# Meta-tools for software language engineering : a flexible collaborative modeling language for efficient telecommunications service design

Vanea Chiprianov, Yvon Kermarrec, Siegfried Rouvrais

# Meta-tools for Software Language Engineering: A Flexible Collaborative Modeling Language for Efficient Telecommunications Service Design

Vanea Chiprianov, Yvon Kermarrec
Institut Telecom, Telecom Bretagne, UMR CNRS 3192 Lab-STICC, Technopole Brest Iroise, CS 83818 29238, Brest Cedex 3, France
vanea.chiprianov@telecom-bretagne.eu

Siegfried Rouvrais
Institut Telecom, Telecom Bretagne, Technopole Brest Iroise, CS 83818 29238, Brest Cedex 3, France
siegfried.rouvrais@telecom-bretagne.eu

## 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.

## Keywords

Service design, DSML, view, meta-model, meta-tool, MDE

## 1. KEY ISSUES OF TELECOM SERVICES

Imagine yourself in the comfort of your hotel conference room, in, say, Cape Town. As a service customer, your hotel offers you, as an end-user, Internet access and other telecom services through a solution and service provider, e.g. BT or NeoTel. You can work securely and collaboratively, through a Virtual Private Network (VPN) [19], at your newly funded project with your team based at, say, Paris, and your academic or industrial partners from e.g. Romania, New York and Tokyo. You should have access at your required documents and services even if, say, there were a storm in New York and the VPN server was temporary brought down. This depends on the quality of service the VPN provider offers. At the same time, you are talking, using a Voice over IP (VoIP) phone system, e.g. Skype, with your life partner, which is in the city center and tries to decide for which June 2010 World Cup football match he/she should buy tickets. While doing all these seemingly simple and fairly common activities, you are transparently using telecom services, implemented on computer networks managed by, say, Cisco.

End-users or customers, however, do not need to be aware how these services work. A telecom service becomes more and more complex, growing until mature enough to be adopted. To date, the majority of service designers capture its functional and non-functional requirements in natural language descriptions and free-form diagrams. Most often, network designers have to implement its design as a distributed set of components. Finally, network operators select, configure and deploy the physical entities and protocols of the service.

This method, proposed by the ITU-T Intelligent Network Conceptual Model (INCM) [21], still presents several problems, such as the long concept-to-market time of new services. It is mainly influenced by the following factors:

1) The *low reusability rate* of documents and diagrams produced by service designers;

2) Numerous *errors* introduced by the *ambiguity* of *natural language* in specifications;

3) The difficulty of *communicating* and remembering about *design decisions* taken collaboratively;

4) The *diversity* and specificity of *types of jobs, viewpoints* involved in the method.

The concept-to-market time is also affected by properties of services themselves, like:

a) The augmenting number of services being deployed on the same platform increases the probability of *undesired (feature) interactions* and incoherence;

b) The variety of *heterogeneous technologies* used to implement a service introduces interoperability concerns;

c) Increasing expectations of users induce a *higher* degree of service creation *complexity*.

The INCM is just a framework of thought. It is not an architecture, it does not provide tools, or means to check, validate, or ensure coherence. We argue that one way to reduce


*ICSE 2010 Workshop on Flexible Modeling Tools* Cape Town, South Africa

the concept-to-market time is by building flexible collaborative domain specific modeling languages (DSMLs) for each viewpoint involved in the creation of a telecom service, as identified in the INCM. Using model-based meta-tools, that rapidly and iteratively generate DSML associated tools, will allow more rapid and iterative prototyping and integration of end-users' feedback.

Using the INCM as a guide, we propose some of these tools as part of our approach. As our work has begun in the context of an industrial project with a major telecom service provider, we focus on one specific view, the one for service designers, and its interactions with adjacent views in the INCM. In the remaining of this position paper, we present our INCM case study in Section 2. We refine the problem identified in this introduction section into four issues and propose roadmaps to each of them, indicating partial results, in Section 3. We synthesize our proposed solutions into a larger method in Section 4 and conclude on future work in the last section.

