generated input models for an existing network simulation tool. To enable the *MA 3 Collaborate*, we defined a Domain Specific Modeling Language for capturing Design Rationale. The *MA 4 Inter-operate* is mainly automatic, based on relations defined between ArchiMate layers. We present the integration of these Domain Specific Modeling Languages and Off the Shelf components in the Service Creation Environment.

## 6.1 An Enterprise Architecture Modeling Language Extension for Telecommunications Service Construction

In this section we present the tools proposed for the *MA 1 Model*. We decided to reuse by extension an Enterprise Architecture Modeling Language, particularly in order to answer *Req 1 An overall model*. As discussed in Chapter 2, Enterprise Architecture strives to achieve such an overall representation. If Telecommunications Domain Specific Modeling Languages are designed as extensions/profiles (e.g. [Bernardi 11]) of an Enterprise Architecture Modeling Language, this gives them the potential of the latter, that is, the capability of describing overall models. The chosen Enterprise Architecture Modeling Language is ArchiMate (cf. Chapter 2). As ArchiMate has three layers, we propose extending each of them with concepts specific to Telecommunications. These three extensions cover

the concerns of Service Providers and Service Developers. They represent Telecommunications specific Domain Specific Modeling Languages. For these, we decided to reuse an Enterprise Architecture Modeling Language in the design phase.

To define the Telecom Domain Specific Modeling Languages, we followed the development approach proposed in Section 5.1.2. To define the Meta-Models (cf. *Define* $MM_i$, Figure 5.1), we used discussions with telecommunications experts, Meta-Models from literature (e.g. proposed by Orange Labs experts [Bertin 09a]) and modeling of service platforms (e.g. IMS [3GPP 10]) to build initial Telecommunications Meta-Models. The initial Telecommunications Meta-Models were used to extend the abstract syntax of the three layers of ArchiMate. In the next part, we present the Meta-Model extension approach used for this purpose.

### 6.1.1 Practical Meta-Model Extension for Modeling Language Profiles

Model extension is defined by [Barbero 07] as the operation that produces the result model - $M_r$, by applying a mapping $\epsilon$ between two models: $M_i$ - initial, and $M_f$ - fragment. The mapping $\epsilon$ specifies that, if a node from $M_f$ is equivalent (=) with a node from $M_i$, the node from $M_i$ will be copied in $M_r$. If a node from $M_f$ is not equivalent with any node from $M_i$, a new node will be created in $M_r$, not necessarily the initial node from $M_f$. Defined in this way, model extension is a particular case of model combination (also called model composition or weaving) [Vallecillo 10].

Model composition has been widely addressed. For example, [France 07] proposes a generic framework for model composition that is independent from a modeling language. The framework is based on an algorithm structured in two steps:

1. *Matching*: identifies model elements that describe the same concepts in the different models that have to be composed;
2. *Merging*: matched model elements are merged to create new model elements that represent an integrated view of the concepts.

Model merging is kept flexible through pre and post-merge directives. This composition approach is intended for the automatic composition of models belonging to separate aspect-oriented views, to build a global view of a system.

[Barbero 07] notes that the node comparison operators = and $\neq$ must be defined in a MM specific way (e.g. if the Meta-Model has classes, class names can be used). This is part of the more general issue of model matching [Kolovos 09], defined as a theoretically NP-hard graph isomorphism problem. Approaches to tackle it automatically, usually consider models as typed attribute graphs and calculate the similarity of nodes based on the combined weighted similarity of their features. However, these methods are intended for large models. Unlike models, Meta-Models are relatively small and strongly influence all models that are conformant with them. This is why a precise, Meta-Model-dedicated approach for comparing and extending them (describing the $\epsilon$ mapping) is both possible and preferable to an automatic, general one.

The notations introduced by [Barbero 07] take into account only nodes for comparison. To obtain better matching, edges must be considered as well, and even patterns - formations of multiple nodes and edges. Likewise, having only two values (equal, not equal) for equivalence is not enough. A node (edge) from $M_f$ may be equivalent with several nodes (edges) from $M_i$, but with different degrees. Consequently, we propose using the concept


Figure 6.1 : Illustration of the *Prin. 2 Transitivity of similarity for edges (nodes)* ($E_{f1} \sim E_{f2} \sim E_i$).

of *similarity* $\sim$ to replace the dichotomic equivalence $=/\neq$. Based on their semantics (definitions/ontologies), the matching of any two nodes (edges) from the two models - $M_i$ and $M_f$ - can be calculated using a similarity measure (continuous - e.g. $\in [0, 1]$), chosen by an expert. If a node (edge) from $M_i$ has a similarity degree, greater than a certain threshold (established by an expert), with another node (edge) from $M_f$, the two can be considered similar ($\sim$). Now we introduce the principles (Prin.) (without any claim on completion) that define our approach for Meta-Model extension:

1. *Prin. 1 Generalization of similarity of nodes.* If a node from $M_f$ is similar with a node from $M_i$, which is derived from another node (generally, it is part of an inheritance hierarchy, at some level), the node from $M_f$ is similar with the parent node of the hierarchy the node from $M_i$ is part of. Therefore, the parent node from $M_i$ is copied to $M_r$.
2. *Prin. 2 Transitivity of similarity for edges (nodes).* Let us consider a node (edge) (e.g. $N_{f1}$ in Figure 6.1) from $M_f$, that is connected with two edges (nodes) (e.g. $E_{f1}$ and $E_{f2}$). Let both edges (nodes) have a (high and) close degree of similarity ($\sim$) with an edge (node) from $M_i$ (e.g. $E_i$). Both edges (nodes) from $M_f$ are similar with the edge (node) from $M_i$. Only this edge (node) (e.g. $E_i$) from $M_i$ is copied to $M_r$.
3. *Prin. 3 Pattern matching.* Let us consider several nodes from two models that are similar. If they form a connected graph in $M_f$, the edges connecting the nodes in $M_f$ have to be similar with the edges connecting the nodes in $M_i$.

We now define the concept of *dissimilarity* $\not\sim$. Two individual nodes (edges) are dissimilar if their similarity degree is under the established threshold. Dissimilarity can also occur when the *Prin. 3 Pattern matching* conditions are not fulfilled.

*Priority of principles.* The *Prin. 1 Generalization of similarity of nodes* should be applied before the *Prin. 2 Transitivity of similarity for edges (nodes)*, as it provides looser node semantics and more possibilities of matching, albeit with a smaller similarity degree. The *Prin. 3 Pattern matching* is the most difficult to fulfill (it involves multiple nodes and edges), hence it should be applied last. Semantics of groups (global semantics) is more important than semantics of individual nodes or edges. Therefore (minor) changes may be beneficial in the overall similarity degree of the pattern. Changing the order of principle application may change the obtained results.

The approach was presented in [Chiprianov 11c] as well.

### 6.1.2 The Abstract Syntax

To obtain the Meta-Models for the ArchiMate Telecommunications extension, the practical Meta-Model extension approach (cf. previous subsection) was employed. We exem-

plify it here by applying the proposed three principles to extend the Meta-Model of the ArchiMate [The Open Group 09a] Business layer - $M_i$ (cf. Figure 2.3, or Figure 6.2 without the nodes marked with a red **T** and the edges that connect them), with the telecommunications specific Meta-Model - $M_f$, Figure 6.3, proposed by [Bertin 09a]:

1. We start by determining the similarity degree of nodes; *Stakeholder*, *Action*, *ServiceProcess* from Figure 6.3 can be considered similar respectively with *BusinessRole*, *BusinessFunction*, *BusinessService* from Figure 6.2. Moreover, through the *Prin. 1 Generalization of similarity of nodes* between *BusinessFunction* and *BusinessBehaviourElement* from Figure 6.2, *Action* is similar to *BusinessBehaviourElement*. The *is done by* edge between *Action* and *Stakeholder* can be considered similar with the edge *assigned to* between *BusinessBehaviourElement* and *BusinessRole* from Figure 6.2.
2. We have seen that *ServiceProcess* is most similar with *BusinessService*. We apply the *Prin. 3 Pattern matching*. However, the edges that connect it must be taken into account. The only edge is *aggregation* with *Action*. An edge similar with it must be found in the ArchiMate Meta-Model. If none is found, a new one will be introduced. The candidate aggregation edges from the ArchiMate Meta-Model, that relate to *BusinessBehaviourElement* are: the one with *Product* through the *realizes* edge with *Business Service*, and the one with *BusinessCollaboration* through the *assigned to* edge. The *BusinessColaboration* node has a higher similarity degree with *ServiceProcess* than that of *Product*. Consequently, through the *Prin. 3 Pattern matching* and the priority of principles, *ServiceProcess* can be considered similar with *BusinessCollaboration*. However, due to the rather low similarity value, we have decided to introduce *ServiceProcess* as a derived node (Figure 6.2).
3. *ServiceEntityState* is most similar with *BusinessObject*, but even so, their similarity is quite limited. As the *produces* and *uses* edges between *Action* and *ServiceEntityState* from Figure 6.3 have a strong similarity degree with the *accessed by* edge between *BusinessBehaviorElement* and *BusinessObject* from Figure 6.2, we apply the *Prin. 3 Pattern matching* and introduce *ServiceEntityState* as derived from *BusinessObject*. *ServiceEntity* has a low similarity degree with all other nodes from the ArchiMate MM, so it is introduced as a new node, with edges *refers to* and *in relation with*. The result is $M_f$ (Figure 6.2 with a red **T**).

More details can be found in [Chiprianov 11c].

Following the same practical Meta-Model extension approach, the Meta-Models for the Telecommunications extension of ArchiMate Application and Technology layers are obtained. Three new concepts are added to the initial (cf. Figure 2.4, or Figure 6.4 without the nodes marked with a red **T** and the edges that connect them) ArchiMate Application Meta-Model: *ServiceFunctionalOperation*, *ServiceFunctionalComponent* and *ServiceElementaryFunction* (cf. Figure 6.4, nodes marked with a red **T**). These concepts have been found to be similar to the ones that they inherit, but also sufficiently dissimilar (i.e. specific to the Telecommunications domain) to be introduced as new concepts. More details can be found in [Chiprianov 11a].

At the ArchiMate Technology layer, the concepts of the initial (cf. Figure 2.5, or Figure 6.5 without the nodes marked with a red **T** and the edges that connect them) ArchiMate Meta-Model have been derived using inheritance by telecommunications IP Multimedia Subsystem (cf. Section 1.4.1) specific concepts: 1) Structural: *Proxy-*

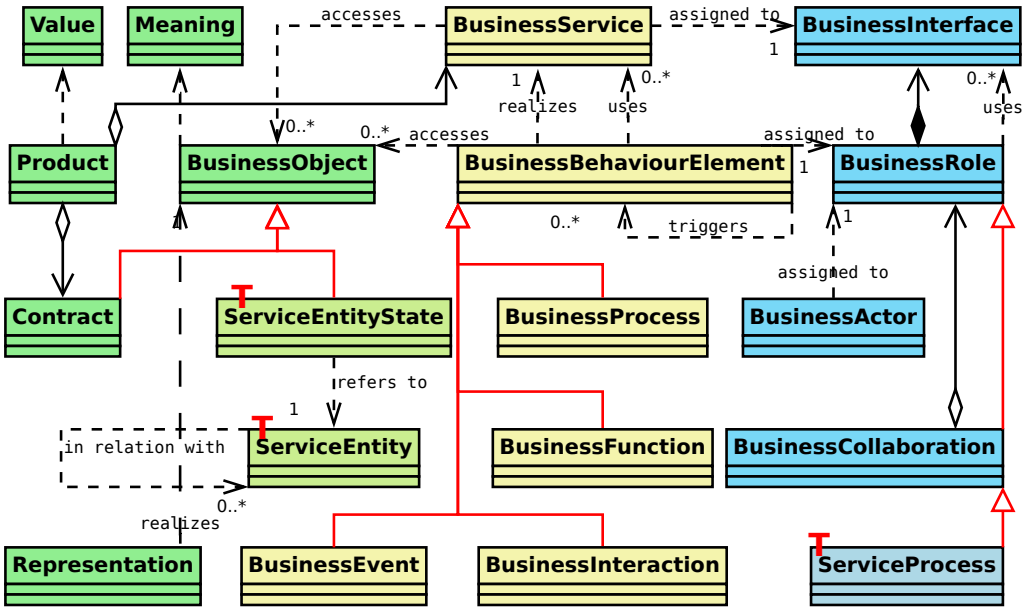


Figure 6.2 : The Telecommunications ArchiMate Business layer Meta-Model extension. More details in [Chiprianov 11c].


Figure 6.3 : Meta-model of the telecommunications reference business view, after [Bertin 09a].

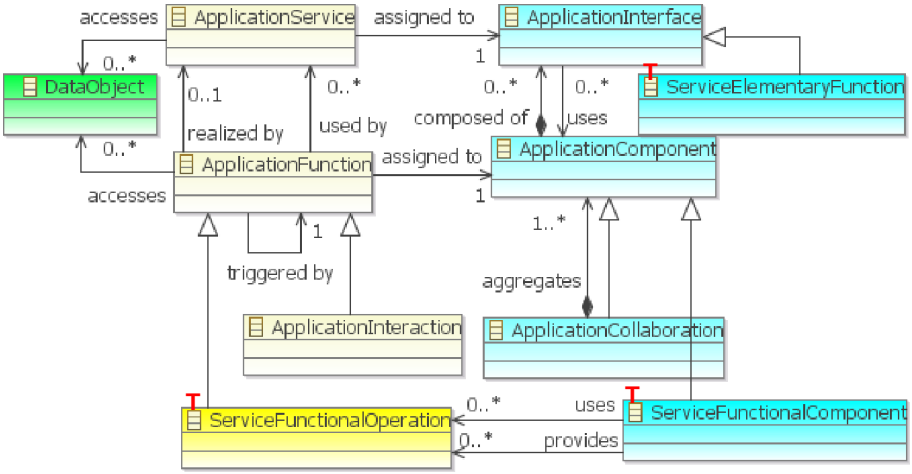


Figure 6.4 : The Telecommunications ArchiMate Application layer Meta-Model extension. More details in [Chiprianov 11a].

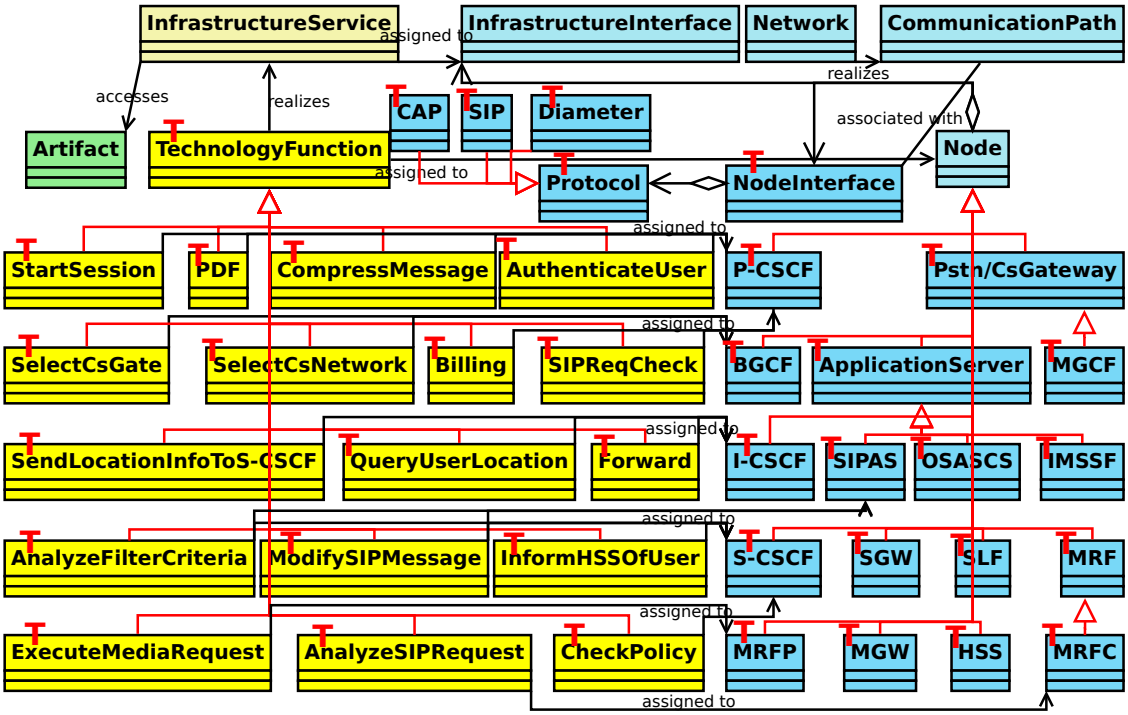


Figure 6.5 : The Telecommunications ArchiMate Technology layer Meta-Model extension. More details in [Chiprianov 11d].

*Call/Session Control Function (P-CSCF)*, *PSTN/CS Gateway*, *Media Gateway Control Function (MGCF)*, *Breakout Gateway Control Function (BGCF)*, *ApplicationServer*, *Session Initiation Protocol Application Server (SIPAS)*, *Open Service Access Service Capability Server (OSASCS)*, *IP Multimedia Service Switching Function (IMSSF)*, *Interrogating-CSCF (I-CSCF)*, *Serving-CSCF (S-CSCF)*, *Signaling Gateway (SGW)*, *Subscription Locator Function (SLF)*, *Media Resource Function (MRF)*, *Media Resource Function Controller (MRFC)*, *Home Subscriber Server (HSS)*, *Media Gateway (MGW)*, *Media Resource Function Processor (MRFP)*, which are derived from the ArchiMate *Node* concept, *NodeInterface*, *Protocol*, *Session Initiation Protocol (SIP)*, *Diameter*, *Configuration Access Protocol*; 2) Behavioral: *StartSession*, *Policy Decision Function (PDF)*, *CompressMessage*, *AuthenticateUser*, *SIP Request Check (SipReqCheck)*, *Billing*, *SelectCsNetwork*, *SelectCsGate*, *SendLocationInfoToS-CSCF*, *QueryUserLocation*, *Forward*, *InformHssOfUser*, *ModifySIPMessage*, *AnalyzeFilterCriteria*, *CheckPolicy*, *AnalyzeSIPRequest*, *ExecuteMediaRequest*, which are derived from the *TechnologyFunction* concept. More details can be found in [Chiprianov 11d].

The entities specific to Telecom are marked in all figures with a red T. The number of Telecommunications-specific entities introduced in the ArchiMate Meta-Models grows as the level of abstraction diminishes (from three at the Business and Application layer, to forty at the Technology layer). These entities comprise, for Service Provider, at the Telecommunications ArchiMate Business layer, the concepts: 'ServiceEntityState', 'ServiceProcess' and 'ServiceEntity' (cf. Figure 6.2); for Service Developer, at the Telecommunications ArchiMate Technology layer, the concepts from IMS (cf. Figure 6.5). Of course, as

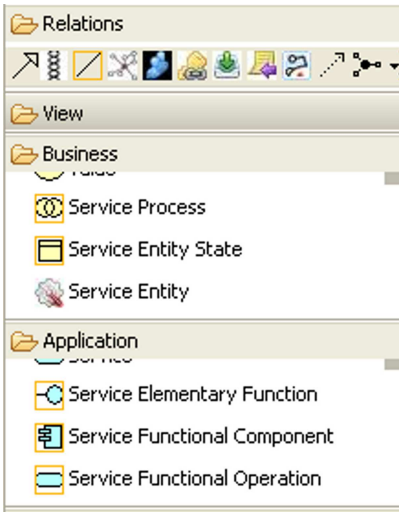


Figure 6.6 : The Telecom ArchiMate Business and Application layers extension concrete syntax.

even the definition of roles may be somehow fuzzy, and as in industry one enterprise may fulfill several roles, any separation between roles and the concepts that are specific to each of them is going to be somehow artificial and sure to evolve in time. However, just the fact that there are different roles implies that different types of actions involving different concepts are being taken. Moreover, there is a (large) number of concepts that is shared between the roles (e.g. the concepts at the Telecommunications ArchiMate Application layer: 'ServiceFunctionalOperation', 'ServiceElementaryFunction' and 'ServiceFunctional-Component', cf. Figure 6.4 and Figure 6.14 for this sharing).



### 6.1.3 The Concrete Syntax

The next activity in developing the Telecommunications Domain Specific Modeling Languages is defining the concrete syntaxes (cf. *Define "display-surface" MM*, Figure 5.1). For each concept and relation introduced in the Meta-Models (i.e. abstract syntaxes), a graphical representation was defined. In their definition, closeness to concepts from ArchiMate was emphasized (cf. Section 2.5.2), which means that concepts that inherited those from ArchiMate have inherited their representation as well, with a mark of distinction (e.g. an orange border, cf. 'ServiceElementaryFunction', Figure 6.6).

The representations of the added concepts and relations for the Business and Application layers are presented in Figure 6.6 (cf. Figure 6.2 and respectively Figure 6.4 for the Business and Application Meta-Models).

The representations of the added concepts for the Technology layer are presented in Figure 6.7 (cf. Figure 6.5 for the Technology Meta-Model).

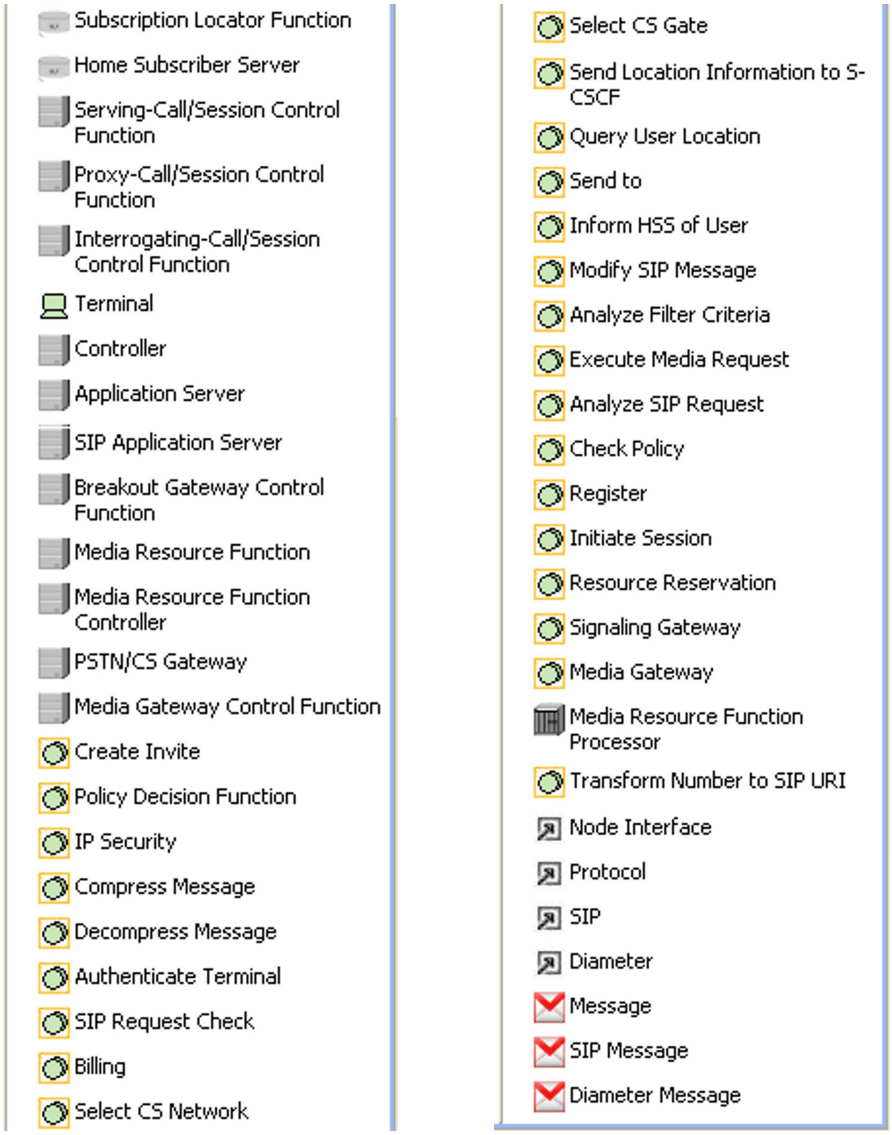


Figure 6.7 : The Telecom ArchiMate Technology layer extension concrete syntax.

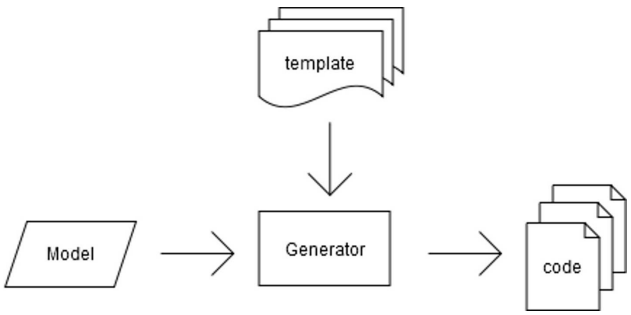


Figure 6.8 : Principle of template-based code generation, after `http://blog.henkvandijken.nl/`.

### 6.1.4 The Semantics

To implement the semantics of the Telecommunications extension (cf. *Define semantics* activity, Figure 5.1), we chose to use template-based code generation (cf. *Generate code template-based* activity, Figure 5.1) (or model-to-text transformation - cf. Section 3.2). Our choice is due to the maturity of this approach, and the availability of powerful, mature, free tools. In this approach, the transformation of the input language is captured in rules called templates. Templates define the translation/transformation rules between two languages. Transformation engines (i.e. generators) take them as input (cf. Figure 6.8), together with models defined in the input Modeling Language, and produce code in the output (programming) language.

We defined the templates using the Xpand [Efftinge 06] template description language, and its editor and compiler, the free Eclipse plug-in OpenArchitectureWare. The static semantics was implemented (cf. Listing 6.1), resulting in Java classes, with attributes, method signatures. For the behavioral semantics, method call was implemented, resulting in the possibility of translating sequence-like diagrams into code.

The rules transform, for example, for the Application layer Meta-Model (cf. Figure 6.4):

1. For static semantics:
   (a) ApplicationInterface and ServiceElementaryFunction into Java *interfaces*;
   (b) ApplicationService into Java *abstract method*;
   (c) ApplicationComponent and ServiceFunctionalComponent into Java *classes*;
   (d) The *composition* relation (between ApplicationInterface and ApplicationComponent) into the Java *implements* relation (between an interface and a class);
   (e) The *aggregation* relation (between ApplicationColaboration and ApplicationComponent) into Java *attributes*;
   (f) ApplicationFunction into a Java *method*;
   (g) DataObject into Java *parameter* for method signature.
2. For behavioral semantics:
   (a) The *triggered by* relation (between ApplicationFunction and itself) into Java *method call*.

The generated skeletons can be further detailed/implemented using Java libraries that abstract Telecommunications-specific protocols, like JAIN [Ferry 08]. Obviously, the code generation encompasses not only the Telecommunications extension, but also the initial

```
1  <<DEFINE javaClass (String FolderName, String PackageName) FOR model::
       ApplicationComponent>>

3  <<FILE FolderName+"\\"+PackageName+"\\"+this.name.replaceAll("[:]","_").replaceAll
       ("[ ]+","")+".java">>
   <<EXPAND AddJavaHeader(this,name,FolderName,PackageName) FOR Void->>
5    public class <<this.name.replaceAll("[:]","_").replaceAll("[ ]+","")->>
       <<EXPAND CheckApplicationAggregation(this,name) FOR Void->>
7      <<EXPAND CheckSpecialisation(this,name) FOR Void->>
       <<EXPAND CheckRealisation(this,name) FOR Void->>
9      {
         <<FOREACH this.properties AS prop>>
11         <<prop.metaType.name>> <<prop.key>> = <<prop.value>>;
         <<ENDFOREACH>>
13     ...
       }
15 <<ENDDEFINE>>
```

Listing 6.1 : Excerpt of Xpand template for Java class generation. Expressions between « » are Xpand instructions. The most common one, «EXPAND ...», is equivalent with a method call, it expands a rule previously defined. Text which is not in « » is just copied to the output by the generator; it usually corresponds to keywords from the output language.

ArchiMate language.

6

### 6.1.5 Language-specific Tools

Language-specific tools are usually the editor (for Modeling Languages the editor is graphical) and the compiler describing the operational semantics of the language.To implement language-specific tools (cf. *Generate graphical editor*, Figure 5.1), we use Meta Tools (cf. Section 3.3).

Meta Tools targeted at the generation of graphical editors used in this thesis are the Eclipse Graphical Editing Framework (GEF) and the Eclipse Modeling Framework (EMF) (cf. Section 3.5). EMF is used to generate a part of the graphical editor from a Meta-Model. The generated stubs can be further customized and expended using GEF. An existing open source ArchiMate graphical editor, Archi [1], developed as an Eclipse plug-in, has been selected and reused, expanded. Indeed, the plug-in architecture is particularly suited to tools for Domain Specific Modeling Languages designed as extensions to other languages,

#### 6.1.5.1 Developing Domain Specific Modeling Language Tools as Plug-ins

A software plug-in is a set of software components that adds specific capabilities to a larger software application [2]. As an auxiliary "client" module or expansion, it allows the adding of specific capabilities to a larger "host" software application. For example, external capabilities may be functions, services, features, or support for handling a file format. The plug-in pattern, Figure 6.9, from [Mayer 03], shows how to design an application in order

1. http://archi.cetis.ac.uk/ , accessed on 06.11.2011
2. http://en.wikipedia.org/wiki/Plug-in_%28computing%29 , accessed on 06.11.2011

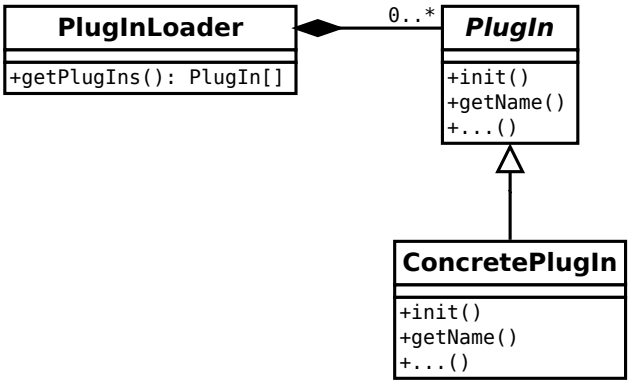


Figure 6.9 : UML class diagram for the plug-in pattern, from [Mayer 03].

to allow its extension at runtime by dynamically loaded modules or classes. The plug-in loader is part of what is called the framework.

Well-known examples of systems based on plug-ins include web-browsers (e.g., the add-ons [3] for Firefox), graphics editing programs [4], games (plug-ins are called mods [5]), integrated development environments (IDE) (e.g., Eclipse), tools for formal analysis and verification.

Plug-in systems are developed in order to benefit from the following advantages [Wagner 07]:

- Stability of system design. New features are added through plug-ins, independent of the core functionalities of the application.
- Reduced frequency of context switches. The user remains in the same integrated environment, experiencing a feeling of continuity.
- Increased usability. The user does not need to learn to use a new environment for the system functionality.
- Re-usability of framework functionality. Basic shared functionality is provided by the framework, thus liberating the plug-ins from assuring it, reducing complexity and increasing modularity and understandability [Mayer 03].
- High flexibility in tool customization. The user can select the specific plug-ins tailored to his/her needs.
- Interoperability. In many research communities, all tools are developed using the same framework.
- Easy extensibility [Paulisch 93]. New tools can be added without the need to understand the framework code. Extendibility [JTC1/SC7/WG6 09] is defined as the degree of usability and safety in contexts beyond those initially intended. Extendibility includes, but is not restricted to, extensibility.

However, plug-ins present drawbacks as well:

- Difficult installation of new plug-ins. Compatibility issues with already present plug-ins, versioning problems, impede users and may even provoke reliability problems if

3. `https://addons.mozilla.org/en-US/firefox/` , accessed on 06.11.2011
4. `http://thepluginsite.com/knowhow/tutorials/introduction/introduction.htm` , accessed on 06.11.2011
5. `http://www.civfanatics.com/civ4/downloads` , accessed on 06.11.2011

the existing application stops.
- Restriction to the chosen framework. The framework may not adequately support all the necessary functionality and/or technologies (e.g., limitation to only a certain operating system).

Plug-ins are conceived as extensions to "host" applications. However, certain categories of plug-ins may benefit from or even require to be extended as well. One such category are tools for Domain Specific Languages designed as language extensions (e.g. profiles). The implementation of a Domain Specific Language and its associated tools has as starting point an existing, more general purpose, base language and its tools. Developing tools (e.g., editors, code generators) for such Domain Specific Languages benefits from reusing base language tools. The base language tools are often part of a tool-set, an IDE, and are implemented as plug-ins. Therefore, developing tools for Domain Specific Languages based on another language often consists in extending plug-ins.

Plug-in extension may be considered as a subproblem of the customizing libraries issue. Library customization consists in adding new or modifying existing pieces of code. A recent comparative study [Aktemur 09] surveyed most of the techniques for library customization. Despite their differences, all techniques require some sort of hole/hook point/expansion in the "host" code. The extensions have to "hook" into a main application or framework environment [Rubel 06]. For this, "client" extensions must declare how they interact with the "host" and the "host" must provide interaction points.

We have presented a more detailed discussion of plug-in extensibility in [Chiprianov 11b].



#### 6.1.5.2 The Graphical Editor

An existing open source ArchiMate graphical editor, Archi, developed as an Eclipse plug-in, has been selected and reused, expanded with the Java code needed to describe the concrete syntax of the Telecommunications extension. Archi is a free, open source, cross-platform editor which helps create ArchiMate models. Archi is developed as a plug-in for Eclipse 3.6.1. The editor presents the classical divisions of an Eclipse-based editor (cf. Figure 6.10). At the left, there is the model navigator and an outline of the graphical model. The central window presents views (defined as tabs) of the graphical model. At the right, the palette was extended with telecommunications-specific concepts and relations (cf. Figure 6.6 and Figure 6.7), from which the designer can select, drag and drop the desired ones.

#### 6.1.5.3 The Compiler

OpenArchitectureWare (OAW), a Meta Tool, was used for compiler implementation. If software compilers, that translate from a high level of abstraction language into machine language, are considered to be tools, then any other means of implementing language translations can be considered to be a tool as well. Consequently, model transformations may be seen as tools (in [Czarnecki 98], Section 6.3, transformation systems are said to be often referred to as open compilers). It follows also that language translation implemented as template-based code generation (a model-to-text transformation) may be seen as a tool. Hence, the templates used for translating the Telecommunications extension of ArchiMate

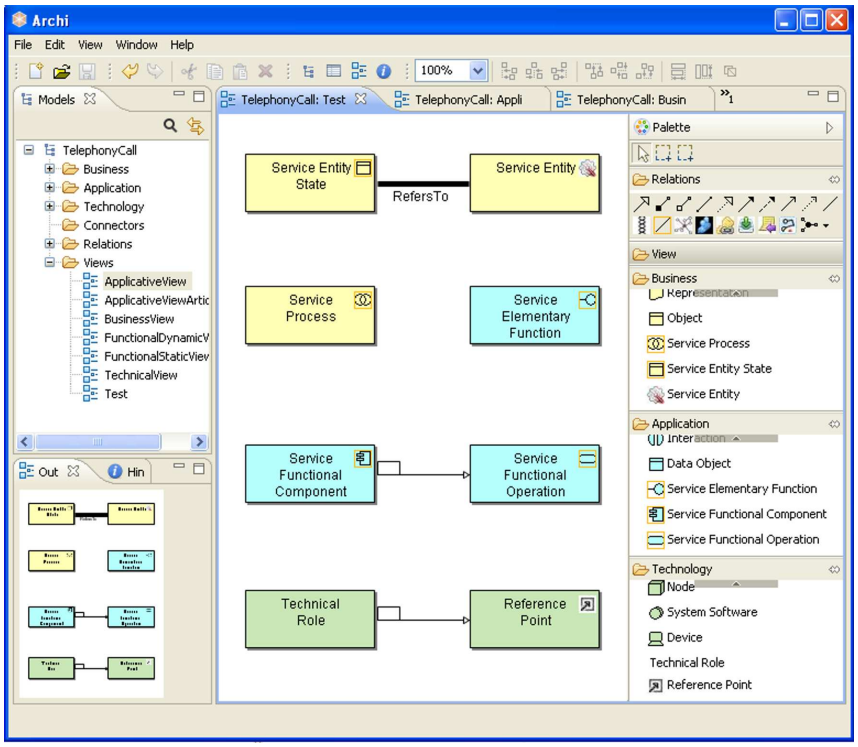


Figure 6.10 : The Archi editor with the Telecommunications extension.

into Java (e.g. Listing 6.1), written in Xpand and ran with OpenArchitectureWare, describe the configuration of a compiler.

An overview of the language tools is presented in Section 6.5.

## 6.2 Leverage of Network Simulators for Testing Telecommunications Service Models

In this section we present the tools proposed for the *MA 2 Test*.

### 6.2.1 Telecommunications Service Simulation

Ten simulators are compared by [Sarkar 11], which finds that the most frequently used network simulators in scientific articles are NS-2 (the latest version is NS-3 [6]) and OPNET [7]. While NS-2 is open source, OPNET is commercial. However, the complete set of OPNET modules provides more features than NS-2 [Phillips 07]. Optimized Network Engineering Tool (OPNET) Modeler® is a discrete event, object-oriented, general purpose network simulator. It provides a comprehensive development environment for the specification, simulation and performance analysis of computer and data communication networks. It includes an extensive, well-document library of protocols, a powerful state-machine based representation of processes, a comprehensive model library. It offers modular model

6. http://www.nsnam.org/ , accessed on 06.11.2011
7. http://www.opnet.com/ , accessed on 06.11.2011

```
1  <<DEFINE GenerateTopologyXMLOPNET (ArchimateModel type, String FolderName, String
       PackageName, List LinkedToRouter, List DirectLinks, List Ports, List Ports_Router)
       FOR model::Node->>
   ...
3  <<IF model::PCSCF.isInstance(this.metaType.newInstance()) || model::SCSCF.
       isInstance(this.metaType.newInstance()) || model::ICSCF.isInstance(this.
       metaType.newInstance())->>
     <<DestPrefApp.add("Video Conference").toString().replaceAll(".*","")->>
5    <<EXPAND Tech_NodeXMLOPNET(L, this.name,"Linux_Server","dell_pe8450","Linux Server
         ","14:08:46 Mar 17 2011",DestPrefApp, DestPrefSymb, DestPrefAct, SelectedWeights
         , SupProfiles) FOR Void->>
   <<ENDIF->>
7  ...
   <<ENDDEFINE>>
```

Listing 6.2 : Excerpt of Xpand template for OPNET input model generation. Expressions between « » are Xpand instructions. The most common one, «EXPAND ...», is equivalent with a method call, it expands a rule previously defined.

development, a high level of modeling detail, user-friendly GUI, and customizable presentation of simulation results. It can simulate a model's behavior at different levels of detail. Packet exchange can be visualized, errors in the signaling logic can be detected. Systems are represented hierarchically. OPNET supports teaching and research under the OPNET university academic program.

As such powerful software tools already exist for telecommunications service simulation, we leverage them (cf. Section 5.2.2) by ensuring the interoperability between the Domain Specific Modeling Languages used for the *MA 1 Model* and a network simulator. We chose OPNET, taking advantage of its university academic program.

### 6.2.2 Leverage of Network Simulators

To leverage the OPNET network simulator (cf. *Transform to* $MM_{testing}$, Figure 5.1), we used a Model Transformation. This Model Transformation is written between the integrated Meta-Model of the Telecommunications ArchiMate extension (cf. Figure 6.2, Figure 6.4, Figure 6.5) and the Meta-Model deduced from the XML structure of the OPNET input model file. To implement this model-to-model transformation, Xpand was used. For example, Listing 6.2 presents an excerpt from the translation of the (line 3) *P-CSCF*, *S-CSCF* and *I-CSCF* concepts (cf. Figure 6.5) into a *Linux_Server* OPNET node (line 5).

The simulation results are interpreted by a specialist, who makes recommendations on changing the input model (cf. *Integrate results into* $MM_i$, Figure 5.1).

## 6.3 An Enterprise Architecture Modeling Language Extension for Collaboration

In this section we present the tools proposed for the *MA 3 Collaborate*. To define the Design Rationale Domain Specific Modeling Language, we followed the tool building process we proposed in Chapter 5 (cf. Figure 5.1. We began by defining the Meta-Model, the abstract syntax.

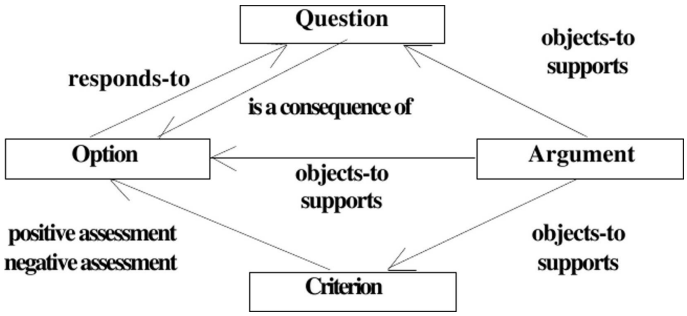


Figure 6.11 : The QOC schema, from [Dutoit 06], after [MacLean 96].

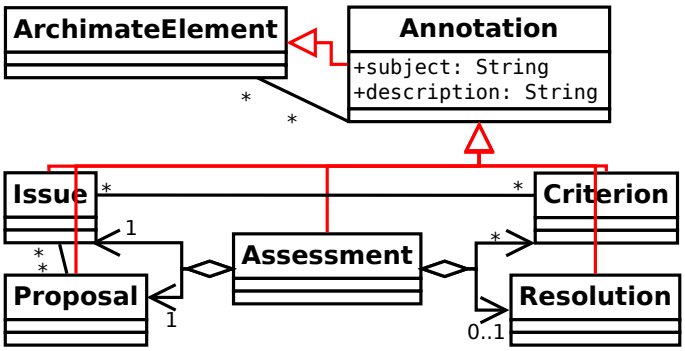


Figure 6.12 : The abstract syntax of the Design Rationale Domain Specific Modeling Language.

### 6.3.1 The Abstract Syntax

As the domain to be analyzed is that of Design Rationale, the information that characterizes it has already been captured for Design Rationale formalization, for example by schemas (e.g. IBIS, QOC, DRL). The QOC [MacLean 96] schema (cf. Figure 6.11) was chosen, due to previous successful use for software engineering [Wolf 08]. It is noteworthy that if any other Design Rationale schema were chosen, the remainder of the approach would apply in the same manner.

The Design Rationale Domain Specific Modeling Language abstract syntax (cf. *Define* $MM_{collab}$, Figure 5.1) was defined as a Meta-Model inspired from the QOC schema. The Meta-Model is presented in Figure 6.12. During the decision-making process, designers have to asses several proposals for solving an issue, comparing them against several criteria and finally choosing one resolution. The Meta-Model captures this by modeling an *Assessment* as containing one *Proposal* to one *Issue* (but, in general, one *Proposal* can be attached to several *Issues*, and one *Issue* usually has several *Proposals* - cf. the association between them). This proposal is assessed against one *Criterion* or more (which differ and depend on the type of *Issue* - the same *Criterion* can be attached to several *Issues*, and one *Issue* may have one *Criterion* or more - cf. the association between them). The *Proposal* could be chosen as *Resolution* or not.