# 2. CASE STUDY: THE INTELLIGENT NETWORK CONCEPTUAL MODEL

To address the increasing degree of complexity of telecom services, the ITU-T has introduced the Intelligent Network Conceptual Model (INCM) [21], as 'a framework for the design and description of the IN architecture'. It consists of four 'planes', or viewpoints [16], each refining the service definition from the previous, upper-level plane and defining a specific and dedicated type of job:

- The *Service Plane* describes a service from the customer's high level point of view, in natural language. It contains no information about the implementation;
- The *Global Functional Plane* (GFP) models a service as a chain of Service Independent Building Blocks (SIB), eventually using a formal method. A chain of SIBs constitutes the global service logic. It is the responsibility of the service designer;
- The *Distributed Functional Plane* (DFP) describes the software implementation of the service as a distributed set of functional entities. Each SIB is constituted from a sequence of actions executed in functional entities. It is the responsibility of the network designer;
- The low-level *Physical Plane* describes the implementation of the DFP on distributed physical machines. It identifies the physical entities, protocols and resources. It is the responsibility of equipment suppliers and network operators.

We propose to refine the long concept-to-market time problem, identified in the previous section, into four issues, in the case of the INCM:

I) Problems 4 and b have as central point heterogeneity. We regroup them into *defining a service in such a manner as to allow separation between any of the four planes of INCM*, thus ensuring separation between heterogeneous technologies and types of jobs.

II) Problems 1, 2 and c are caused by the lack of domain specificity, low formality and automation. Therefore, we regroup them into *offering to each viewpoint a computer tool specific to its domain.*

III) Problem 3 is related to *capturing and using team communication and decisions about artifacts*; this is a concern transversal to all planes of INCM.

IV) Previous issue I introduces a complementary one. It is not enough to provide a separation between planes, at the same time, this separation must *ensure that the artifacts produced at adjacent planes are coherent, consistent, compatible and interoperable.*

# 3. HOW TO REDUCE THE CONCEPT-TO-MARKET SERVICE TIME?

## 3.1 Separating the GFP from the DFP

To follow the INCM recommendations, we defend the separation between the artifacts defined at adjacent planes. As we focus on the GFP, this becomes a need to model the SIBs at the GFP separately from their implementation at the DFP. However, means to ensure the coherence of artifacts from the two planes are also needed.

Our research question for issue I thus becomes: **How to define the Global Functional Plane, so as to allow both independence and communication, between itself and its adjacent (Service and Distributed Functional) Planes?**

We contend that defining the GFP of a telecom service using the Platform Independent Models (PIMs) from the Model Driven Engineering (MDE) approach enables the independence of the GFP artifacts from the DFP ones. As well, it permits a relation easier to maintain, more automatized and coherent, with the artifacts at the DFP, implemented as the Platform Specific Models (PSMs) from the MDE approach.

We have already implemented the artifacts from the GFP as PIMs, for the case of a VPN [4]. Figure 1 presents a VPN example defined at the GFP and the DFP. We design the artifacts at the GFP using a domain specific modeling language (Section 3.2). As discussed in more detail in Section 3.4, we will require a second modeling language to create the PSMs of the DFP. We haven't yet defined this second language and that is why the example at the DFP in Figure 1 is only illustrative, inspired by Cisco. To date, we generate Smalltalk code instead of PSMs, using OpenArchitectureWare [9] as a meta-tool.

## 3.2 Defining the Flexible Domain Specific Modeling Language for Service Design

In the INCM, each plane has a vocabulary, a language, partially common with that of another viewpoint, partially specific. Consequently, our research question for issue II becomes: **How to define a modeling language from the existing language of a particular viewpoint, such that it facilitates its adoption by the professionals doing that type of job?**

A Domain Specific Modeling Language (DSML) [6] is defined in such a way that integrates the vocabulary specific to a viewpoint, to a domain. We argue that defining a DSML, using model-based meta-tools, facilitates its adoption. Meta-tools enable rapid and interactive prototyping and thus rapid integration of feedback from designers.

We have implemented a DSML prototype for designing VPNs at the GFP [3]. We chose to implement it as a visual modeling language to lower the difficulty of service designers

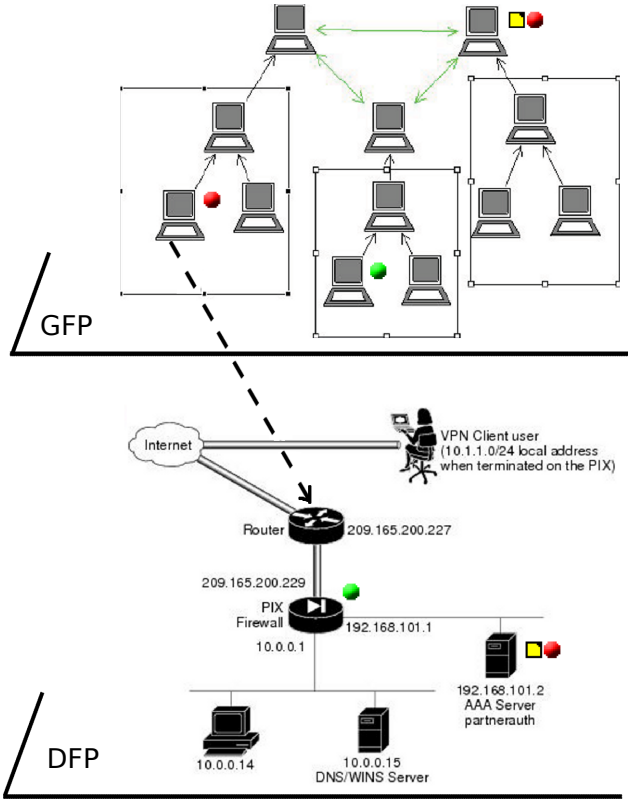


**Figure 1: The GFP implemented with PIMs and the DFP implemented with PSMs.**

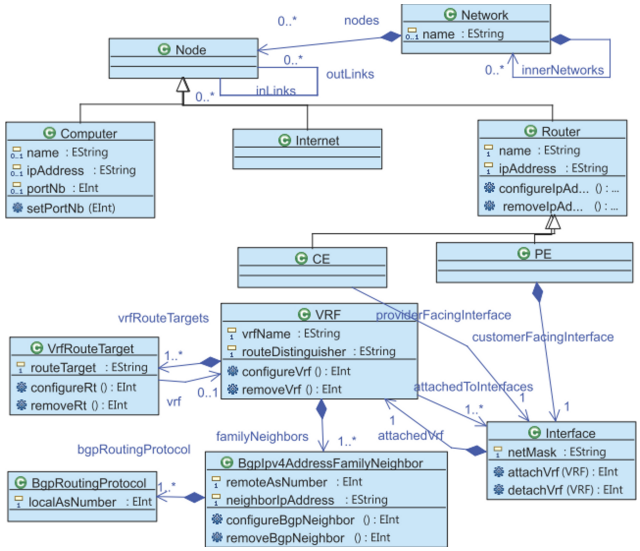


**Figure 2: Abstract syntax of the DSML prototype for VPN design.**

to create domain specific entities [15], i.e. services. We use a model-based approach, in which the abstract syntax is defined by means of a meta-model (MM), presented in Figure 2. Starting from this MM, we generated a visual editor, that also provides the possibility of specifying OCL constraint rules, using TOPCASED [8] as a meta-tool. Using this editor, service designers can select the entities specific to a VPN and connect them by predefined relationships. Although the designers have a high degree of liberty, they must pay attention to the semantics of each entity and relationship, as this will be used to generate PSMs at the DFP.