To connect the Design Rationale concepts with the Enterprise Architecture Modeling Language (ArchiMate) concepts (cf. *Combine* $MM_{collab}$ *with* $MM_i$, Figure 5.1), the

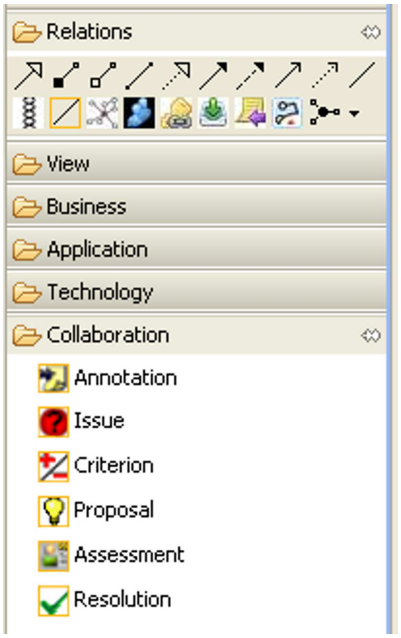


Figure 6.13 : The concrete syntax of the Design Rationale Domain Specific Modeling Language.



*Annotation* is introduced as inheriting from *ArchimateElement*. This is how the Design Rationale concepts, which in turn inherit from *Annotation*, extend the Enterprise Architecture Modeling Language definition (i.e. the Design Rationale Domain Specific Modeling Language is designed as an Enterprise Architecture Modeling Language profile), and can be processed by the Enterprise Architecture Modeling Language associated tools. Hence, the combination approach used is that of extension. One or more *Annotations* can be associated to one or more *ArchimateElements* (cf. the association between them in the Meta-Model, Figure 6.12). *ArchimateElement* is a root concept in the ArchiMate Meta-Model (not in the standard definition, but in the Archi implementation used for editor building).

### 6.3.2 The Concrete Syntax

The concrete syntax was then defined (cf. *Define "display-surface" MM*, Figure 5.1 - as the collaboration Domain Specific Modeling Language is a Domain Specific Modeling Language, we used the same approach for language definition). The graphical representations have been chosen, following, for the most [Wolf 08]. They are presented in Figure 6.13 (cf. concepts from the Meta-Model, Figure 6.12).

### 6.3.3 The Semantics

The semantics (cf. *Define semantics*, Figure 5.1) of a Design Rationale Domain Specific Modeling Language is particular regarding implementation and consists in storing and retrieving the Design Rationale. Currently, the Design Rationale is stored in the model and

can be retrieved by direct visualization. More advanced retrieving systems can be integrated as Off the Shelf components, using a process like the one we proposed in Section 5.2. Or even some reasoning operations that propose advice to designers, may be integrated (cf. *Reason on DR*, Figure 5.1).

### 6.3.4 Language-specific Tools

The editor (cf. *Generate graphical editor*, Figure 5.1), as in the case of the Telecommunications ArchiMate extension, was built on top of Archi. The palette was extended with Design Rationale specific concepts (cf. Figure 6.13), from which the designer can select, drag and drop the desired ones. Association relations can be used to connect the Design Rationale concepts to *ArchimateElements* (cf. the association relation between *Annotation* and *ArchimateElement* in the Design Rationale Meta-Model, Figure 6.12).

## 6.4 On Modeling Language Interoperability

This section deals with the *MA 4 Inter-operate*. Interoperability between the system Domain Specific Modeling Languages is ensured by the relations defined between ArchiMate layers (cf. Section 2.5.4). As they are extensions of ArchiMate, these relations apply to them as well. This solves the main problem of interoperability between the system Domain Specific Modeling Languages. Therefore, in this thesis, there was no need to implement the Model Transformations proposed in Section 5.4. However, it is the responsibility of the designer to make sure the relations between ArchiMate layers are correct and complete. In addition, the designer is also responsible for a number of constraints coming from downstream.

However, these relations are limited to layers. Interoperability between viewpoints still remains an issue. The proposal presented by us in Section 5.4 can be used to tackle this. To write Model Transformations, languages like QVT or ATL can be used. For ontology matching, evaluations [Euzenat 10] suggest ASMOV [Jean-Mary 09] as a good mature candidate tool.

## 6.5 Integration of Domain Specific Modeling Languages and Off the Shelf Tools in the Service Creation Environment

All tools have been developed as Eclipse plug-ins or plug-in extensions or are executed by Eclipse plug-ins. Together, they form the Service Creation Environment (cf. Figure 6.14). Figure 6.14 presents how the interactions between two roles follow a general trend of a waterfall (cf. Figure 4.2), starting from Service Provider, to Service Developer. Of course, these roles interact more closely here than in the more abstract telecommunications service construction process. The separation between their activities and the template of four activities are much less clear at this more detailed level of abstraction. However, the general trend of waterfall is still present.

The Telecommunications ArchiMate extension has an editor which is a plug-in extension of the ArchiMate editor Archi, an Eclipse plug-in. The Telecommunications ArchiMate

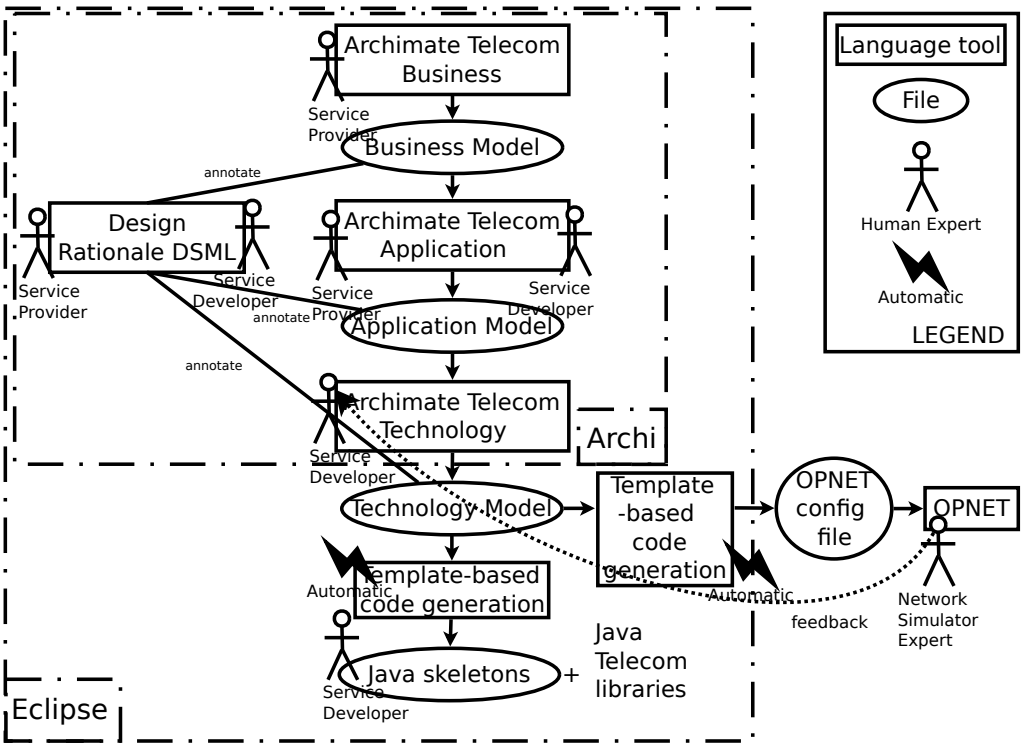


Figure 6.14 : Overview of the Service Creation Environment architecture.

Business and Application layers are Domain Specific Modeling Languages targeted at Service Providers. The Telecommunications ArchiMate Application and Technology layers are Domain Specific Modeling Languages targeted at Service Developers. Designers of both Service Provider and Service Developer roles use the Design Rationale Domain Specific Modeling Language to annotate concepts of the Telecommunications ArchiMate Business, Application and Technology layers. The Design Rationale Domain Specific Modeling Language editor is also an extension of Archi.

The technology model contains sufficient details to be fed as input to a network simulator (cf. Figure 6.14). Of course, a model-to-model transformation is necessary, implemented as a template-based code generation executed by OpenArchitectureWare, an Eclipse plug-in. The simulator expert then interprets the simulation results and makes recommendations to the Service Developer, who changes the technology model accordingly. Java code can be generated from models at any of the three ArchiMate layers, but the code generated from the Technology layer is the most useful. Code generation is based on templates that are run by OpenArchitectureWare, an Eclipse plug-in. The resulted Java code can be detailed and expanded by Service Developers using Telecommunications specific libraries.

Among the four modeling activities proposed in the telecommunications service construction process (cf. Section 4.1, Figure 4.1), tools for three of them have been developed in this thesis:

1. For *MA 1 Model*:
   (a) The Domain Specific Modeling Languages, for whose grammars (Meta-Models) (cf. Section 6.1.2) we have added a total of 46 concepts specific to telecommu-

nications;

(b) The Domain Specific Modeling Languages, for whose concrete syntaxes (cf. Section 6.1.3) we have added a total of 46 images, corresponding to the 46 concepts from Meta-Models;

(c) The Domain Specific Modeling Languages, for whose semantics (code generation templates developed in Xpand) (cf. Section 6.1.4 and Section 6.1.5.3) Iyass Alloush developed 650 lines, 39802 B (bytes) of code during his six-month masters internship (cf. Acknowledgements);

(d) A graphical editor (the Java code specific to the telecommunications concepts added to ArchiMate) (cf. Section 6.1.5.2), for which we developed 395,8 KB (kilobytes) of manually added code (without the generated code from the model through EMF).

2. For *MA 2 Test*: the code generation/model transformation from the Telecommunications ArchiMate Technical layer Meta-Model extension to the Meta-Model of OPNET network simulation tool (Section 6.2.2), for which Iyass Alloush developed 1486 lines, 122429 B of code during his six-month masters internship (cf. Acknowledgements).
3. For *MA 3 Collaborate*: the Design Rationale Domain Specific Modeling Language (Meta-Model, concrete syntax and graphical editor - cf. respectively Section 6.3.1, Section 6.3.2 and Section 6.3.4), for which: we have added 6 concepts and 6 images, and the graphical editor (108,8 KB of manually added code) has been developed as part of a two-week masters internship of Adil Meribaa and Mosbah Lassoued - cf. Acknowledgements.
4. For *MA 4 Inter-operate*: no tools, model transformations or ontologies have been developed.

As future developments of these tools, we indicate:

1. For *MA 1 Model*: extending the Domain Specific Modeling Languages to model more behavior;
2. For *MA 2 Test*: automatizing more the integration of simulation/testing results into the model;
3. For *MA 3 Collaborate*: integrating a Design Rationale engine;
4. For *MA 4 Inter-operate*: writing the proposed model transformations.

In conclusion, all components that form our proposed Service Creation Environment are integrated in Eclipse. OPNET is an Off the Shelf component and therefore, exterior to Eclipse and the Service Creation Environment.

## 6.6 Discussion

The Service Creation Environment we proposed in this chapter contributes towards fulfilling the requirements of Service Providers and Service Developers for Service Creation Environment:

– *Req 1 An overall model*: through the use of Enterprise Architecture, by extending an Enterprise Architecture Modeling Language. The models produced at the three layers of ArchiMate, together with the relations between models (conform to the relations between layer Meta-Models), offer a unified, overall model of business, application and technological issues.

- *Req 2 Domain specificity*: by proposing Telecommunications-specific Domain Specific Modeling Languages. These offer focused power and expressiveness for the telecommunications domain. This requirement is also fulfilled through the integration of telecommunications-specific Off the Shelf components.
- *Req 3 Rapid prototyping*: through a high degree of automation, offered by code generation towards both executable code and simulation tools.
- *Req 4 Collaborative support*: through the Design Rationale Enterprise Architecture Modeling Language extension, which contributes towards capturing different points of view, promoting coordination and building consensus.
- *Req 5 Early verification/simulation*: by leveraging existing network simulators. Their results are interpreted by human specialists who formulate recommendations for changing simulated models. This is done on models, not on fully-implemented services, thus solving issues early and reducing costs.
- *Req 6 Integration*: through the reuse of relations defined by ArchiMate between the three layers which apply at model level as well. Also, through the use of Eclipse plug-in architecture, which facilitates exchanging data (models) between the various tools. Finally, through the approach of using Model Transformations, ontologies and Higher Order model Transformations for viewpoint interoperability.
- *Req 7 Reuse*: by separating models at different layers, as the more abstract ones are less likely to change than the ones closer to the technological platform. Also, by enabling the capture and retrieval of decision rationale, which can be reused for other projects. Finally, by code generation, which enables reuse of models from which code is generated.
- *Req 8 Wide range of services*: through separating models at different layers. Different technological platforms can be considered at e.g. the Technology layer.
- *Req 9 Easy evolution of services*: through a high degree of automation: code generation, use of models and Model Transformations between them.



Regarding the Telecommunications Enterprise Architecture Modeling Language extension, other sources, such as Frameworx (cf. Section 2.2), can be used to extract initial Telecommunications Meta-Models. The Telecommunications Enterprise Architecture Modeling Language extension (i.e. the system Modeling Language family), the Design Rationale Enterprise Architecture Modeling Language extension, the integration of network simulators, finally the entire proposed Service Creation Environment need testing (cf. *Validate DSML family*, Figure 5.1). Depending on designers' feedback, a new iteration may be needed before the Service Creation Environment is deployed. However, designers from Service Providers and Service Developers are rare, expensive and subject to professional secret. Moreover, special skills are needed to devise pertinent surveys, interviews and other such means of gathering feedback.

As far as model testing is concerned, approaches for the (semi-)automatic integration of simulation results can be investigated.

The Design Rationale Domain Specific Modeling Language is integrated with the system Modeling Languages (i.e. the Enterprise Architecture Modeling Language) as far as both language definition and associated tools are concerned. Therefore, designers' reluctance to capture raising issues is decreased and thus, the Design Rationale capture issue is reduced as well. This will contribute to a better collaboration among designers, not only of the same team, but from teams of different roles.

To ensure viewpoint interoperability, alignment measures [Simonin 11] may be necessary to ensure that models remain coherent. Also, more automation can be assured to integrate constraints from downstream, from lower abstraction layers.

In this chapter we have proposed software tools, a Service Creation Environment, that answer the *RQ 2 Software tools*, to the Service Provider and Service Developer roles in the telecommunications service creation process. This Service Creation Environment contributes towards fulfilling all the identified Service Provider's and Service Developer's requirements for a Service Creation Environment. The Domain Specific Modeling Languages, through their focused expressive power, enable designers to achieve increased performance and accuracy. The transformation to the network simulator allows designers to simulate their models, to discover early errors or dimensioning problems. Capturing Design Rationale is beneficial for collaboration, as it promotes coordination, exposes different points of view, and promotes building consensus. Ensuring interoperability between software tools of different viewpoints reduces exchange problems and time.

# 7 Application of Proposed Telecom Service Construction Process and Tools to a Complete Case Study

”He placed both hands on the ground, then he said to him: 136
...
'... let me send you my big sister Enmebaragesi, to be your wife on the mountain!
...
Give me one of your auras of terror, and I will become your kinsman!'
His first aura of terror he gave him.
...
And a seventh time he said:
...
let me bring you in the mountain ...
...
Give me one of your auras of terror, and I will become your kinsman!'
His seventh aura of terror he gave him.
...
Having used up his seven auras of terror, he drew near to his lair.
Like a snake in the wine harbour, he followed behind him,
and making as if to kiss him, struck him on the cheek with his fist.”

Bilgames and Huwawa: 'The lord to the Living One's Mountain', in
*The Epic of Gilgamesh. A new translation*, Andrew George, Penguin Books, 2000

We applied the telecommunications service creation process proposed in Chapter 4, with the tools proposed in Chapter 6, to a conferencing telecommunications service, focusing on the Service Provider and Service Developer roles. We began with *MA 1 Model* of the telecommunications service using the Telecommunications Enterprise Architecture Modeling Language extension proposed in Section 6.1. We continued with *MA 2 Test*, using the network simulator chosen in Section 6.2. For the *MA 3 Collaborate*, design decisions on this model were captured using the Design Rationale Enterprise Architecture Modeling Language extension proposed in Section 6.3. The *MA 4 Inter-operate* of two adjacent layer Telecommunications Domain Specific Modeling Languages, due to the choice of designing them as ArchiMate extensions, is greatly simplified by the inter-layer relations defined by ArchiMate.

A conferencing service is a virtual meeting, done with the help of a set of telecommunications technologies (e.g., telephone, video, web), which allows two or more geographically

remote locations to interact in real-time via two-way video and/or audio and/or text transmissions simultaneously. Notable examples include WebEx™, Skype™ for software solutions, Polycom®, Tandberg™ for hardware or complete solutions. The conferencing service has been the subject of a significant amount of research (cf. e.g. the related work mentioned in Section 7.5).

## 7.1 Modeling a Telecom Service with the Proposed Service Creation Environment

This section exemplifies the use of the Telecommunications ArchiMate extension proposed in Section 6.1 for the *MA 1 Model* (cf. also the *Describes ...* activity, Figure 4.1).

### 7.1.1 The Business Model

We present here the business model of a multimedia conferencing service, described using the Telecommunications ArchiMate extension (cf. Figure 6.2, Figure 2.6, Figure 6.6). We chose to model only the successful case for the multimedia conferencing service, for simplicity reasons. This choice, however, omits much of the complexity of this service. This complexity has generated much research on the conferencing service (cf. e.g. Section 7.5).

A description of a conferencing service may use two roles: *Moderator* and *Participant* (cf. Figure 7.1). The concepts of Business: 'Role', 'Event', 'Process', Actor', 'Service', 'Activity' and relations of 'triggering', 'assignment', 'and-junction' used in Figure 7.1 are the ones defined by ArchiMate [The Open Group 09a]. After taking the decision (cf. Figure 7.1a)), the *Moderator* creates the conference and sends the connection details to the other participants (cf. Figure 7.1b)). A *Participant*, enters the conference at the appropriate time, using the details received from the *Moderator*, starts the conference, chats, speaks, and/or sends video at the same time, as many times and in any order (s)he may want, until exiting the conference (cf. Figure 7.1c)). Then the *Moderator* terminates the conference (cf. Figure 7.1d)).

### 7.1.2 The Application Model

Here, we discuss the application model of a multimedia conferencing service, described using the Telecommunications ArchiMate extension (cf. Figure 6.4, Figure 2.7, Figure 6.6).

Because of space constraints, the focus of this example for the Application layer is limited to the phase of entering the conference. This is the most difficult/important phase, because this is when the signaling is established (cf. Figure 7.2). The concepts of Application: 'ServiceFunctionalOperation', 'ServiceFunctionalComponent', and relations of 'triggering', 'assignment' used in Figure 7.2 are those defined by the Telecommunications ArchiMate extension. The *Client part of the conference system* launches the console and tries to join the *Participant* to the conference. If after three tries the connection is refused by the *Conference system*, an error message is displayed. If everything is alright, the *Participant* subscribes and joins the conference.

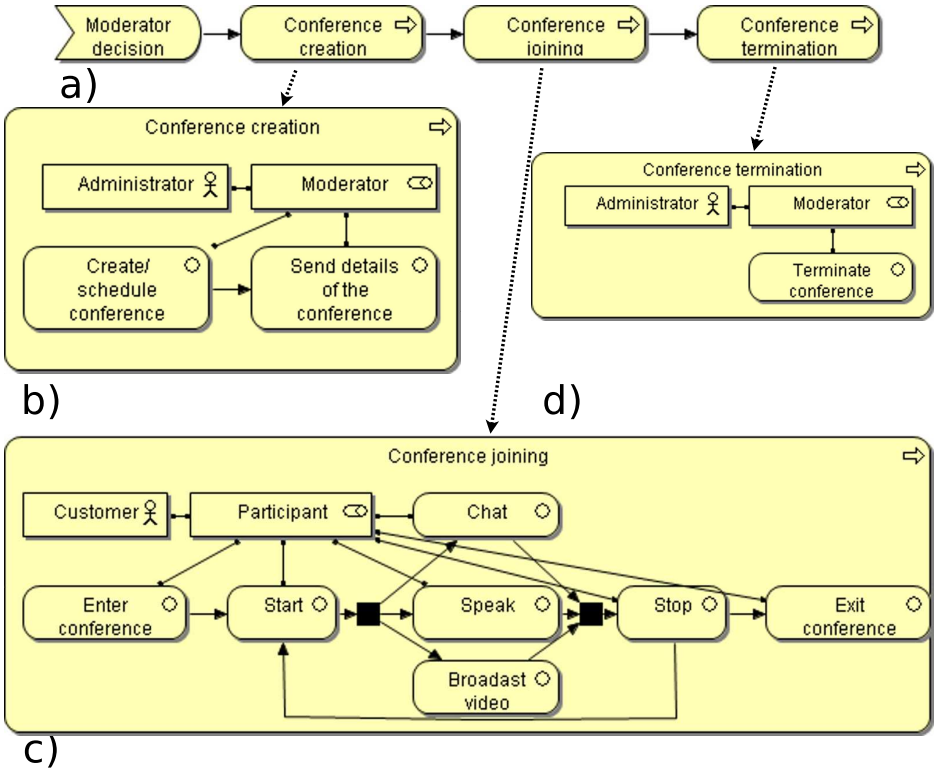


Figure 7.1 : The model of a conferencing service at the Telecom ArchiMate Business layer.