We expect our DSML to offer, "through appropriate notations and abstractions, expressive power focused on [...] a particular problem domain" [6]. Due to its ease of use by service designers, its increase in reusability (i.e. the modeled entities conserve domain knowledge), its power of artifact generation and of checking, the DSML will determine a productivity increase. A DSML also introduces more formality than natural language descriptions and free-form diagrams. This formality enables domain-specific verifications and assistance, thus reducing the number of errors. However, in the exploratory phases of service design, it is important that techniques and associated tools allow designers (collaborative) creative freedom. Therefore, a compromise between the creative freedom and the degree of formality is needed. Consequently, the DSML for service designers is a flexible modeling tool, a blend between the creative liberty offered to designers and the formal semantics of the modeled entities.

However, the design, implementation and maintenance of a DSML and its associated tools is costly. To reduce this cost, we use model-based meta-tools, that enable iterative definition of the DSML and rapid generation of associated and specific tools. Nevertheless, the complexity of the language tools we can generate using this approach is limited by the meta-tools, which are not yet in their maturity (cf. the TOPCASED limitation in [3]).

## 3.3 Enhancing a DSML with Collaborative Capabilities

As part of their collaborative activities, professionals need to communicate and especially provide justifications about the decisions they have made. For example, a study [13] on the information needs of software developers finds that the most absent information is about design and program behavior. Dutoit et al. [7] define *Design Rationale* (DR) as 'the justification behind decisions', as 'the reasoning that goes into determining the design of the artifact'.

To facilitate the interactions between designers and supply the information they need, we propose to offer them the possibility of specifying the rationale behind their decisions and adding annotations with social tagging [20]. This is a concern that appears at each plane of the INCM, transversal to all planes. As previously, we restrict here to the GFP.

Therefore, the research question associated with the issue III becomes: **How to include annotations and rationale about designed artifacts in the service designer's DSML? How are designers influenced by rationale in their reuse of designed artifacts?**

We propose to integrate a MM for annotations and rationale in the MM defining the abstract syntax of the DSML for service designers, in order to increase the reusability of designed artifacts.

Dutoit et al. [7] argue DR supports collaboration, by: promoting coordination in design teams, exposing differing points of view, facilitating participation and collaboration and building consensus. They also indicate other uses of DR, related to supporting knowledge transfer, among which reuse [17]. Consequently, we expect that integrating DR into our approach will increase the reusability of SIBs, designers

will capitalize on past experience, thus reducing the design time and the concept-to-market time of new services.

Moreover, integrating the DR MM into the abstract syntax MM means that the DR tools, that will be generated from the integrated MM, will be thus integrated with the language tools. We hope this will stimulate designers to capture DR [2]. For example, Figure 1 presents collaboration indicators, inspired from [23], annotating the entities. We use a red ball for open and a green ball for closed issues, and yellow notes for comments. However, adapting existing DR representation schema into MMs involves making choices: some concepts from the schema will not appear in the MM, some relations from the schema cannot be modeled in the MM because of the limitations of the meta-MM to which the MM conforms.

### 3.4 Providing Syntactic and Semantic Interoperability between Two DSMLs

The approach proposed by INCM implies a refinement process in four viewpoints and three refinements. Because it is a refinement process, one may think that the language for a viewpoint that is farther in the refinement process includes the entire language of a viewpoint that is closer in the process. This is not the case: while there is a common set of entities between, for example, the language of the service designer and that of the network designer, each of them has its own, specific vocabulary.

Thus, the research question related to issue IV becomes: **How can one ensure syntactic and semantic interoperability between the artifacts defined with the DSMLs for the GFP and the DFP respectively?**

We investigate capturing the common vocabulary between the DSMLs for the GFP and DFP, using model transformations enhanced with ontologies. This ensures syntactic and respectively semantic interoperability between artifacts conceived with the two DSMLs and between their associated tools, in a manner that is flexible to change and allows coherency analysis between views.

## 4. META-TOOLS FOR LANGUAGE ENGINEERING

We have generated a *visual editor* for the service designers' DSML. Using the same approach, we will generate a second editor for the network designers' DSL. Between these two DSLs, we will define a model transformation enriched with ontologies. We argue that, if software compilers, that translate from a high level of abstraction language into machine language, are considered tools, any other means of implementing language translations be considered tools as well. Consequently, *model transformations* may be seen as tools (in [5], 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 may be seen as a tool. In the real-world refinement process of the INCM, there is also a need to generate code from certain entities defined using a DSL, into general purpose programming languages. That is why we include in our approach *code generation* as a tool as well.