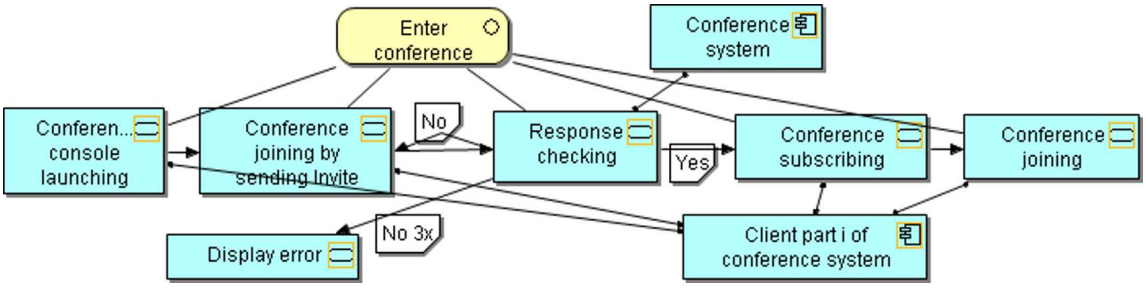


Figure 7.2 : Excerpt from the model of a conferencing service at the Telecom ArchiMate Application layer.

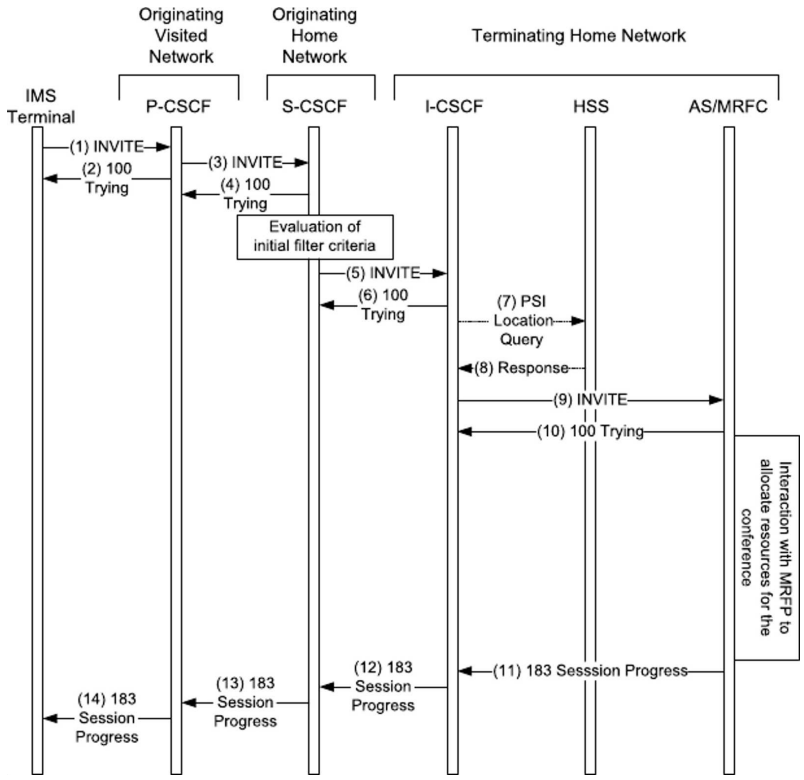


Figure 7.3 : The conference joining signals in IP Multimedia Subsystem, from [Camarillo 08].

### 7.1.3 The Technology Model

We present here the technology model of a multimedia conferencing service, described using the Telecommunications ArchiMate extension (cf. Figure 6.5, Figure 2.8, Figure 6.7).

The model is inspired from the signaling provided by [Camarillo 08]. Figure 7.3 shows an *IMS terminal* (cf. Section 1.4.1 for IP Multimedia Subsystem architecture) sending an *INVITE* request to a conference URI that is hosted at a different domain. The *I-CSCF* of the terminating domain consults the *HSS (7)* in order to resolve the *PSI* (i.e. the conference URI). The *HSS* provides the *I-CSCF (8)* with the address of the *AS/MRFC* corresponding to the *PSI* and the *I-CSCF* relays the *INVITE* request to that *AS/MRFC (9)*. Before returning a *183 (Session Progress)* response *(11)*, the *AS/MRFC* allocates resources at an *MRFP*.

It is at the Technology layer that a clear distinction between static entities and behavior is made. The static architecture of the conferencing service is presented in Figure 7.4. In this figure we modeled the network topology presented also in Figure 7.6 using the Telecommunications ArchiMate extension.

The behavior of the conferencing service is presented in Figure 7.5. In this figure we modeled the signaling from Figure 7.3 using the Telecommunications ArchiMate extensions. In the case of both static and behavioral technology models, as seen from comparing Figure 7.4 with Figure 7.6 and Figure 7.5 with Figure 7.3, it seems that the concrete syn-

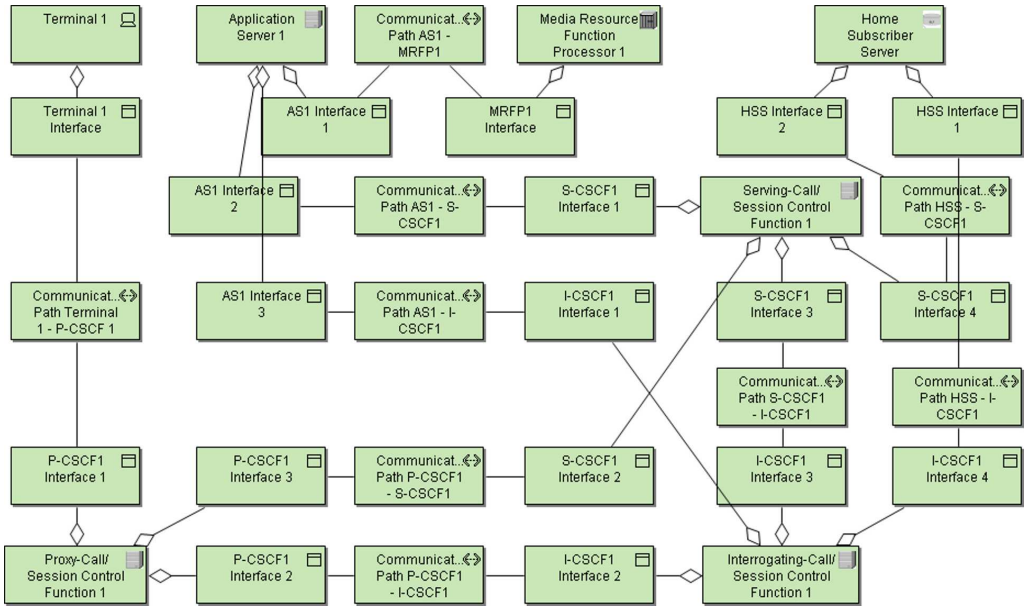


Figure 7.4 : Excerpt from the static model of a conferencing service at the Telecom ArchiMate Technology layer.

tax of the Technology layer of the Telecommunications ArchiMate extension is not the most concise one.

### 7.1.4 Executable Code

Here, we present an excerpt of the result of applying the template-based code generation that describes the semantics (cf. Section 6.1.4) of the Telecommunications ArchiMate extension. The excerpt shows part of the class *Clientpart1ofconferencesystem*, line 1 from Listing 7.1, corresponding to the *Client part i of conference system* 'ServiceFunctionalComponent' from Figure 7.2. Lines 16, 17, 18 from Listing 7.1 are especially interesting because they show the method *joinconferencebysendingInvite* calling the *checkresponse* method of class *Conferencesystem*, corresponding (cf. Figure 7.2) to the 'triggering' between the *Conference joining by sending Invite* 'ServiceFunctionalOperation' of the *Client part i of conference system* 'ServiceFunctionalComponent' and the *Response checking* 'ServiceFunctionalOperation' of the *Conference system* 'ServiceFunctionalComponent'.

## 7.2 Testing

This section illustrates the use of the selected network simulator and of the Model Transformation between the Telecommunications ArchiMate Technology layer MetaModel and its internal format, in Section 6.2, for the *MA 2 Test* (cf. also the *Tests ...* activity, Figure 4.1).

Both a static and a behavioral configuration of the OPNET model are needed. The static, topology model is presented in Figure 7.6. It corresponds to the Telecom ArchiMate Technology model, presented in Figure 7.4. The OPNET model has a star topology, with

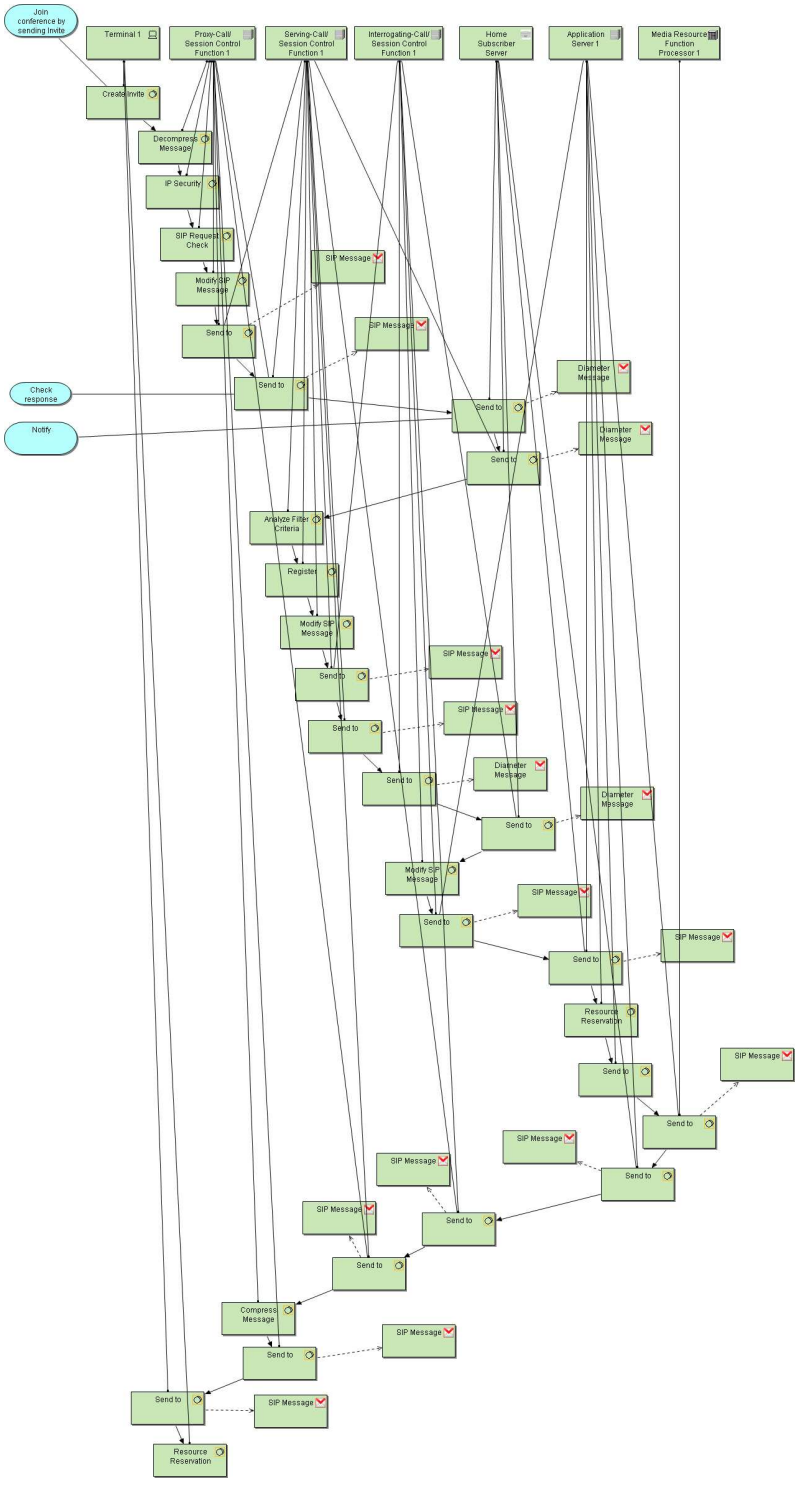


Figure 7.5 : Excerpt from the behavioral model of a conferencing service at the Telecom ArchiMate Technology layer.

```
public class Clientpart1ofconferencesystem{
  private String name = "Clientpart1ofconferencesystem";
  private String id = "78d717c8";
  public Clientpart1ofconferencesystem(){}
  public void setName(String newName){
    this.name = newName;}
  public String getName(){
    return name;}
  public void setId(String newId){
    this.id = newId;}
  public String getId(){
    return id;}
  public void joinconference(){}
  public void launchconferenceconsole(){
    this.joinconferencebysendingInvite();}
  public void joinconferencebysendingInvite(){
    Conferencesystem conferencesystem = new Conferencesystem();
    conferencesystem.checkresponse();}
  public void displayerror(){}
  public void subscribetoconference(){
    this.joinconference();}}
```

Listing 7.1 : Excerpt of generated Java code for a conferencing service application model (cf. Figure 7.2 )

one central router. Servers have been instantiated for each Telecom ArchiMate Technology *Node* type, and are connected to the central router.

The behavior of the OPNET model is captured in the configuration window (Figure 7.7). It corresponds to the behavioral model of the Telecom ArchiMate Technology model (Figure 7.5), both of them describing the conference joining signals in IP Multimedia Subsystem (cf. Figure 7.3).

The model is simulated with OPNET. Results on e.g. the traffic through each node (cf. Figure 7.8) can be obtained.

Depending on the test/simulation results, several iterations resulting in changes on the model may be necessary (cf. the *Test satisfactory?* activity, Figure 4.1). The decision to stop iterating is taken by the modeler together with the network simulation expert.

## 7.3 Collaboration

This section exemplifies the use of the Design Rationale ArchiMate extension proposed in Section 6.3 for the *MA 3 Collaborate* (cf. also the *Interacts ...* activity, Figure 4.1).

The example (Figure 7.9) builds upon the conferencing service Application model (Figure 7.2). The latter model presents the *Client part of the conference system* launching the console and trying to join the *Participant* to the conference. If after three tries the connection is refused by the *Conference system*, an error message is displayed. If everything is alright, the *Participant* subscribes and joins the conference.

However, displaying an error is not straightforward in a distributed system, and could pose a problem (cf. *Should the error be displayed?* in Figure 7.9). Not only *Usability*, but also system *Performance* and *Programming time* criteria should be considered. The basic proposals of *Displaying* and *Not displaying* are each assessed against all three criteria and the resolution to *Display* the error is made. The arguments, the rationale for this decision

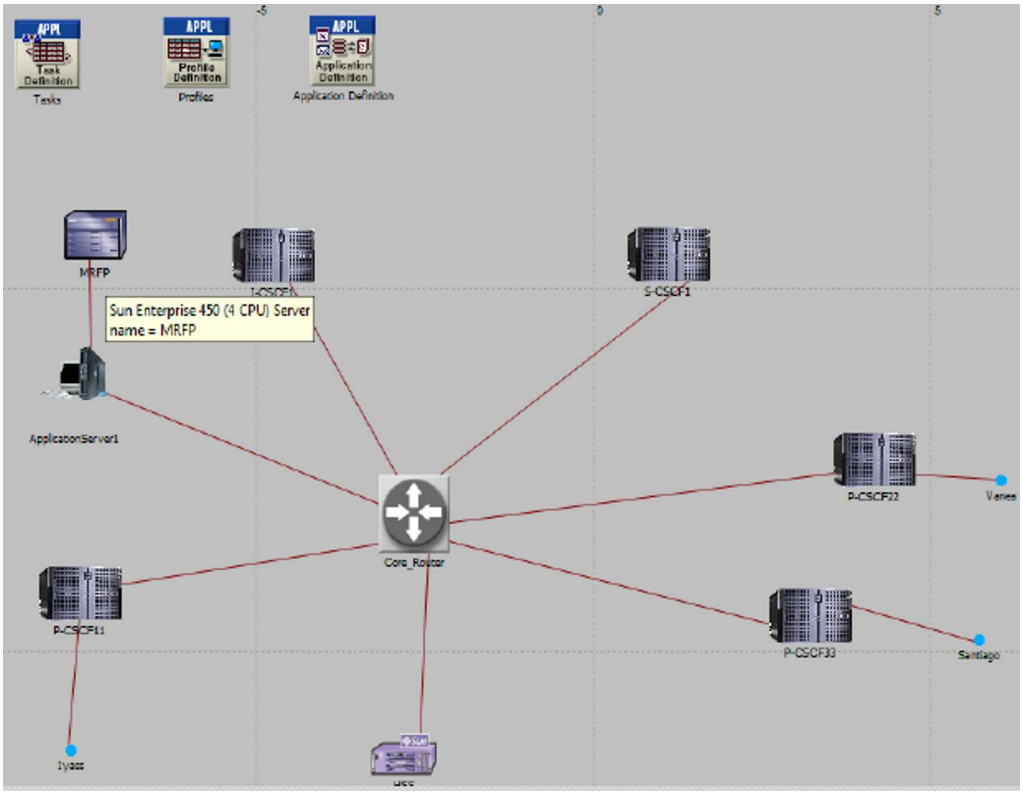


Figure 7.6 : The static configuration of the conferencing service model for OPNET.

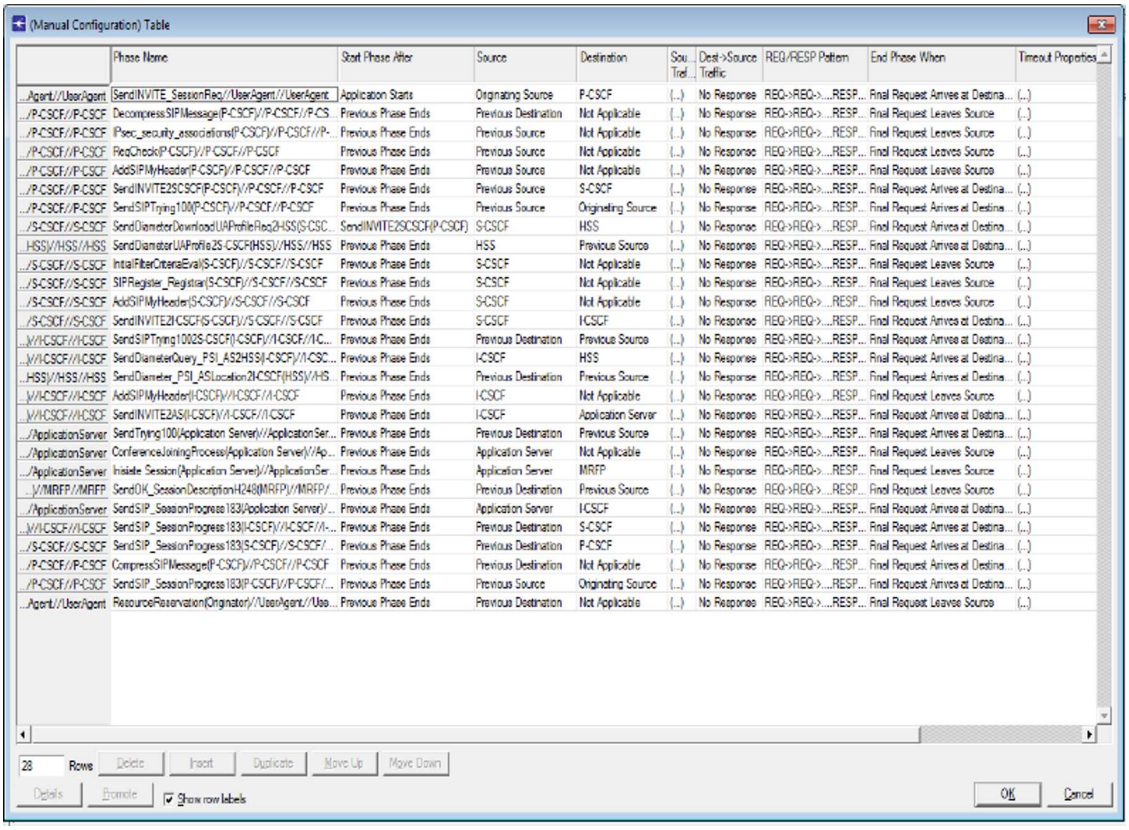


Figure 7.7 : The dynamic configuration of the conferencing service model for OPNET.

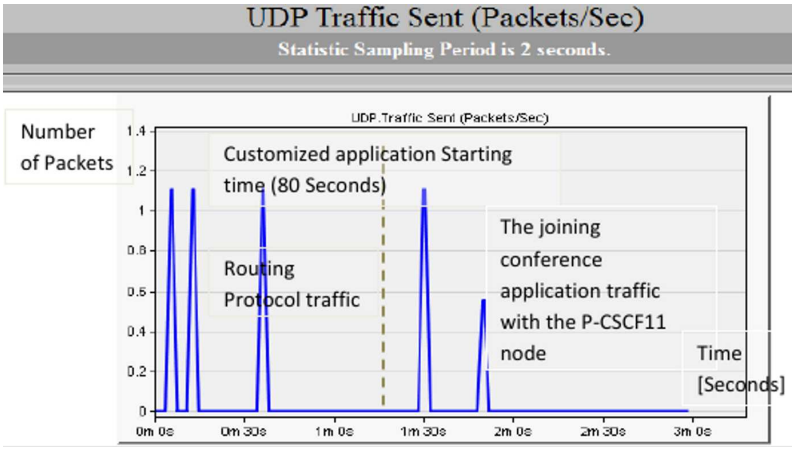


Figure 7.8 : OPNET simulation results for an IMS node of the conferencing service.

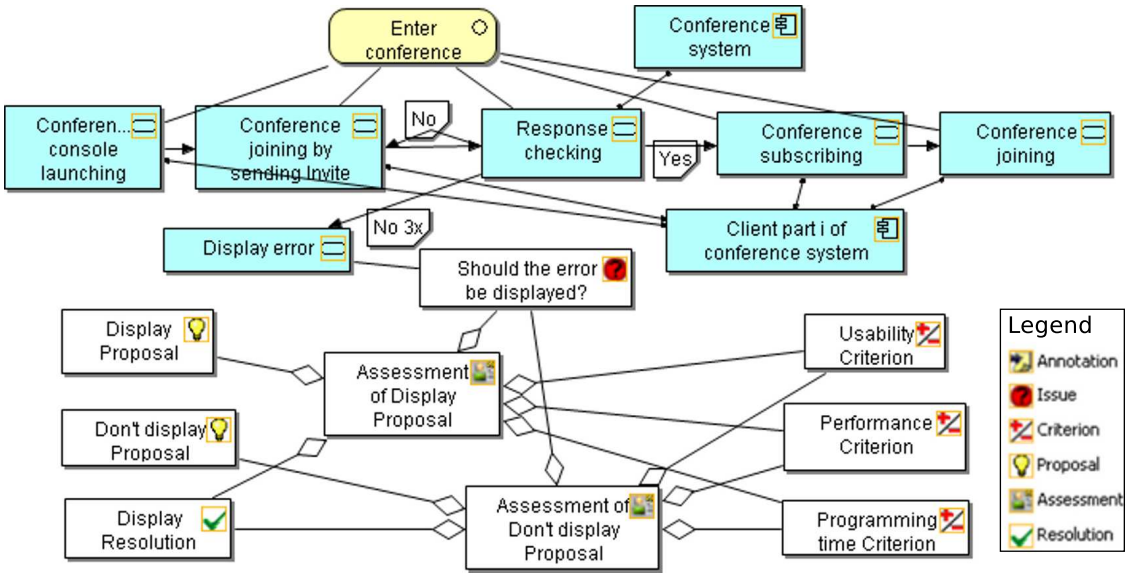


Figure 7.9 : Example of Collaboration Design Rationale ArchiMate extension used with a conferencing service example (cf. Figure 7.2) developed at the Application layer of the Telecommunications ArchiMate extension.

(no doubt related to giving much more importance, weight, to the *Usability* criterion than to the other two), is captured in the two *Assessments*. Of course, the same decision holds true for all errors, which means that the assessments can be capitalized upon or retrieved for similar cases.

The example shows how the Design Rationale Domain Specific Modeling Language can be used with a Telecommunications Domain Specific Modeling Language to capture, store (together with the telecommunications models) and retrieve (by direct visualization) decisions and their rationale.

## 7.4 On Interoperability

This section illustrates the use of the ArchiMate inter-layer relations as described in Section 6.4 for the *MA 4 Inter-operate* (cf. also the *Imports ...* and *Integrate ...* activities, Figure 4.1).

Figure 7.2 presents the Business 'Activity' *Enter conference* in relation with the Application 'Services'. The relation between them is that of *uses* (cf. Figure 2.10, Section 2.5.4).

## 7.5 Comparison with other Conferencing Service Models

Much research has been done on the conferencing service. Most of it focused on the technology, with few modeling it at the application or business layers.

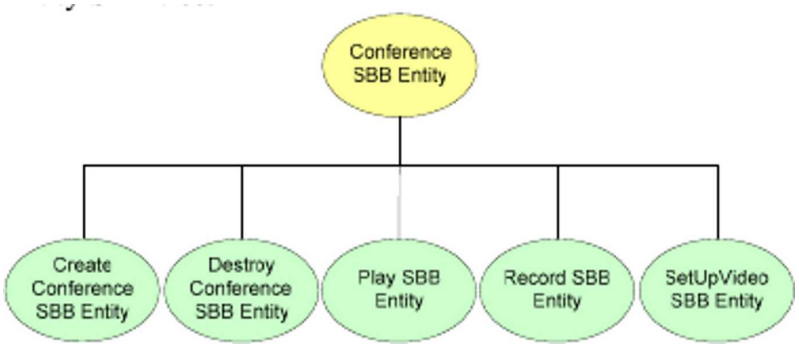


Figure 7.10 : Conference Service Building Block Entity tree, from [Bo 10a].

### 7.5.1 Conferencing Service Business Models

As far as we know, there is no model of a conferencing service that could be considered similar to that of a business layer model.

### 7.5.2 Conferencing Service Application Models

An example of a model that could be considered similar to an application layer model is presented in Figure 7.10, from [Bo 10a]. The figure presents a JAIN-SLEE Service Building Block (SBB), the *Conference SBB Entity*, as a tree, and its functionalities (e.g. *Create Conference SBB Entity*), as children of the parent node. The functionalities of Service Building Blocks may be considered similar to *ServiceFunctionalComponents* (cf. Figure 6.4). However, there is no formalism to specify the difference between e.g. *SBB Entity* and *SBB Functionality*.

### 7.5.3 Conferencing Service Technology Models

Most models of a conferencing service can be considered similar to models at the technology layer. For example, Figure 7.11, from [Cho 05], presents a static technology architecture. However, the types of concepts and the relations between them are not specified.

A frequent means of modeling (the static technology layer of) a conferencing service is by showing the topology of its network. For example, Figure 7.12, from [Cho 05], presents a distributed topology (cf. also Figure 7.6). These models also lack a formalism for representing concepts and relations.

Another frequent means of modeling (the behavioral technology layer of) a conferencing service is by representing the exchanged signals (cf. e.g. Figure 7.3). These models seem to be close to UML sequence diagrams, although authors usually do not specify whether they have used this or any other formalism to describe them.

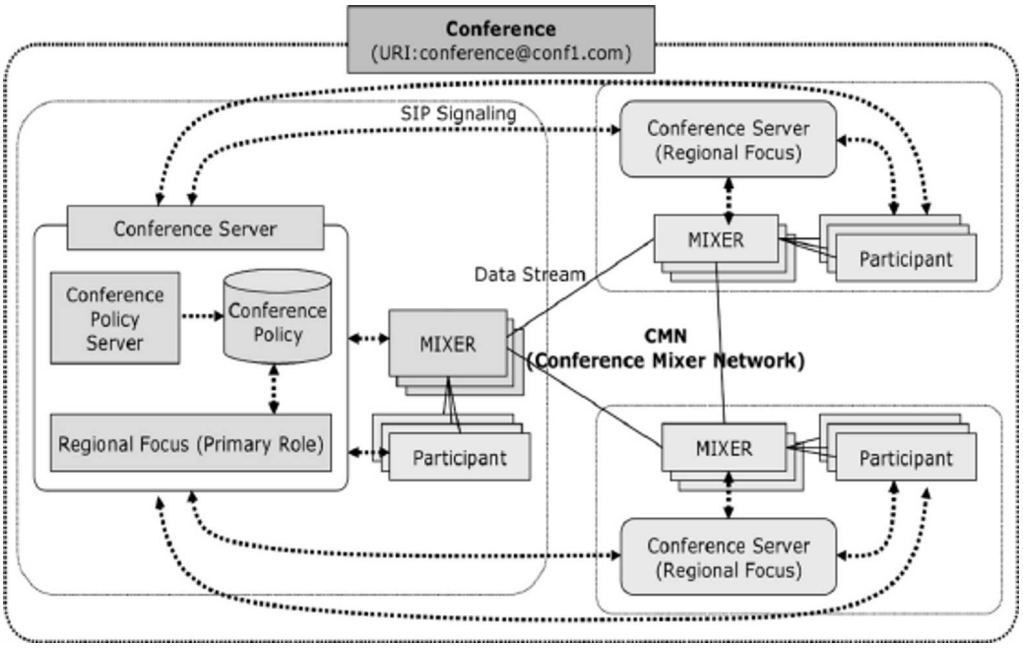


Figure 7.11 : Policy-based distributed management architecture for large-scale enterprise conferencing, from [Cho 05].

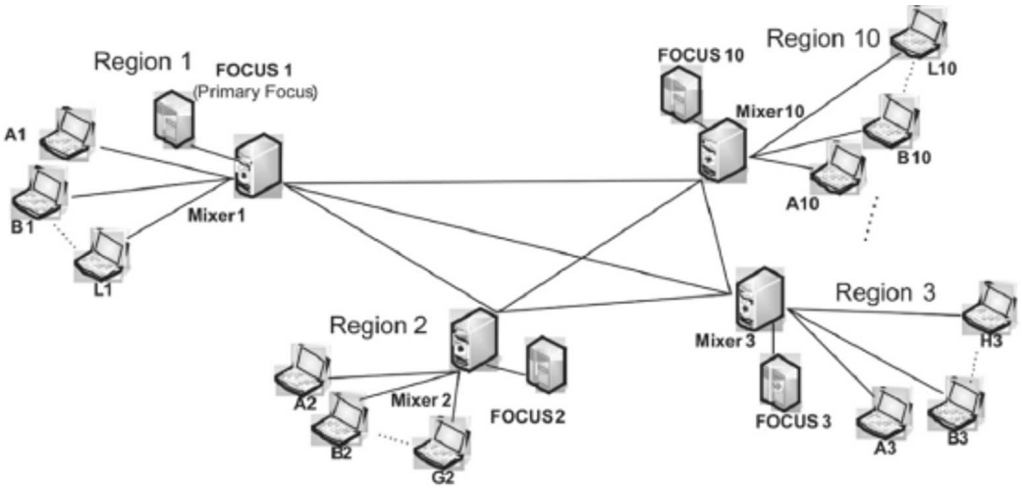


Figure 7.12 : Distributed conferencing network, from [Cho 05].

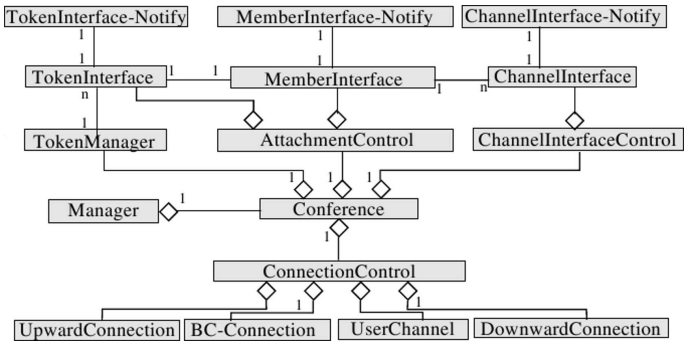


Figure 7.13 : Object model of the conferencing service, from [Trossen 98].

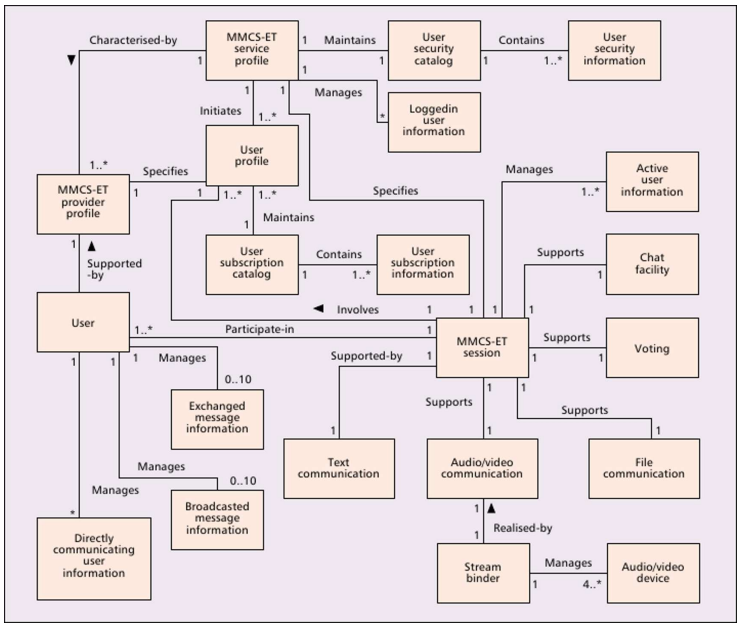


Figure 7.14 : The main conferencing service conceptual model, from [Adamopoulos 02].

### 7.5.4 Conferencing Service Object-Oriented Models

The models presented in the previous sections lacked a clear formalism. Some approaches modeled a conferencing service using object-oriented formalisms. For example, Figure 7.13 presents a Corba-based conferencing service, described in UML. Another example, Figure 7.14, models the main concepts of a conferencing service as a UML class diagram taking into account the service components proposed by TINA-C [TINA-C 97].

Most models of a conferencing service lack a formalism, a Modeling Language. Authors presenting such models usually use pictures to help convey their meaning, and as such, these models have only a communication value. They can not be used to generate executable code or perform tests, simulations. Moreover, because of the lack of formality, there may be some ambiguity over the meaning of certain concepts or relations. In contrast, the manner in which we model telecommunications services such as a conferencing service, does use a clear formalism and allows generating executable code and simulating.

The next step consists in doing more tests with real designers and industrial services. After a training session with the tools, designers should be given the task of modeling a service. The time they spend doing this should be compared with the time they need when using the traditional approach. In this way, our premise of reducing construction time, while maintaining at least the same level of quality and not increasing the cost, will be verified.



In this chapter we have exemplified our telecommunications service creation process and tools proposals using a conferencing service and have thus shown the advantages of our proposals.

# Conclusion and Perspectives

”Honour to the mighty Bilgames” 201

Bilgames and Huwawa: ’The lord to the Living One’s Mountain’, in
*The Epic of Gilgamesh. A new translation*, Andrew George, Penguin Books, 2000

We end this thesis by showing how and to what extent we provide answers to each of the research questions. Based on our proposals, more can be done in the future to leverage our results. We indicate some of these directions.

## Answering the Research Questions

In the context of world economies transitioning to services, telecommunications services, as the primary means of communication between different economic entities, are essential. The focus on the End User, the convergence with the Internet, the separation between the software and the hardware implementing a service system, and the telecommunications market deregulation have led to a revolution and a new era in the telecommunications industry. To answer these challenges, former national telecommunications providers have to reduce the construction time, from months to days, while affecting non-negatively other parameters (e.g., cost, quality of service - QoS, quality of experience - QoE, designer creativity) of new telecommunications services.

To tackle this problem, we have first broken it into three smaller research questions. To answer the *RQ 1 Construction process*, we have discussed several construction processes, based on Enterprise Architecture frameworks and inspired from software engineering. We have concluded that a waterfall-like process is the most suited, at the moment, for telecommunications actors. We have identified common designer activities, for modelers of all roles:

1. *MA 1 Model*: modeling the service from their viewpoint,
2. *MA 2 Test*: testing/verifying the service modeled from their viewpoint,
3. *MA 3 Collaborate*: collaborating with other designers from the same role,
4. *MA 4 Inter-operate*: inter-operating with the software used by designers from other roles.

We have chosen to base the process on models, the process having a high enough degree of formality to enable code generation, but still remaining flexible enough for designers to express their creativity. This choice of putting models in the center of the process is independent of the previous choice of the waterfall process, and so compatible with other types of processes (e.g. spiral).

The process describes the main activities of each role in the telecommunications service life-cycle. However, it does not provide details about each of the activities. Also, there

are other activities which are not captured by this process. Moreover, it is based on a waterfall model; therefore, it lacks the flexibility of iterative processes, for example.

Regarding the *RQ 2 Software tools*, we have focused on the Service Providers and Service Developers. For these two roles, for each of the main activities composing the proposed telecommunications service construction process, we have proposed a software tool or an approach for using existing tools. To *MA 1 Model* the service from their points of view, we have defined Domain Specific Modeling Languages as profiles for an Enterprise Architecture Modeling Language, ArchiMate. Through their focused expressive power, these languages enable designers to achieve increased performance and accuracy. However, determining the most appropriate concrete syntax (e.g. icons) needs several iterations.

To enable designers to *MA 2 Test* their models, we have generated configuration files, from models, for network simulators. Simulation allows designers to discover early errors or performance/dimensioning problems. However, simulation is a testing approach, it is not proof that the model is error-free, it can detect only certain errors. To achieve this, (formal) validation methods have to be adapted to models.

To enable designers to *MA 3 Collaborate*, we have defined a Domain Specific Modeling Language for capturing what is considered to be one of the most important elements of collaboration: Design Rationale. Design Rationale is the reasoning that goes into determining the design of an artifact. It can support collaboration by promoting coordination, exposing differing points of view, promoting building consensus, and thus reducing the time spent for collaboration purposes. However, collaboration is a much more vast subject and many improvements can be integrated (cf. next section).

In order to allow designers to *MA 4 Inter-operate* software from different points of view (e.g. Domain Specific Modeling Languages), we have proposed a process that lifts models - the artifacts central to our proposal - into ontologies. This enables the usage of existing software tools for ontologies to enrich and align them two by two. The alignment process results in a common ontology. The ontology is the bridge that enables interoperability between the two models that were initially lifted and the software tools that are based on those models. This reduces the number of exchange problems between software tools and it also saves construction time. While this interoperability approach is suitable for a family of Domain Specific Modeling Languages, it produces poor results if models do not have many elements in common.

To define these Domain Specific Modeling Languages and build these software tools, we have used a model-driven approach, our proposal to the *RQ 3 Tool building process*. This approach is based on models. They can be used to configure Meta Tools to generate specific tools (e.g. graphical editors), as input for code generation rules (e.g. for generating Java code), as input for transformations that lift them into ontologies (e.g. to ensure interoperability), as input for Model Transformations that translate them into formats understood by other tools (e.g. for simulation with OPNET). It confers much importance to the capacity to evolve. Changes in the model can be easily and almost automatically transmitted to the generated software tools. However, most of the Meta Tools are not industrialized yet and still have to implement features and repair bugs.

We have illustrated the construction process, together with the tools, using a multimedia conferencing service. The next step consists in doing more tests with real designers and industrial services. After a training session with the tools, they would be given the task of modeling a service. The time they spend doing this should be compared with the

time they need for the same activity, but using the traditional approach. Their feedback has to be integrated in the process and in the tools. Building new versions of tools will also test the evolution capacity of the tool construction process.

The service construction process, software languages and tools, together with the tool construction process we proposed in this work contribute to reducing the construction time of new telecommunications services, while providing the possibility of improved quality of service and increased involvement of the End User. Many developments can be envisaged based on the proposals we have made. We indicate some of these in the next section.

## Perspectives

Based on our work, more can be done to reduce new telecommunications service construction time. The focus on the End User will bring about the need to answer changing requirements faster and cheaper than a waterfall process can. Iterative processes (e.g. spiral), allowing integration of End User feedback earlier in the development process, will probably replace waterfall processes. This will probably have a negligible effect, if any, on Domain Specific Modeling Languages, and will emphasize their generative benefits even more. However, having several development iterations requires an even higher degree of interoperability and integration between Domain Specific Modeling Languages, to reduce interoperability/translation time. Therefore, the introduction of an iterative process will increase the interoperability challenge.

In the light of the recent document of mapping between TOGAF and Frameworx, publicized only at the end of this thesis, the proposed ArchiMate telecommunications extension could be updated so that it contains the Frameworx concepts at the Business and Application layers, as mapped with TOGAF. In this thesis, we used the concepts proposed by the Meta-Models of [Bertin 09a].

An open question central to service engineering is how to deal with issues related to the dynamic semantics of services. In this thesis we focused on the static semantics of Domain Specific Modeling Languages. The only behavior that we did tackle is related to method call (cf. Section 6.1.4). Further developments of modeling the dynamic semantics remain in perspective.

Although we provide a way of testing the models through simulation, one can envision more effective methods. The same generative approach of configuration files we have used for enabling simulation, can be used to connect to existing (formal) validation tools. These can validate different general properties of the models, like correctness, or domain specific properties, like lack of deadlock.

One way of tackling the interoperability challenge, besides increasing automation, consists in offering more opportunities and support for human collaboration. Tools for distributed work (e.g. multi-writer, real-time, distributed collaborative editors that also include VoIP, chat, white-board, and screen sharing functionality [Prechelt 11]), or dedicated hardware (e.g. multi-touch table), or automatic extraction of collaboration information from recordings can affect collaboration quality and reduce the time allotted to it. Another interoperability-related perspective consists in the implementation of the Model Transformations proposed in Section 5.4.

The proposed Service Creation Environment can be extended to encompass composition of telecommunications services that have been previously defined. Recommendations

based on service signatures can be proposed. These are particularly pertinent at the Application layer, where constraints from both Business and Technology layers can meet.

Another way to increase automation and generation, and also answer the *Req 7 Reuse*, is to model telecommunications services as part of a product-line. This can be done for existing telecommunications services, through analysis of their commonalities and variabilities. The discovery of variation points is especially pertinent for the possibility of discovering new telecommunications services, which can then be proposed to the End User.

To conclude, our work reduces the construction time of new telecommunications services and represents the basis for other improvements that will reduce it even more. Faster provisioning of new telecommunications services, that answer better to consumers' needs, will increase the rate of development of new economic services in general, and will ultimately have a positive impact on world economic development.