The general accepted meaning of a meta-tool is a tool that allows specification and generation of another tool. For example, in [18] is proposed a semantically configurable code-generator generator (CGG) that creates Java code generators, or in [11] is presented Marama, a suite of meta-tools that permits rapid specification of notational elements, meta-models, view editors and view-model mappings, and that provides model transformation and code generation support.

The main advantages of using meta-tools are:

- The rapid building and low cost of tools;
- The simple changing process, that consists in updating only the specification, i.e. either MM, or model transformation, or template.

So a meta-tool based approach is well suited for an iterative approach of telecom service creation. However, meta-tools also have disadvantages:

- When using several meta-tools as a chain, inputing the output of one into another, we felt the need for a framework;
- We would like to define multiple views on a MM, for example, work only with the DR part of the MM, and then integrate it into the main MM;
- We would also find useful the possibility to integrate a part of the MM (e.g. the DR MM) into MMs of several DSLs, in a cross-cutting (AOP-like) manner [22];
- As meta-tools are generic, they sometimes have poor configuration power, and the generated tools are not as specific as we would like them to be.

As meta-tools to specify, generate and execute the tools necessary to our approach, we are investigating: TOPCASED to generate visual editors for the DSLs, ATL [12] to specify and execute model transformations, OpenArchitectureWare to specify and execute template-based code generation.

## 5. FUTURE WORK

In this position paper, we focused on the Telecommunications service domain, via the INCM, to defend tool interoperability and meta-modeling and to instantiate them more rapidly and efficiently for designers. We identified several factors that induce the long concept-to-market time of new telecommunications services. We refined and grouped them into four issues. To tackle them in an integrative manner, we defended using model-based meta-tools to specify and generate tools for DSLs. As our case study is the viewpoint of service designers in the INCM, we implemented a flexible prototype for their domain specific modeling language.

Our plan for the future consists in addressing:

- The interaction between the Global Functional Plane and the Service Plane. Because the customer is present at the Service Plane, an even higher flexibility and liberty of language and tools are needed [1].
- The ease of adoption of our flexible DSML by service designers, e.g. by applying cognitive dimensions [10].
- The integration of collaborative capabilities into our DSML. We plan to define a Design Rationale meta-model after one of the main rationale argumentative representation schemas. This MM will be integrated, together with the MM for annotations, in the abstract syntax MM of the DSML for service designers. Starting from this MM, we will be able to systematically generate the language environment. We may consider reusing or implementing a rationale inference engine.

- The interoperability between two DSMLs. Either a tool or a method for managing ontologies together with model transformations will be selected.

Our proposal could be investigated in other domains as well. For example, interoperability and flexibility are also major concerns in modern Information Systems (IS). Evolution of an IS should be a continuous process, and thus tools need to be iteratively adapted as well. Enterprise architects are pressured hardly to react to changing requirements, but fully integrated tools are not still mature. Several views are also considered in IS, e.g. ArchiMate(R) [14] offers a language standard, supported by different tool vendors, for describing respectively business processes, applications, and technological infrastructure.

## 6. ACKNOWLEDGMENTS

The authors wish to thank Patrick Alff, from BT-North America, and Martin Woods, from BT, for their early guidelines.

## 7. REFERENCES

[1] E. Bertin and N. Crespi. Service business processes for the next generation of services: a required step to achieve service convergence. *Annals of Telecommunications*, 64(3-4):187–196, 2009.

[2] J. E. Burge and D. C. Brown. Software engineering using rationale. *J. Syst. Softw.*, 81(3):395–413, 2008.

[3] V. Chiprianov and Y. Kermarrec. Model-based DSL Frameworks: A Simple Graphical Telecommunications Specific Modeling Language. In *IDM2009 French Colloquium on Model Driven Engineering*, pages 179–186, 2009.