# Index

# Bibliography


[3GPP 06] 3GPP. *Technical Specification Group Services and System Aspects; IP Multimedia Subsystem (IMS), Stage 2 (Release 7.5.0)*, 2006. 11

[3GPP 10] 3GPP. *TS 23.228 V10.3.1 IP Multimedia Subsystem (IMS) Stage 2 (Release 10)*, 2010. 76

[Abhayawardhana 07] V. Abhayawardhana, K. Kocan & J. Matthews. *Service blending for converged IMS services a BT Alcatel-Lucent collaboration.* BT Technology Journal, vol. 25, pp. 211–218, 2007. 15

[Achilleos 07] Achilleas Achilleos, Nektarios Georgalas & Kun Yang. *An Open Source Domain-Specific Tools Framework to Support Model Driven Development of OSS.* In David Akehurst, Regis Vogel & Richard Paige, editors, Model Driven Architecture- Foundations and Applications, volume 4530 of *LNCS*, pp. 1–16. Springer, 2007. 46

[Achilleos 08a] A. Achilleos, K. Yang & N. Georgalas. *A Model Driven Approach to Generate Service Creation Environments.* In IEEE Global Telecommunications Conference GLOBECOM, pp. 1 –6, 2008 2008. 46

[Achilleos 08b] A. Achilleos, Kun Yang, N. Georgalas & M. Azmoodech. *Pervasive Service Creation using a Model Driven Petri Net Based Approach.* In International Wireless Communications and Mobile Computing Conference (IWCMC '08), pp. 309 –314, aug. 2008. 46, 57

[Adamopoulos 00] D.X. Adamopoulos, G. Pavlou & C.A. Papandreou. *Development of new telecommunications services in distributed platforms: a structured approach.* In IEEE International Conference on Communications (ICC), volume 1, pp. 222 –226 vol.1, 2000. 6

[Adamopoulos 02] D.X. Adamopoulos, Dionisis X. Adamopoulos & George Pavlou. *Advanced Service Creation Using Distributed Object Technology*, 2002. xxix, xxx, 5, 109

[Adamopoulos 09] D.X. Adamopoulos. *A service-centric approach for exploiting network intelligence.* In 2nd International Conference on the Applications of Digital Information and Web Technologies (ICADIWT), pp. 145 –150, aug. 2009. 6

[Aho 72] A. V. Aho & J. D. Ullman. The theory of parsing, translation, and compiling. Prentice-Hall, Inc., 1972. 41

[Ahuja 10] A. Ahuja, J. Simonin & R. Nedelec. *A Model-Driven Tool for Telecom Service Development Process.* In IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 427 –429, 2010. 45

[Aktemur 09] Baris Aktemur & Sam Kamin. *A comparative study of techniques to write customizable libraries.* In ACM Symposium on Applied Computing, pp. 522–529, Hawaii, USA, 2009. 86

[Alhir 02] Sinan Si Alhir. A Guide to Successfully Applying the UML. Springer, 2002. 42

[Arnold 10] Heinrich Arnold, Michael Erner, Peter Mockel & Christopher Schlaffer. *Cross-over Application of Enterprise Architecture and Modularization in Telco R&D.* In Heinrich Arnold, Michael Erner, Peter Mockel & Christopher Schlaffer, editors, Applied Technology and Innovation Management, pp. 116–131. Springer, 2010. xxxvi, 36

[AT&T News 09] AT&T News. *AT&T U-verse TV Marks 2 Million Customer Milestone*, 2009. http://www.att.com/gen/press-room?pid=4800&cdvn=news&newsarticleid=30203&mapcode=. 12

[Aubonnet 01] T. Aubonnet & N. Simoni. *PILOTE: a service creation environment in next generation networks.* In IEEE Intelligent Network Workshop, pp. 36–40. IEEE, 2001. 13

[Avison 03] David E. Avison & Guy Fitzgerald. *Where now for development methodologies?* Commun. ACM, vol. 46, pp. 78–82, January 2003. 51

[Ayala 11] Claudia Ayala, Oyvind Hauge, Reidar Conradi, Xavier Franch & Jingyue Li. *Selection of third party software in Off-The-Shelf-based software development-An interview study with industrial practitioners.* J. Syst. Softw., vol. 84, pp. 620–637, April 2011. 68

[Azmoodeh 05] M. Azmoodeh, N. Georgalas & S. Fisher. *Model-driven systems development and integration environment.* BT Technology Journal, vol. 23, pp. 96–110, 2005. 45, 46

[Baker 05] Paul Baker, Shiou Loh & Frank Weil. *Model-Driven Engineering in a Large Industrial Context - Motorola Case Study.* In Lionel Briand & Clay Williams, editors, Model Driven Engineering Languages and Systems, volume 3713 of *LNCS*, pp. 476–491. Springer, 2005. 56

[Barbero 07] Mikaël Barbero, Frédéric Jouault, Jeff Gray & Jean Bézivin. *A practical approach to model extension.* In Proc. of the 3rd European conf. on Model driven architecture-foundations and applications, ECMDA-FA, pp. 32–42, Haifa, Israel, 2007. 76

[Basole 08] R. C. Basole & W. B. Rouse. *Complexity of service value networks: conceptualization and empirical investigation.* IBM Syst. J., vol. 47, pp. 53–70, January 2008. xxxv

[Belina 89] F. Belina & D. Hogrefe. *The CCITT-specification and description language SDL.* Comput. Netw. ISDN Syst., vol. 16, pp. 311–341, March 1989. 14

[Bernardi 11] Simona Bernardi, Jose Merseguer & Dorina Petriu. *A dependability profile within MARTE.* Software and Systems Modeling, vol. 10, pp. 313–336, 2011. 75

[Berndt 94] Hendrik Berndt, Peter Graubmann & Masaki Wakano. *Service Specification Concepts in TINA-C.* In Proc. of the 2nd Intl. Conf. on Intelligence in Broadband Services and Networks: Towards a Pan-European Telecommunication Service Infrastructure, pp. 355–366, London, UK, 1994. xxix, 4, 6

[Berrisford 09] G. Berrisford & M. Lankhorst. *Using ArchiMate with TOGAF–Part 1: Answers to nine general questions about methods.* Via Nova Architectura, https://doc.novay.nl/dsweb/Get/Document-101474, 2009. 29

[Bertin 09a] Emmanuel Bertin. *Architecture of communication services in a convergence context (in French).* PhD thesis, National Institut of Telecommunications and Pierre and Marie Curie University - Paris 6, 2009. xxix, 76, 78, 79, 113

[Bertin 09b] Emmanuel Bertin & Noel Crespi. *Service business processes for the next generation of services: a required step to achieve service convergence.* Annals of Telecommunications, vol. 64, pp. 187–196, 2009. 36

[Bezivin 04] Jean Bezivin. *In Search of a Basic Principle for Model Driven Engineering.* Novatica Journal, vol. 2, pp. 21–24, 2004. xxix, 40

[Biggerstaff 98] Ted J. Biggerstaff. *A perspective of generative reuse.* Ann. Softw. Eng., vol. 5, pp. 169–226, January 1998. 56

[Blum 09a] N. Blum, T. Magedanz, J. Kleessen & T. Margaria. *Enabling eXtreme Model Driven Design of Parlay X-based Communications Services for End-to-End Multiplatform Service Orchestrations.* In 14th IEEE Intl. Conf. on Engineering of Complex Computer Systems, pp. 240 –247, 2009. 16, 46

## BIBLIOGRAPHY


[Blum 09b] Niklas Blum, Thomas Magedanz, Florian Schreiner & Sebastian Wahle. *From IMS Management to SOA Based NGN Management.* J. Netw. Syst. Manage., vol. 17, pp. 33–52, June 2009. xiii, 9, 11

[Blum 10] Niklas Blum, Irina Boldea, Thomas Magedanz & Tiziana Margaria. *Service-oriented Access to Next Generation Networks - from Service Creation to Execution.* Mobile Networks and Applications, vol. 15, pp. 356–365, 2010. 17

[Bo 10a] Cheng Bo, Zhang Shicheng, Hu Xiaoxiao & Chen Junliang. *Design and implementation of real-time communication components based open multimedia conferencing Web service over converged networks.* In IEEE GLOBECOM Workshops, pp. 632 –636, dec. 2010. xxx, 107

[Bo 10b] Cheng Bo, Zhang Yang, Zhou Peng, Duan Hua, Hu Xiaoxiao, Wang Zheng & Chen Junliang. *Development of Web-Telecom based hybrid services orchestration and execution middleware over convergence networks.* Journal of Network and Computer Applications, vol. 33, no. 5, pp. 620 – 630, 2010. xi, xxxvi, 17

[Booch 05] G. Booch, J. Rumbaugh & I. Jacobson. Unified Modeling Language User Guide, The Addison-Wesley Object Technology Series. Addison-Wesley Professional, Reading, MA, USA, 2005. 25, 42, 54

[Boucharas 10] Vasilis Boucharas, Marlies van Steenbergen, Slinger Jansen & Sjaak Brinkkemper. *The Contribution of Enterprise Architecture to the Achievement of Organizational Goals: A Review of the Evidence.* In Trends in Enterprise Architecture Research (TEAR), pp. 1–15, 2010. 25

[Bowen 89] TF Bowen, FS Dworack, CH Chow, N. Griffeth, GE Herman & Y.J. Lin. *The feature interaction problem in telecommunications systems.* In Software Engineering for Telecommunication Switching Systems (SETSS), 1989. xxxv

[Braek 96] Rolv Braek. *SDL basics.* Comput. Netw. ISDN Syst., vol. 28, pp. 1585–1602, June 1996. 14

[Braga 08] Daniele Braga, Stefano Ceri, Florian Daniel & Davide Martinenghi. *Mashing Up Search Services.* IEEE Internet Computing, vol. 12, pp. 16–23, 2008. 16

[Bryant 11] B.R. Bryant, J. Gray, M. Mernik, P.J. Clarke, R.B. France & G. Karsai. *Challenges and directions in formalizing the semantics of modeling languages.* Computer Science and Information Systems, vol. 8, no. 2, pp. 225–253, 2011. 65

[BT 10] BT. *21CN progress*, 2010. http://www.webcitation.org/5ttGQ7Sln. 12

[Camarillo 08] Gonzalo Camarillo & Miguel A. Garcia-Martin. The 3G IP Multimedia Subsystem: Merging the Internet and the Cellular Worlds. John Wiley & Sons Ltd, 2008. xxix, xxx, 11, 12, 100

[Carroll 06] Ray Carroll, Claire Fahy, Elyes Lehtihet, Sven van der Meer, Nektarios Georgalas & David Cleary. *Applying the P2P paradigm to management of large-scale distributed networks using a Model Driven Approach.* In Network Operations and Management Symposium (NOMS), 2006. 45, 47

[Cavalli 96] Ana R. Cavalli, Byoung-Moon Chin & Kilnam Chon. *Testing methods for SDL systems.* Comput. Netw. ISDN Syst., vol. 28, pp. 1669–1683, 1996. 57

[Ceh 11] Ines Ceh, Matej Crepinsek, Tomaz Kosar & Marjan Mernik. *Ontology driven development of domain-specific languages.* Comput. Sci. Inf. Syst., vol. 8, no. 2, pp. 317–342, 2011. 64, 65

[Chapman 93] M Chapman & N Gatti. *A model of a service life cycle.* In Proceedings of TINA '93, pp. 205–215, 1993. 6

[Chen 08] David Chen, Guy Doumeingts & François Vernadat. *Architectures for enterprise integration and interoperability: Past, present and future.* Comput. Ind., vol. 59, pp. 647–659, 2008. 24, 25

[Chesbrough 06] Henry Chesbrough & Jim Spohrer. *A research manifesto for services science.* Commun. ACM, vol. 49, pp. 35–40, July 2006. xxxiv

[Cheung 07] E. Cheung & K.H. Purdy. *Application Composition in the SIP Servlet Environment.* In IEEE International Conference on Communications (ICC), pp. 1985 –1990, june 2007. 15

[Chiprianov 09a] Vanea Chiprianov & Yvon Kermarrec. *An Approach for Constructing a Domain Definition Metamodel with ATL.* In Frédéric Jouault, editor, Model Transformation with ATL. 1st Intl. Workshop, MtATL 2009, Nantes, France, July 8-9, 2009, Preliminary Proc., pp. 18–33, 2009. xxvii

[Chiprianov 09b] Vanea Chiprianov & Yvon Kermarrec. *Model-based DSL Frameworks: A Simple Graphical Telecommunications Specific Modeling Language.* In 5emes journees sur l'ingenierie dirigee par les modeles (IDM), pp. 179–186, 2009. xxviii

[Chiprianov 09c] Vanea Chiprianov, Yvon Kermarrec & Patrick D. Alff. *A Model-Driven Approach for Telecommunications Network Services Definition.* In Springer, editor, Eunice'09: Proceedings of the 15th Open European Summer School and IFIP TC6. 6 Workshop on The Internet of the Future, volume 5733 of *LNCS*, pp. 199–207, Barcelona, Spain, 2009. xxvi

[Chiprianov 10] Vanea Chiprianov, Yvon Kermarrec & Siegfried Rouvrais. *Meta-tools for Software Language Engineering: A Flexible Collaborative Modeling Language for Efficient Telecommunications Service Design.* In FlexiTools2010 ICSE Workshop on Flexible Modeling Tools, 2010. xxvii, 56

[Chiprianov 11a] Vanea Chiprianov, Yvon Kermarrec & Siegfried Rouvrais. *Extending Enterprise Architecture Modeling Languages: Application to Telecommunications Service Creation.* In 9th Enterprise Engineering track at 27th ACM Symposium on Applied Computing (SAC), volume in press, Trento, Italy, 2011. xxiv, xxx, 78, 79

[Chiprianov 11b] Vanea Chiprianov, Yvon Kermarrec & Siegfried Rouvrais. *On the Extensibility of Plug-ins.* In 6th International Conference on Software Engineering Advances (ICSEA), pp. 557–562, Barcelona, Spain, 2011. xxv, 86

[Chiprianov 11c] Vanea Chiprianov, Yvon Kermarrec & Siegfried Rouvrais. *Practical Model Extension for Modeling Language Profiles. An Enterprise Architecture Modeling Language Extension for Telecommunications Service Creation.* In Journees nationales IDM, CAL, et du GDR GPL, 2011. xxvii, xxix, 77, 78, 79

[Chiprianov 11d] Vanea Chiprianov, Yvon Kermarrec & Siegfried Rouvrais. *Telecommunications Service Creation: Towards Extensions for Enterprise Architecture Modeling Languages.* In 6th Intl. Conf. on Software and Data Technologies (ICSOFT), volume 1, pp. 23–29, Seville, Spain, 2011. xxv, xxx, 80

[Chiprianov 11e] Vanea Chiprianov, Yvon Kermarrec & Siegfried Rouvrais. *Towards semantic interoperability of graphical DSMLs for telecommunications service design.* In 2nd Intl. Conf. on Models and Ontology-based Design of Protocols, Architectures and Services (MOPAS), Budapest, Hungary, 2011. xxvi, 73

[Chiprianov 12a] Vanea Chiprianov, Yvon Kermarrec & Siegfried Rouvrais. Formal and practical aspects of domain-specific languages: Recent developments, volume submitted, chapter Integrating DSLs into a Software Engineering Process: Application to Collaborative Construction of Telecom Services. IGI Global, 2012. xxiii

[Chiprianov 12b] Vanea Chiprianov, Yvon Kermarrec, Siegfried Rouvrais & Jacques Simonin. *Extending Enterprise Architecture Modeling Languages for Collaboration. Application to Telecommunications Service Design.* Software and Systems Modeling, vol. submitted, 2012. xxiii

[Cho 05] Yeong-Hun Cho, Moon-Sang Jeong, Jae-Wook Nah, Wee-Hyuk Lee & Jong-Tae Park. *Policy-based distributed management architecture for large-scale*

*enterprise conferencing service using SIP*. IEEE Journal on Selected Areas in Communications Magazine, vol. 23, no. 10, pp. 1934 – 1949, 2005. xxx, 107, 108

[Choi 06] N. Choi, I.Y. Song & H. Han. *A survey on ontology mapping*. ACM Sigmod, vol. 35, no. 3, p. 41, 2006. 71

[Cinderella 11] Cinderella. *Cinderella SDL. http://www.cinderella.dk/*, 2011. 14

[Ciocoiu 00] M. Ciocoiu & D.S. Nau. *Ontology-based semantics*. In International Conference on Principles of Knowledge Representation and Reasoning, pp. 539–546, 2000. 42

[Clark 01] T. Clark, A. Evans, S. Kent & P. Sammut. *The MMF approach to engineering object-oriented design languages*. In Workshop on Language Descriptions, Tools and Applications (LDTA), 2001. xxix, 42, 43, 66

[Combes 99] Pierre Combes & Beatrice Renard. *Service validation*. Computer Networks, vol. 31, no. 17, pp. 1817 – 1834, 1999. 8, 57

[Coplien 98] James Coplien, Daniel Hoffman & David Weiss. *Commonality and Variability in Software Engineering*. IEEE Softw., vol. 15, no. 6, pp. 37–45, 1998. 64

[Cosmadopoulos 08] Yannis Cosmadopoulos. *JSR 289: SIP Servlet v1.1*, 2008. 11, 14, 15

[Czarnecki 98] Krzysztof Czarnecki. *Generative Programming: Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models*. PhD thesis, Technical University of Ilmenau, Germany, October 1998. 86

[Czarnecki 06] K. Czarnecki & S. Helsen. *Feature-based survey of model transformation approaches*. IBM Syst. J., vol. 45, pp. 621–645, July 2006. 41

[de Almeida Falbo 02] Ricardo de Almeida Falbo, Giancarlo Guizzardi & Katia Cristina Duarte. *An ontological approach to domain engineering*. In SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering, pp. 351–358, New York, NY, USA, 2002. 64

[de Vrieze 11] Paul de Vrieze, Lai Xu, Athman Bouguettaya, Jian Yang & Jinjun Chen. *Building enterprise mashups*. Future Generation Computer Systems, vol. 27, no. 5, pp. 637 – 642, 2011. 17

[Demestichas 99] P.P. Demestichas, N.P. Polydorou, A.K. Kaltabani, N.I. Liossis, S. Kotrotsos, E.C. Tzifa & M.E. Anagnostou. *Issues in service creation for future open distributed processing environments*. In IEEE International Conference on Communications (ICC), 1999. 3

[Demirkan 08] Haluk Demirkan, Robert J. Kauffman, Jamshid A. Vayghan, Hans-Georg Fill, Dimitris Karagiannis & Paul P. Maglio. *Service-oriented technology and management: Perspectives on research and practice for the coming decade*. Electronic Commerce Research and Applications, vol. 7, no. 4, pp. 356 – 376, 2008. xxxiv

[Deursen 00] A. Deursen, P. Klint & J. Visser. *Domain-specific languages: an annotated bibliography*. SIGPLAN Not., vol. 35, no. 6, pp. 26–36, 2000. 56

[Droegehorn 08] Olaf Droegehorn, Immanuel Konig, Goulven Le-Jeune, Julien Cupillard, Mariano Belaunde & Erno Kovacs. *Professional and end-user-driven service creation in the SPICE platform*. In Proceedings of the 2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks, pp. 1–8, Washington, DC, USA, 2008. 17

[Dutoit 06] A.H. Dutoit, R. McCall, I. Mistrík & B. Paech. *Rationale management in software engineering: Concepts and techniques*. Rationale Management in Software Engineering, pp. 1–48, 2006. xxx, 58, 59, 89

[Efftinge 06] Sven Efftinge & Clemens Kadura. *OpenArchitectureWare 4.1 Xpand Language Reference*. Technical report, OpenArchitectureWare, 2006. 83

[Egyed 01] Alexander Egyed & Robert Balzer. *Unfriendly COTS Integration-Instrumentation and Interfaces for Improved Plugability.* In Proceedings of the 16th IEEE international conference on Automated software engineering, ASE '01, pp. 223–231, Washington, DC, USA, 2001. 68

[ETSI 06] ETSI. *Standard ES 202 391-1, Open Service Access (OSA); Parlay X Web Services; Part 1: Common (Parlay X 2), version 1.2.1*, 2006. 13

[ETSI 07] ETSI. *Standard ES 203 915-1, Open Service Access (OSA); Application Programming Interface (API); Part 1: Overview (Parlay 5), version 1.2.1*, 2007. 13, 15

[Euzenat 10] Jerome Euzenat, Alfio Ferrara, Christian Meilicke, Juan Pane, Francois Scharffe, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Svab-Zamazal, Vojtech Svatek & Cassia Trojahn dos Santos. *First results of the Ontology Alignment Evaluation Initiative 2010.* In Proc. of the 5th Intl. Wksh. on Ontology Matching (OM), with the 9th Intl. Semantic Web Conf. (ISWC), Shanghai, China, 2010. 91

[Evans 02] A. Evans, G. Maskeri, P. Sammut & J.S. Willans. *Building families of languages for model-driven system development.* In Proceedings of 2nd Workshop in Software Model Engineering, San Francisco, CA, 2002. 56

[Falcarin 08] P. Falcarin & C. Venezia. *Communication web services and JAIN-SLEE integration challenges.* International Journal of Web Services Research, vol. 5, no. 4, pp. 59–78, 2008. xiv, 10, 17

[Fatolahi 06] Ali Fatolahi & Fereidoon Shams. *An investigation into applying UML to the Zachman framework.* Information Systems Frontiers, vol. 8, pp. 133–143, 2006. 25

[FCC 96] FCC. *Telecommunications Act of 1996*, 1996. xxxiv, xxxvi

[Femminella 09] M. Femminella, R. Francescangeli, F. Giacinti, E. Maccherani, A. Parisi & G. Reali. *Scalability and performance evaluation of a JAIN SLEE-based platform for VoIP services.* In 21st International Teletraffic Congress, pp. 1 –8, 2009. 15

[Ferry 08] David Ferry. *JSR 240: JAINTM SLEE (JSLEE) v1.1*, 2008. 15, 83

[France 04] R. France, I. Ray, G. Georg & S. Ghosh. *Aspect-oriented approach to early design modelling.* IEE Proceedings - Software, pp. 173–185, 2004. 43

[France 07] Robert France & Bernhard Rumpe. *Model-driven Development of Complex Software: A Research Roadmap.* In 2007 Future of Software Engineering, FOSE '07, pp. 37–54, Washington, DC, USA, 2007. 40, 41, 76

[Frank 07] L. Frank, E. Luoma & P. Tyrvainen. *Market scope of vendors in the OSS software market.* In IEEE International Conference on Industrial Engineering and Engineering Management, pp. 2096 –2100, 2007. 63, 64

[Gabriel 10] P. Gabriel, M. Goulao & V. Amaral. *Do Software Languages Engineers Evaluate their Languages.* In Proc. of the XIII Congreso Iberoamericano en "Software Engineering" (CIbSE), pp. 149–162, Cuenca, Ecuador, 2010. 65

[Georgalas 07] Nektarios Georgalas, Shumao Ou, Manooch Azmoodeh & Kun Yang. *Towards a Model-Driven Approach for Ontology-Based Context-Aware Application Development: A Case Study.* In 4th International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES), pp. 21 –32, march 2007. 46

[Georgalas 09] N. Georgalas, A. Achilleos, V. Freskos & D. Economou. *Agile Product Lifecycle Management for Service Delivery Frameworks: History, Architecture and Tools.* BT Technology Journal, vol. 26, 2009. 3

[Gherbi 09] T. Gherbi, D. Meslati & I. Borne. *MDE between Promises and Challenges.* In 11th International Conference on Computer Modelling and Simulation (UKSIM), pp. 152 –155, march 2009. 39

## BIBLIOGRAPHY


[Glitho 03] R.H. Glitho, F. Khendek & A. De Marco. *Creating value added services in Internet telephony: an overview and a case study on a high-level service creation environment.* IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 33, no. 4, pp. 446 –457, 2003. 15

[Golaup 06] Assen Golaup & Hamid Aghvami. *A multimedia traffic modeling framework for simulation-based performance evaluation studies.* Comput. Netw., vol. 50, pp. 2071–2087, August 2006. 57

[Goncalves 09] M. K. Goncalves, C. R. B. de Souza & V. M. Gonzalez. Initial findings from an observational study of software engineers, pp. 498–503. International Conference on Computer Supported Cooperative Work in Design. 2009. 58

[Gouya 06] A. Gouya, N. Crespi & E. Bertin. *SCIM (Service Capability Interaction Manager) Implementation Issues in IMS Service Architecture.* In IEEE International Conference on Communications (ICC), volume 4, pp. 1748 –1753, june 2006. 15

[Graham 01] I. Graham. Object-oriented methods: Principles and practice, 3rd ed. Addison-Wesley, Reading, MA., 2001. 52

[Gray 03] Jeff Gray & Gábor Karsai. *An Examination of DSLs for Concisely Representing Model Traversals and Transformations.* In Proceedings of the 36th Annual Hawaii International Conference on System Science - Track 9, volume 9 of *HICSS*, 2003. 56

[Green 96] T.R.G. Green & M. Petre. *Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework.* Journal of Visual Languages and Computing, vol. 7, no. 2, pp. 131–174, 1996. 56

[Guedria 08] W. Guedria, Y. Naudet & D. Chen. *Interoperability Maturity Models - Survey and Comparison.* In Proc. of the OTM Confederated Intl Ws and Posters on On the Move to Meaningful Internet Systems, pp. 273–282, 2008. 25

[Haas 04] H. Haas & A. Brown. *Web services glossary.* W3C Working Group Note, vol. 11, 2004. 13

[Hållstrand 94] Joacim Hållstrand & Declan Martin. *Industrial Requirements on a Service Creation Environment.* In Proceedings of the 2nd Intl. Conf. on Intelligence in Broadband Services and Networks: Towards a Pan-European Telecommunication Service Infrastructure, pp. 17–25, London, UK, 1994. xiii, 7, 9

[Hermans 09] Felienne Hermans, Martin Pinzger & Arie Deursen. *Domain-Specific Languages in Practice: A User Study on the Success Factors.* In Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems (MODELS), pp. 423–437, 2009. 65

[Hetzel 91] William C. Hetzel & Bill Hetzel. The complete guide to software testing. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 1991. 57

[Hill 77] T.P. Hill. *On goods and services.* Review of income and wealth, vol. 23, no. 4, pp. 315–338, 1977. xxxiv

[Hussey 10] Kenn Hussey, Bran Selic & Toby McClean. *An Extended Survey of Open Source Model-Based Engineering Tools. Revision E.* Technical report, zeligsoft, May 2010. 44

[IBM 11] IBM. *IBM Rational SDL Suite. http://www-01.ibm.com/software/awdtools/sdlsuite/*, 2011. 14

[IEEE Comp. Soc. 00] IEEE Comp. Soc. *IEEE Recommended Practice for Architectural Description of Software Intensive Systems. IEEE Standard 1471-2000*, 2000. 24

[IFAC IFIP Task Force 99] IFAC IFIP Task Force. *GERAM: Generalised Enterprise Reference Architecture and Methodology*, 1999. 25

[Ince 06] A. Nejat Ince & Ercan Topuz, editors. Modeling and Simulation Tools for Emerging Telecommunication Networks. Needs, Trends, Challenges and Solutions. Springer, 2006. 57

[Ince 07] A. Nejat Ince & Arnold Bragg, editors. Recent Advances in Modeling and Simulation Tools for Communication Networks and Services. Springer, 2007. 57

[Iris 10] Iris & Reinhartz-Berger. *Towards automatization of domain modeling*. Data & Knowledge Engineering, vol. 69, no. 5, pp. 491 – 515, 2010. 64

[ISO 00] ISO. *ISO 15704:2000 Industrial automation systems - Requirements for enterprise-reference architectures and methodologies*, 2000. 23, 25

[ISO 06] ISO. *ISO 19439:2006 Enterprise integration - Framework for enterprise modelling*, 2006. 25

[ISO/IEC 07] ISO/IEC. *ISO/IEC FDIS 42010. Systems and software engineering Architecture description*, 2007. 24

[ITU-T 07] ITU-T. *Recommendation Z.100 (11/07): Specification and Description Language (SDL). http://www.itu.int/rec/T-REC-Z.100-199911-S/en*, 2007. 14

[ITU-T 08] ITU-T. *Recommendation Z.151 (11/08) User Requirements Notation (URN) - Language definition. http://www.itu.int/rec/T-REC-Z.151/en*, 2008. 14

[ITU-T 11a] ITU-T. *Recommendation Z.120 (O2/11) Message Sequence Chart (MSC). http://www.itu.int/rec/T-REC-Z.120/en*, 2011. 14

[ITU-T 11b] ITU-T. *Recommendation Z.161 (03/11) Testing and Test Control Notation version 3: TTCN-3 core language (TTCN). http://www.itu.int/rec/T-REC-Z.161-201103-I/en*, 2011. 14

[ITU 93] ITU. *Intelligent Network Reccomendations*, 1993. 10

[Jansen 08] Slinger Jansen, Sjaak Brinkkemper, Ivo Hunink & Cetin Demir. *Pragmatic and Opportunistic Reuse in Innovative Start-up Companies*. IEEE Softw., vol. 25, pp. 42–49, November 2008. 68

[Jean-Mary 09] Yves R. Jean-Mary, E. Patrick Shironoshita & Mansur R. Kabuka. *Ontology matching with semantic verification*. Web Semantics: Science, Services and Agents on the World Wide Web, vol. 7, no. 3, pp. 235 – 251, 2009. 91

[Jie 09a] Guo Jie, Cheng Bo, Chen Junliang & Lin Xiangtao. *Applying recommender system based mashup to web-telecom hybrid service creation*. In Proceedings of the 28th IEEE conference on Global telecommunications, GLOBECOM'09, pp. 3321–3325, Piscataway, NJ, USA, 2009. IEEE Press. 17

[Jie 09b] Guo Jie, Chen Junliang, Cheng Bo & Liu Dong. *A choreography approach for value-added services creation*. In Proceedings of the 5th International Conference on Wireless communications, networking and mobile computing, WiCOM'09, pp. 4948–4951, Piscataway, NJ, USA, 2009. IEEE Press. 16

[Jonkers 04] Henk Jonkers, Marc Lankhorst, Rene Van Buuren, Marcello Bonsangue & Leendert Van Der Torre. *Concepts for Modeling Enterprise Architectures*. Intl. Journal of Cooperative Information Systems, vol. 13, pp. 257–287, 2004. 24, 25

[Jonkers 06] Henk Jonkers, Marc Lankhorst, Hugo ter Doest, Farhad Arbab, Hans Bosma & Roel Wieringa. *Enterprise architecture: Management tool and blueprint for the organisation*. Information Systems Frontiers, vol. 8, pp. 63–66, 2006. 24

[Jordan 07] Diane Jordan & John Evdemon. *Web Services Business Process Execution Language Version 2.0*, 2007. 13

[JTC1/SC7/WG6 09] JTC1/SC7/WG6. *ISO/IEC CD 25010.3: Systems and software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Quality models for software product quality and system quality in use. Version 1.46*, 2009. 85

## BIBLIOGRAPHY


[Jung 11] Y. Jung, Y. M. Park, H. J. Bae, B. S. Lee & J. Kim. *Employing Collective Intelligence for User Driven Service Creation.* Ieee Communications Magazine, vol. 49, no. 1, pp. 75–83, 2011. 17

[Kang 90] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak & A. S. Peterson. *Feature-Oriented Domain Analysis (FODA) Feasibility Study.* Technical report, Carnegie-Mellon University Software Engineering Institute, 1990. 64

[Kappel 06] Gerti Kappel, Elisabeth Kapsammer, Horst Kargl, Gerhard Kramler, Thomas Reiter, Werner Retschitzegger & Manuel Wimmer. *Lifting Metamodels to Ontologies - A Step to the Semantic Integration of Modeling Languages.* In Proceedings of the ACM/IEEE 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS/UML), pp. 528–542, 2006. 71, 73

[Khlifi 08] H. Khlifi & J.-C. Gregoire. *IMS Application Servers: Roles, Requirements, and Implementation Technologies.* Internet Computing, IEEE, vol. 12, no. 3, pp. 40 –51, may-june 2008. xiii, xiv, 9, 10

[Khoury 07] Gerald R. Khoury. *A unified approach to enterprise architecture modelling.* PhD thesis, University of Technology, Sydney, 2007. 35, 36

[Kieburtz 96] Richard B. Kieburtz, Laura McKinney, Jeffrey M. Bell, James Hook, Alex Kotov, Jeffrey Lewis, Dino P. Oliva, Tim Sheard, Ira Smith & Lisa Walton. *A software engineering experiment in software component generation.* In Proceedings of the 18th international conference on Software engineering (ICSE), pp. 542–552, 1996. 56

[Kim 08] Sang Kim, Young Shin, Cho Yu, Seung Chung & Byung Lee. *Integrated service creation environment for open network services.* Annals of Telecommunications, vol. 63, pp. 167–181, 2008. 16

[Kitchenham 04] B. Kitchenham. *Procedures for performing systematic reviews.* Keele University, UK, vol. 33, 2004. 13

[Kleppe 03] A. G. Kleppe, J. Warmer & W. Bast. Mda explained: The model driven architecture: Practice and promise. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003. 40

[Kleppe 07] A. Kleppe. *A language description is more than a metamodel.* In 4th Intl. Workshop on Software Language Engineering, volume 1, 2007. 42

[Ko 07] A.J. Ko, R. DeLine & G. Venolia. *Information needs in collocated software development teams.* In Proceedings of the 29th Intl. Conf. on Software Engineering (ICSE), pp. 344–353, 2007. 58

[Kolberg 99] Mario Kolberg, Richard O. Sinnott & Evan H. Magill. *Experiences modelling and using formal object-oriented telecommunication service frameworks.* Comput. Netw., vol. 31, pp. 2577–2592, December 1999. 14

[Kolovos 09] Dimitrios S. Kolovos, Davide Di Ruscio, Alfonso Pierantonio & Richard F. Paige. *Different models for model matching: An analysis of approaches to support model differencing.* In Proc. of the ICSE Ws. on Comparison and Versioning of Software Models, pp. 1–6, Vancouver, Canada, 2009. 76

[Kosar 08] Tomaz Kosar, Pablo E. Martinez Lopez, Pablo A. Barrientos & Marjan Mernik. *A preliminary study on various implementation approaches of domain-specific language.* Information and Software Technology, vol. 50, no. 5, pp. 390–405, 2008. 65

[Kosar 10] T. Kosar, N. Oliveira, M. Mernik, V.J.M. Pereira, M. Črepinšek, C.D. Da & R.P. Henriques. *Comparing general-purpose and domain-specific languages: An empirical study.* Computer Science and Information Systems, vol. 7, no. 2, pp. 247–264, 2010. 56

[Kosmas 97] Nikolaos Kosmas & Kenneth J. Turner. *Requirements for Service Creation Environments.* In 2nd International Workshop on Applied Formal Methods in System Design, p. 133 - 137, 1997. xiii, 9

[KPN 11] KPN. *The All IP world 'Voice over IP'*, 2011. 12

[Kraemer 09] F.A. Kraemer, H. Samset & R. Braek. *An Automated Method for Web Service Orchestration Based on Reusable Building Blocks*. In IEEE International Conference on Web Services (ICWS), pp. 262 –270, july 2009. 16

[Kramler 06] G. Kramler, G. Kappel, T. Reiter, E. Kapsammer, W. Retschitzegger & W. Schwinger. *Towards a semantic infrastructure supporting model-based tool integration*. In Proceedings of the 2006 international workshop on Global integrated model management (GaMMa), pp. 43–46, 2006. 71

[Kruchten 08] Philippe Kruchten. *Controversy Corner: What do software architects really do?* J. Syst. Softw., vol. 81, pp. 2413–2416, December 2008. 52

[Krueger 92] Charles W. Krueger. *Software reuse*. ACM Comput. Surv., vol. 24, pp. 131–183, June 1992. 56

[Kubinidze 06] Nino Kubinidze, Ivan Ganchev & Mairtin O'Droma. *Network Simulator NS2: Shortcomings, Potential Development and Enhancement Strategies*. In A. Nejat Ince & Ercan Topuz, editors, Modeling and Simulation Tools for Emerging Telecommunication Networks, pp. 263–277. Springer US, 2006. 57

[Kunz 70] Werner Kunz, Horst W. J. Rittel, We Messrs, H. Dehlinger, T. Mann & J. J. Protzen. *Issues as elements of information systems*. Technical report, 1970. 59

[Kurtev 06] Ivan Kurtev, Jean Bézivin, Frédéric Jouault & Patrick Valduriez. *Model-based DSL frameworks*. In Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications (OOPSLA), pp. 602–616, 2006. 42, 66

[Ladd 94] David A. Ladd & J. Christopher Ramming. *Two Application languages in software production*. In Proceedings of the USENIX 1994 Very High Level Languages Symposium, Berkeley, CA, USA, 1994. 56

[Laga 08] N. Laga, E. Bertin & N. Crespi. *User-centric Services and Service Composition, a Survey*. In 32nd Annual IEEE Software Engineering Workshop (SEW), pp. 3 –9, oct. 2008. 16

[Lal 01] D. Lal, D.C. Pitt & A. Beloucif. *Restructuring in European telecommunications: modelling the evolving market*. European Business Review, vol. 13, no. 3, pp. 152–158, 2001. xxxvi

[Land 08] Rikard Land, Laurens Blankers, Michel Chaudron & Ivica Crnković. *COTS Selection Best Practices in Literature and in Industry*. In Proceedings of the 10th international conference on Software Reuse: High Confidence Software Reuse in Large Systems (ICSR), pp. 100–111, 2008. 68

[Land 09] Rikard Land, Daniel Sundmark, Frank Lüders, Iva Krasteva & Adnan Causevic. *Reuse with Software Components - A Survey of Industrial State of Practice*. In Proceedings of the 11th International Conference on Software Reuse: Formal Foundations of Reuse and Domain Engineering (ICSR), pp. 150–159, 2009. 68

[Lankhorst 05] Marc Lankhorst. *ArchiMate: a Service-Oriented Enterprise Architecture Modeling Language*. In OMG Technical Meeting SOA WG, December 2005. 35

[Lankhorst 09] Proper H. Lankhorst M. & H. Jonkers. *The Architecture of the ArchiMate Language Enterprise*. In Proceedings of the EMMSAD 2009, held at CAiSE 2009, Amsterdam, the Netherlands, volume 29 of *LNBIP*, p. 367, Berlin, 2009. Springer. 35

[Law 07] A.M. Law. Simulation modeling and analysis. McGraw-Hill, 2007. 57

[Learner 11] Cambridge Advanced Learner. *collaborate. http://dictionary.cambridge.org/define.asp?key=14870&dict=CALD*, 2011. 58

## BIBLIOGRAPHY


[Lee 91] Jintae Lee. *Extending the Potts and Bruns model for recording design rationale.* In Proceedings of the 13th international conference on Software engineering (ICSE), pp. 114–125, Los Alamitos, CA, USA, 1991. 59

[Lehmann 09] A. Lehmann, T. Eichelmann, U. Trick, R. Lasch, B. Ricks & R. Tonjes. *TeamCom: A Service Creation Platform for Next Generation Networks.* In Proceedings of the 2009 Fourth International Conference on Internet and Web Applications and Services, pp. 12–17, 2009. 15

[Leist 06] Susanne Leist & Gregor Zellner. *Evaluation of current architecture frameworks.* In Proc. of the 2006 ACM symposium on Applied computing, SAC, pp. 1546–1553, 2006. 25

[Lennox 01] J. Lennox, H. Schulzrinne & J. Rosenberg. *Common Gateway Interface for SIP.* RFC 3050 (Informational), January 2001. 13

[Lennox 04] J. Lennox & H. Schulzrinne. *Call Processing Language (CPL): A Language for User Control of Internet Telephony Services.* Internet Engineering Task Force (IETF), 2004. 14

[Li 06a] W. D. Li, L. Ding & C. A. McMahon. Visualization models and technologies for collaborative product development: Status and promise, pp. 408–413. International Conference on Computer Supported Cooperative Work in Design. 2006. 58

[Li 06b] W. D. Li & Z. M. Qiu. *State-of-the-art technologies and methodologies for collaborative product development systems.* International Journal of Production Research, vol. 44, no. 13, pp. 2525–2559, 2006. 58

[Li 08] Jingyue Li, Reidar Conradi, Odd Petter Slyngstad, Marco Torchiano, Maurizio Morisio & Christian Bunse. *A State-of-the-Practice Survey of Risk Management in Development with Off-the-Shelf Software Components.* IEEE Trans. Softw. Eng., vol. 34, pp. 271–286, March 2008. 68

[MacLean 91] A. MacLean, R.M. Young, V.M.E. Bellotti & T.P. Moran. *Questions, options, and criteria: Elements of design space analysis.* Human-computer interaction, vol. 6, no. 3, pp. 201–250, 1991. 59

[MacLean 96] Allan MacLean, Richard M. Young, Victoria M. E. Bellotti & Thomas P. Moran. *Design rationale.* chapter Questions, options, and criteria: elements of design space analysis, pp. 53–105. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1996. xxx, 89

[Mahmood 06] Z Mahmood. *Frameworks and Tools for Building Enterprise Information Architectures.* In Proceedings of 6th Int. IBIMA Conf on managing Information In Digital Society, pp. 216–226, Bonn, Germany, 2006. 25

[Mannadiar 11] Raphael Mannadiar & Hans Vangheluwe. *Debugging in domain-specific modelling.* In Proceedings of the Third international conference on Software language engineering (SLE), pp. 276–285, 2011. 65

[Manuel (ed.) 11] Claire Manuel (ed.). *TM Forum Case Study Handbook 2012. Proving the value of TM Forum standards - 11 leading service providers share their succes stories.* 2011. 37

[Maximilien 08] E.M. Maximilien, A. Ranabahu & K. Gomadam. *An Online Platform for Web APIs and Service Mashups.* IEEE Internet Computing, vol. 12, no. 5, pp. 32 –43, sept.-oct. 2008. 17

[Mayer 03] Johannes Mayer, Ingo Melzer & Franz Schweiggert. *Lightweight Plug-In-Based Application Development.* In Intl Conf. NetObjectDays on Objects, Components, Architectures, Services, and Applications for a Networked World, pp. 87–102, London, UK, 2003. xxx, 84, 85

[Mens 06] Tom Mens & Pieter Van Gorp. *A Taxonomy of Model Transformation.* Electron. Notes Theor. Comput. Sci., vol. 152, pp. 125–142, March 2006. 41

[Mernik 05] Marjan Mernik, Jan Heering & Anthony M. Sloane. *When and how to develop domain-specific languages.* ACM Comput. Surv., vol. 37, no. 4, pp. 316–344, 2005. 56, 64

[Merriam-Webster 11] Merriam-Webster. *collaborate. http://www.merriam-webster.com/dictionary/collaborate*, 2011. 58

[Meyers 11] Bart Meyers & Hans Vangheluwe. *A framework for evolution of modelling languages.* Science of Computer Programming, vol. 76, no. 12, pp. 1223 – 1246, 2011. 65, 66

[Moghaddam 08] Mohammad R. Sadeghi Moghaddam, Ali Sharifi & Ehsan Merati. *Using Axiomatic Design in the Process of Enterprise Architecting.* In Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology - Volume 01, pp. 279–284, 2008. 36

[Mohr 02] M. Mohr & SA Russel. *North American product classification system: Concepts and process of identifying service products.* In Proceedings of the 17th Annual Meeting of the Voorburg Group on Service Statistics., Nantes, France, 2002. xxxiv

[Morisio 02] M. Morisio, C. B. Seaman, V. R. Basili, A. T. Parra, S. E. Kraft & S. E. Condon. *COTS-based software development: processes and open issues.* J. Syst. Softw., vol. 61, pp. 189–189, April 2002. 68

[Mullery 99] Al Mullery, Anders Olsen, Didier Demany, Elsa Cardoso, Fiona Lodge, Mario Kolberg, Morgan Bjrkander & Richard Sinnott. *The Pros and Cons of Using SDL for Creation of Distributed Services.* In Mario Campolargo & Jaime Delgado, editors, Intelligence in Services and Networks Paving the Way for an Open Service Market, volume 1597 of *LNCS*, pp. 342–354. 1999. 14