[4] V. Chiprianov, Y. Kermarrec, and P. Alff. A Model-Driven Approach for Telecommunications Network Services Definition. In *Proceedings of the 15th Open European Summer School and IFIP TC6. 6 Workshop on The Internet of the Future*, pages 199–207. Springer, 2009.

[5] K. 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.

[6] A. V. Deursen, P. Klint, and J. Visser. Domain-specific languages: an annotated bibliography. *SIGPLAN Not.*, 35(6):26–36, 2000.

[7] A. Dutoit, R. McCall, I. Mistrík, and B. Paech. Rationale management in software engineering: Concepts and techniques. *Rationale Management in Software Engineering*, pages 1–48, 2006.

[8] P. Farail, P. Gaufillet, A. Canals, C. Le Camus, D. Sciamma, P. Michel, X. Cregut, and M. Pantel. The TOPCASED project: a Toolkit in Open source for Critical Aeronautic Systems Design. In *European Congress on Embedded Real Time Software (ERTS)*, number 781, pages 54–59, 2006. Tool available at http://www.topcased.org/, accessed 24th February 2010.

[9] C. Features. openarchitectureware 4.2. Technical report, Eclipse, 2007. Tool available at http://www.openarchitectureware.org/, accessed 24th February 2010.

[10] T. Green and M. Petre. Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. *Journal of Visual Languages and Computing*, 7(2):131–174, 1996.

[11] J. Grundy, J. Hosking, J. Huh, and K. N.-L. Li. Marama: an eclipse meta-toolset for generating multi-view environments. In *ICSE '08: Proceedings of the 30th international conference on Software engineering*, pages 819–822, New York, NY, USA, 2008. ACM.

[12] F. Jouault and I. Kurtev. Transforming models with ATL. *Satellite Events at the MoDELS 2005 Conference, W5 - MTiP*, LNCS 3844:128–138, 2006. Tool available at http://www.eclipse.org/m2m/atl/, accessed 24th February 2010.

[13] A. Ko, R. DeLine, and G. Venolia. Information needs in collocated software development teams. In *Proceedings of the 29th international conference on Software Engineering*, pages 344–353. IEEE Computer Society, 2007.

[14] M. Lankhorst and H. van Drunen. Enterprise Architecture Development and Modelling–Combining TOGAF and ArchiMate. *Via Nova Architectura*, 21, 2007. Available at http://tinyurl.com/yaxfh73, accessed 24th February 2010.

[15] C. Neumann, R. Metoyer, and M. Burnett. End-user strategy programming. *Journal of Visual Languages and Computing*, 20(1):16–29, 2009.

[16] B. Nuseibeh, J. Kramer, and A. Finkelstein. A framework for expressing the relationships between multiple views in requirements specification. *IEEE Transactions on Software Engineering*, 20:760–773, 1994.

[17] F. Penamora and S. Vadhavkar. Design rationale and design patterns in reusable software design. *Artificial Intelligence*, pages 251–268, 1996.

[18] A. Prout, J. Atlee, N. Day, and P. Shaker. Semantically Configurable Code Generation. In *Proceedings of the 11th international conference on Model Driven Engineering Languages and Systems*, pages 705–720. Springer, 2008.

[19] C. Scott, P. Wolfe, and M. Erwin. *Virtual private networks*. O'Reilly Media, Inc., 1999.

[20] M.-A. Storey, J. Ryall, J. Singer, D. Myers, L.-T. Cheng, and M. Muller. How software developers use tagging to support reminding and refinding. *IEEE Transactions on Software Engineering*, 35:470–483, 2009.

[21] 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.

[22] P. Tarr, H. Ossher, W. Harrison, and S. M. Sutton, Jr. N degrees of separation: multi-dimensional separation of concerns. In *ICSE '99: Proceedings of the 21st international conference on Software engineering*, pages 107–119, New York, NY, USA, 1999. ACM.

[23] T. Wolf. *Rationale-based unified software engineering model*. PhD thesis, TU Munchen, Germany, 2007.