[Myers 04] G.J. Myers & C. Sandler. *The art of software testing - 2nd edition*, 2004. 57

[Ncube 08] Cornelius Ncube, Patricia Oberndorf & Anatol W. Kark. *Opportunistic Software Systems Development: Making Systems from What's Available.* IEEE Softw., vol. 25, pp. 38–41, November 2008. 68

[Niemi 06] E. Niemi. *Enterprise architecture benefits: Perceptions from literature and practice.* Internet & Information Systems in the Digital Age: Challenges and Solutions, pp. 161–168, 2006. 24

[Niemöller 09] Jörg Niemöller, Roman Levenshteyn, Eugen Freiter, Konstantinos Vandikas, Raphaël Quinet & Ioannis Fikouras. *Aspect Orientation for Composite Services in the Telecommunication Domain.* In Proceedings of the 7th International Joint Conference on Service-Oriented Computing (ICSOC-ServiceWave), pp. 19–33, 2009. 17

[Ober 08] Ileana Ober, Ali Abou Dib, Louis Féraud & Christian Percebois. *Towards Interoperability in Component Based Development with a Family of DSLs.* In Proc. of the 2nd Europ. conf. on Software Architecture (ECSA), pp. 148–163, Paphos, Cyprus, 2008. 56

[Ohnishi 07] H. Ohnishi, Y. Yamato, M. Kaneko, T. Moriya, M. Hirano & H. Sunaga. *Service Delivery Platform for Telecom-Enterprise-Internet Combined Services.* In IEEE Global Telecommunications Conference (GLOBECOM), pp. 108 –112, nov. 2007. 10

[OMA 07] OMA. *Open Mobile Alliance Service Environment*, 2007. 14

[OMA 11] OMA. *Open Mobile Alliance Website*, 2011. 12

[OMG 03] OMG. *MDA Guide Version 1.0.1.* `http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf`, June 2003. 43

[OMG 10a] OMG. *Meta Object Facility (MOF) Core Specification. V. 2.4*, 2010. 44

[OMG 10b] OMG. *Object Constraint Language. Version 2.3*, 2010. 44

[OMG 11a] OMG. *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Version 1.1*, 2011. 44

[OMG 11b] OMG. *OMG Unified Modeling Language (OMG UML) Superstructure. Version 2.4*, 2011. 44

## BIBLIOGRAPHY


[Ou 06] Shumao Ou, Nektarios Georgalas, Manooch Azmoodeh, Kun Yang & Xiantang Sun. *A Model Driven Integration Architecture for Ontology-Based Context Modelling and Context-Aware Application Development.* In Arend Rensink & Jos Warmer, editors, Model Driven Architecture - Foundations and Applications, volume 4066 of *LNCS*, pp. 188–197. 2006. 46

[Pal 05] N. Pal & R. Zimmerie. *Service innovation: a framework for success.* White Paper, eBusiness Research Center, Smeal College of Business, Pennsylvania State University, University Park, PA, pp. 1–32, 2005. xxxiv

[Pan 08] P. Pan, L. Jin, C. Ying & J. H. Liu. Template based rapid service creation environment for service delivery platform, pp. 534–545. IEEE IFIP Network Operations and Management Symposium. 2008. xi, xxxvi, 3, 6, 7

[Parreiras 07] Fernando Silva Parreiras, Steffen Staab & Andreas Winter. *On marrying ontological and metamodeling technical spaces.* In ESEC-FSE companion '07: The 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, pp. 439–448, 2007. 71

[Paulisch 93] Frances Newbery Paulisch. The Design of an Extendible Graph Editor. Springer-Verlag, Secaucus, NJ, USA, 1993. 85

[Pavan 07] Kumar Pavan. *Relevance of SDK to foster IMS application development.* In International Conference on IP Multimedia Subsystem Architecture and Applications, pp. 1–4, 2007. xiii, 9

[Pérez-Medina 07] Jorge-Luis Pérez-Medina, Sophie Dupuy-Chessa & Agnès Front. *A survey of model driven engineering tools for user interface design.* In Proceedings of the 6th international conference on Task models and diagrams for user interface design (TAMODIA), pp. 84–97, 2007. 44

[Peristeras 06] V. Peristeras & K. Tarabanis. *The Connection, Communication, Consolidation, Collaboration Interoperability Framework (C4IF) For Information Systems Interoperability.* Intl. Journal of Interoperability in Business Information Systems, vol. 1, no. 1, pp. 61–72, 2006. 60, 61

[Phillips 07] Chris Phillips. *A Review of High Performance Simulation Tools and Modeling Concepts.* In A. Nejat Ince & Arnold Bragg, editors, Recent Advances in Modeling and Simulation Tools for Communication Networks and Services, pp. 29–47. Springer, 2007. 87

[Pollet 06] Thierry Pollet, Gerard Maas, Johan Marien & Albert Wambecq. *Telecom Services Delivery in a SOA.* In Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA), volume 2, pp. 529–533, 2006. xiv, 10

[PragmaDev 11] PragmaDev. *Real Time Developer Studio. http://www.pragmadev.com/product/modeling.html*, 2011. 14

[Prechelt 11] Lutz Prechelt & Karl Beecher. *Four Generic Issues for Tools-as-Plugins Illustrated by the Distributed Editor Saros.* In Proc. of the 1st Ws. on Developing Tools as Plug-ins (TOPI) at ICSE, 2011. xviii, 62, 113

[Ramsin 08] Raman Ramsin & Richard F. Paige. *Process-centered review of object oriented software development methodologies.* ACM Comput. Surv., vol. 40, pp. 3:1–3:89, February 2008. 51, 52

[Reed 11] Rick Reed. *SDL-2010: Background, Rationale, and Survey.* In Iulian Ober & Ileana Ober, editors, SDL 2011: Integrating System and Software Modeling, volume 7083 of *Lecture Notes in Computer Science*, pp. 4–25. Springer Berlin / Heidelberg, 2011. 14

[Ren 08] L. Ren, Jia Jia Wen, Qi Yu & L. Longmore. *SIP application composition framework based on business rules.* In IEEE Network Operations and Management Symposium (NOMS), pp. 823 –826, april 2008. 15

[Renger 08a] Michiel Renger, Gwendolyn L. Kolfschoten & Gert-Jan Vreede. *Challenges in Collaborative Modeling: A Literature Review*. In Wil Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, Clemens Szyperski, Jan L. G. Dietz, Antonia Albani & Joseph Barjis, editors, Advances in Enterprise Engineering I, volume 10 of *LNBIP*, pp. 61–77. 2008. 58

[Renger 08b] Michiel Renger, Gwendolyn L. Kolfschoten & Gert-Jan Vreede. *Groupware: Design, Implementation, and Use*. chapter Using Interactive Whiteboard Technology to Support Collaborative Modeling, pp. 356–363. 2008. 62

[Rosenberg 02] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley & E. Schooler. *SIP: session initiation protocol*. Internet Engineering Task Force (IETF): RFC 3261, 2002. 11

[Rosenberg 08] Florian Rosenberg, Francisco Curbera, Matthew J. Duftler & Rania Khalaf. *Composing RESTful Services and Collaborative Workflows: A Lightweight Approach*. IEEE Internet Computing, vol. 12, pp. 24–31, 2008. 17

[Rothenberg 89] Jeff Rothenberg. *The Nature of Modeling*. In L.E. William, K.A. Loparo & N.R. Nelson, editors, Artificial Intelligence, Simulation, and Modeling, pp. 75–92. New York, John Wiley and Sons, Inc., 1989. 39

[Rubel 06] Dan Rubel. *The Heart of Eclipse*. Queue, vol. 4, pp. 36–44, 2006. 86

[Sabbouh 07] Marwan Sabbouh, Jeff Higginson, Salim Semy & Danny Gagne. *Web mashup scripting language*. In Proceedings of the 16th international conference on World Wide Web (WWW), pp. 1305–1306, 2007. 17

[Sarkar 11] N.I. Sarkar & S.A. Halim. *A Review of Simulation of Telecommunication Networks: Simulators, Classification, Comparison, Methodologies, and Recommendations*. Cyber Journals: Multidisciplinary Journals in Science and Technology-Journal of Selected Areas in Telecommunications (JSAT), 02 (03), 10-17., 2011. 57, 87

[Schobbens 06] Pierre-Yves Schobbens, Patrick Heymans & Jean-Christophe Trigaux. *Feature Diagrams: A Survey and a Formal Semantics*. pp. 136–145, 2006. 64

[Schoenherr 09] Marten Schoenherr. *Towards a Common Terminology in the Discipline of Enterprise Architecture*. In George Feuerlicht & Winfried Lamersdorf, editors, Service-Oriented Computing — ICSOC Ws., pp. 400–413, 2009. 24

[Sedigh-Ali 01] Sahra Sedigh-Ali, Arif Ghafoor & Raymond A. Paul. *Software Engineering Metrics for COTS-Based Systems*. Computer, vol. 34, pp. 44–50, 2001. 68

[Sessions 07] Roger Sessions. *Comparison of the Top Four Enterprise Architecture Methodologies*. Technical report, ObjectWatch, Inc., May 2007. 26

[Shafique 10] Muhammad Shafique & Yvan Labiche. *A Systematic Review of Model Based Testing Tool Support*. Technical report, 2010. 47

[Shen 08] W. M. Shen, Q. Hao & W. D. Li. *Computer supported collaborative design: Retrospective and perspective*. Computers in Industry, vol. 59, no. 9, pp. 855–862, 2008. 58

[Sienel 09] Jürgen Sienel, Alberto León Martín, Carlos Baladrón Zorita, Laurent-Walter Goix, Álvaro Martínez Reol & Belén Carro Martínez. *OPUCE: A telco-driven service mash-up approach*. Bell Lab. Tech. J., vol. 14, pp. 203–218, May 2009. 17

[Simonin 07] J. Simonin, Y. Le Traon & J. M. Jezequel. *An enterprise architecture alignment measure for telecom service development*. 11th Ieee International Enterprise Distributed Object Computing Conference, pp. 476–483, 2007. 36

[Simonin 08] J. Simonin, F. Alizon, J.-P. Deschrevel, Y. Le Traon, J.-M. Jezequel & B. Nicolas. *EA4UP: An Enterprise Architecture-Assisted Telecom Service Development Method*. In 12th International IEEE Enterprise Distributed Object Computing Conference (EDOC), pp. 279 –285, 2008. 36

## BIBLIOGRAPHY


[Simonin 10] J. Simonin, E. Bertin, Y.L. Traon, J.-M. Jezequel & N. Crespi. *Business and Information System Alignment: A Formal Solution for Telecom Services.* In Software Engineering Advances (ICSEA), 2010 Fifth International Conference on, pp. 278 –283, aug. 2010. 36

[Simonin 11] Jacques Simonin, Emmanuel Bertin, Yves Le Traon, Jean-Marc Jezequel & Noel Crespi. *Analysis and improvement of the alignment between business and information system for telecom services.* International Journal On Advances in Software, vol. 4, no. 1, pp. 117–128, 2011. 25, 95

[Simos 95] Mark A. Simos. *Organization domain modeling (ODM): formalizing the core domain modeling life cycle.* pp. 196–205, 1995. 64

[Sinnott 99] Richard O. Sinnott & Mario Kolberg. *Engineering Telecommunication Services With SDL.* In Proceedings of the IFIP TC6/WG6.1 Third International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS), pp. 187–204, 1999. 14

[Spinellis 01] Diomidis Spinellis. *Notable design patterns for domain-specific languages.* J. Syst. Softw., vol. 56, pp. 91–99, February 2001. 56

[Spohrer 05] J. Spohrer. *Services sciences, management, engineering: a next frontier in education, innovation and economic growth. Presentation.* Technical report, Services Marketing Workshop, Center for Services Leadership, Arizona State University, Tempe AZ, 2005. xxxiv

[Spohrer 08] J. Spohrer & P.P. Maglio. *The Emergence of Service Science: Toward Systematic Service Innovations to Accelerate Co-Creation of Value.* Production and Operations Management, vol. 17, no. 3, pp. 238–246, 2008. xxxiv

[Sprinkle 09] Jonathan Sprinkle, Marjan Mernik, Juha-Pekka Tolvanen & Diomidis Spinellis. *Guest Editors' Introduction: What Kinds of Nails Need a Domain-Specific Hammer?* IEEE Software, vol. 26, pp. 15–18, 2009. 64

[Steen 04] M. W. A. Steen, D. H. Akehurst, H. W. L. ter Doest & M. M. Lankhorst. *Supporting Viewpoint-Oriented Enterprise Architecture.* In Proceedings of the 8th IEEE International Enterprise Distributed Object Computing Conference (EDOC), pp. 201–211, 2004. xxix, 35, 36

[Steffen 07] Bernhard Steffen, Tiziana Margaria, Ralf Nagel, Sven Jörges & Christian Kubczak. *Model-driven development with the jABC.* In Proceedings of the 2nd international Haifa verification conference on Hardware and software, verification and testing (HVC), pp. 92–108, 2007. 46

[Strembeck 09] Mark Strembeck & Uwe Zdun. *An approach for the systematic development of domain-specific languages.* Softw. Pract. Exper., vol. 39, pp. 1253–1292, 2009. 64, 65

[Study Group XVIII 10] Study Group XVIII. *Principles of Intelligent Network Architecture. ITU-T Recommendation Q.1201*, October 1992. Available at http://electronics.ihs.com/, accessed 24th February 2010. 10

[Taylor 08] Steve Taylor & Larry Hettick. *AT&T moves ahead with IMS, unveils VoIP service for its IPTV customers*, 2008. http://www.networkworld.com/newsletters/2008/0128converge2.html. 12

[Terrasse 06] Marie-Noölle Terrasse, Marinette Savonnet, Eric Leclercq, Thierry Grison & George Becker. *Do we need metamodels AND ontologies for engineering platforms?* In Proc. of the Intl Wksh on Global integrated Model Management (GaMMa), pp. 21–28, 2006. 70

[The Open Group 07] The Open Group. *TOGAF Version 8.1.1*, 2007. 28

[The Open Group 09a] The Open Group. *ArchiMate 1.0 Specification*, 2009. xxix, 25, 27, 29, 30, 31, 32, 33, 34, 35, 78, 98

[The Open Group 09b] The Open Group. *TOGAF Version 9*, 2009. 26, 28

[TINA-C 97] TINA-C. *Definition of Service Architecture. Version 5.0*, 1997. 3, 4, 7, 110

[Tisi 09] Massimo Tisi, Frédéric Jouault, Piero Fraternali, Stefano Ceri & Jean Bézivin. *On the Use of Higher-Order Model Transformations.* LNCS, vol. 5562, pp. 18–33, 2009. 41

[TM Forum 09] TM Forum. *TR144, Tooling Environment - Requirements and Description, Release 1.0.* Technical report, TM Forum, 2009. 35, 36

[TM Forum 10] TM Forum. *Frameworx Introduction. Enabling Successful Business Transformation.* Technical report, TM Forum, 2010. 6, 26

[TOGAF and TM Forum 11] TOGAF and TM Forum. *Mapping of TOGAF and Frameworx Solutions Business Process Framework (eTOM), Information Framework (SID) and Application Framework (TAM). Version 1.0.* Technical report, Industry Group Liaison TOGAF and TM Forum Frameworx Collaboration Project, 2011. xxix, 26, 28, 29

[Torchiano 04] Marco Torchiano & Maurizio Morisio. *Overlooked Aspects of COTS-Based Development.* IEEE Softw., vol. 21, pp. 88–93, March 2004. 68

[Troche 06] Carlos Troche. *Documenting Complex Systems in the Enterprise.* In Intl. conf. on Complex Systems, 2006. 24

[Trossen 98] Dirk Trossen & Karl-Heinz Scharer. *CCS: CORBA-Based Conferencing Service.* In Proceedings of the 5th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services, pp. 71–76, London, UK, 1998. Springer-Verlag. xxx, 109

[Umali 02] C.L. Umali. *Asian telecommunications: market deregulation and competition.* International Journal of Management and Decision Making, vol. 3, no. 3, pp. 256–279, 2002. xxxvi

[Urbaczewski 06] L. Urbaczewski & S. Mrdalj. *A comparison of enterprise architecture frameworks.* Issues in Information Systems, vol. 7, no. 2, pp. 18–23, 2006. 25, 28

[Vallecillo 10] Antonio Vallecillo. *On the Combination of Domain Specific Modeling Languages.* In Proceedings of the 6th European Conference on Modelling Foundations and Applications (ECMFA), volume 6138 of *LNCS*, Paris, France, 2010. 61, 69, 70, 76

[van Deursen 98] Arie van Deursen & Paul Klint. *Little languages: little maintenance.* Journal of Software Maintenance, vol. 10, pp. 75–92, March 1998. 56

[Wagner 07] Stefan Wagner, Stephan Winkler, Erik Pitzer, Gabriel Kronberger, Andreas Beham, Roland Braune & Michael Affenzeller. *Benefits of plugin-based heuristic optimization software systems.* In Proc. of the 11th intl conf. on Computer aided systems theory, pp. 747–754, Las Palmas de Gran Canaria, Spain, 2007. 85

[Wang 02] L. H. Wang, W. M. Shen, H. Xie, J. Neelamkavil & A. Pardasani. *Collaborative conceptual design - state of the art and future trends.* Computer-Aided Design, vol. 34, no. 13, pp. 981–996, 2002. 58

[Welty 03] C. Welty. *Ontology research.* AI Magazine, vol. 24, no. 3, pp. 11–12, 2003. 70

[White 09] Jules White, James H. Hill, Jeff Gray, Sumant Tambe, Aniruddha S. Gokhale & Douglas C. Schmidt. *Improving Domain-Specific Language Reuse with Software Product Line Techniques.* IEEE Softw., vol. 26, pp. 47–53, July 2009. 65

[Wimmer 11] Manuel Wimmer, Andrea Schauerhuber, Gerti Kappel, Werner Retschitzegger, Wieland Schwinger & Elizabeth Kapsammer. *A survey on UML-based aspect-oriented design modeling.* ACM Comput. Surv., vol. 43, pp. 28:1–28:33, October 2011. 43

[Wolf 07] T. Wolf. *Rationale-based unified software engineering model.* PhD thesis, TU Munchen, Germany, 2007. 59

## BIBLIOGRAPHY


[Wolf 08] Timo Wolf. Rationale-based unified software engineering model. VDM Verlag, Germany, 2008. 59, 89, 90

[Woodside 09] Murray Woodside, Dorina C. Petriu, Dorin B. Petriu, Jing Xu, Tauseef Israr, Geri Georg, Robert France, James M. Bieman, Siv Hilde Houmb & Jan Jürjens. *Performance analysis of security aspects by weaving scenarios extracted from UML models.* J. Syst. Softw., vol. 82, pp. 56–74, 2009. 43

[Wu 09] Hui Wu, Jeff Gray & Marjan Mernik. *Unit Testing for Domain-Specific Languages.* In Proceedings of the IFIP TC 2 Working Conference on Domain-Specific Languages (DSL), pp. 125–147, 2009. 65

[Yang 05] K. Yang, S. Ou, M. Azmoodeh & N. Georgalas. *Policy-based model-driven engineering of pervasive services and the associated OSS.* BT Technology Journal, vol. 23, pp. 162–174, July 2005. 46

[Yang 06a] K. Yang, S. Ou, M. Azmoodeh & N. Georgalas. *Model-based service discovery – prototyping experience of an OSS scenario.* BT Technology Journal, vol. 24, pp. 145–150, 2006. 45

[Yang 06b] Kun Yang, Chris Todd & Shumao Ou. *Model-based service discovery for future generation mobile systems.* In Proceedings of the 2006 international conference on Wireless communications and mobile computing (IWCMC), pp. 973–978, 2006. 45

[Yang 07] Kun Yang, J. Wittgreffe & M. Azmoodeh. *Policy-based model-driven creation of adaptive services in wireless environments.* Vehicular Technology Magazine, IEEE, vol. 2, no. 3, pp. 14 –20, sept. 2007. 46

[Yelmo 07] Juan Yelmo, Ruben Trapero, Jose del Alamo, Juergen Sienel, Marc Drewniok, Isabel Ordas & Kathleen McCallum. *User-Driven Service Lifecycle Management - Adopting Internet Paradigms in Telecom Services.* In Bernd Kramer, Kwei-Jay Lin & Priya Narasimhan, editors, Service-Oriented Computing - ICSOC, volume 4749 of *LNCS*, pp. 342–352. 2007. 17

[Yelmo 08] J. C. Yelmo, J. M. del Alamo, R. Trapero, P. Falcarin, J. Yu, B. Carro, C. Baladron & Itu. A user-centric service creation approach for Next Generation Networks. Proceedings of the First ITU-T Kaleidoscope Academic Conference Innovations in NGN: Future Network and Services. Int Telecommunication Union, 2008. xiv, 10, 11, 15, 17

[Yelmo 11] Juan C. Yelmo, Jose M. del Alamo, Ruben Trapero & Yod-Samuel Martin. *A user-centric approach to service creation and delivery over next generation networks.* Computer Communications, vol. 34, no. 2, pp. 209 – 222, 2011. Special Issue: Open network service technologies and applications. 16, 17

[Zhang 04] Yingzhou Zhang & Baowen Xu. *A survey of semantic description frameworks for programming languages.* ACM SIGPLAN Notices, vol. 39, no. 3, pp. 14–30, March 2004. 42

[Zoric 11] Josip Zoric & Rolv Braek. *Scenario based techno-business analysis of service platforms and their service portfolios.* Telecommunication Systems, vol. 46, pp. 95–116, 2011. 36

Technopole Brest-Iroise - CS 83818
29238 Brest Cedex 3
France
Tel : + 33 (0)2 29 00 11 11
www.telecom-bretagne.eu