Experiences of Developing a Network Modeling Tool Using the Eclipse Environment

Andy Evans[1], Miguel A. Fernández[2], and Parastoo Mohagheghi[3]

[1] Xactium Ltd. UK, Sheffield - UK
andy.evans@xactium.com
[2] Telefónica Research & Development – Valladolid, Spain
mafg@tid.es
[3] SINTEF, P.O.Box 124- Blindern, N-0314 Oslo, Norway
parastoo.mohagheghi@sintef.no

Abstract. Domain-specific modeling solutions have been promoted for some time in order to improve the productivity of software developers by providing them with modeling environments that are easier to learn, integrate best solutions and provide the possibility to automate software development by generating code from models. This paper presents experiences of developing a network modeling tool in Telefónica using Eclipse GMF. A metamodel based on Common Information Model was used in this development. While we experienced benefits in terms of better usability by domain experts, we also faced challenges such as the high level of expertise required to develop a good enough language and tool, the shortcomings of the tools in providing support for modeling at different abstraction levels, and the difficulties in updating the modeling tool with changes in the metamodel. These challenges must be overcome before the tool can be a part of our development environment.

Keywords: domain-specific modeling, language design, metamodeling, Eclipse.

1 Introduction

Throughout the history of software, developers have always sought to increase their productivity by improving abstraction. Domain-Specific Modeling (DSM) raises the abstraction level by offering the possibility to specify solutions directly using problem domain concepts [8]. Other artifacts are then generated from these high-level specifications. A modeling environment that fits to the concepts of the domain and the problem in hand is expected to be easier to be used by domain experts. A domain here is an area of interest; either a horizontal functional domain (such as user interface or persistency) or a vertical business domain such as telecommunications or retail. In this paper, network modeling in telecommunication is the domain of interest.

DSML tools are visual modeling tools based on some Domain-Specific Modeling Language (DMSL). DSMLs are promoted by some as the next big thing after General-Purpose Languages (GPLs) and there are reports of successful application in industry such as in Motorola [2] and examples presented in [8]. However, reports of experience are still few and far between, as a review of literature on industry experience with

Model-Driven Engineering (MDE) has confirmed [1]. There may be several reasons for that; firstly the development of DSM solutions does not have a long history; and secondly DSM solutions are valuable assets that give companies competitive advantage so experience reports may be kept private and not published.

The European research project MODELPLEX (MODELing solution for comPLEX software systems)[1] aims at evolving MDE tools and technologies to be applicable for developing complex software systems and evaluating them in the context of four, very different, industrial scenarios. As an industry case provider in MODELPLEX, Telefónica has participated in specifying requirements regarding tools and technologies, customizing the solutions, evaluating them and providing feedback to both tool providers and industry interested in applying these solutions. One of the areas of research has been the development of a DSML for network modeling based on the CIM (Common Information Model) metamodel defined by the DMTF [5].

Taking into account the high cost of developing a DSML, a company needs to make a serious evaluation of the Return On Investment (ROI), balancing the expected benefits and productivity improvement against the cost of development and future maintenance of the tools before moving to a new development environment. The purpose of this paper is to report on our experiences of developing a DSML based around the aforementioned Telefónica case using the widely used Eclipse GMF tool development framework. We also identify useful criteria for evaluating the resulting DSML tool which might be part of a ROI analysis when a solution is developed.

The remainder of the paper is organized as follows. Requirements of the DSML and the criteria for evaluating it are presented in Section 2. Section 3 presents the steps of the development while Sections 4 and 5 present our experience with the environment used for developing the DSM tool and the DSM tool itself. Section 6 presents the challenges for developing a DSML based on our experience (Xactium has long experience with language-driven development[2]). Finally Section 7 presents a conclusion and discussion of future work.

2 Requirements of DSML and Criteria for Evaluation

2.1 Requirements

As mentioned in the introduction, the purpose of this report is to discuss the development of a specific DSML for network service modeling. The key driver for this DSML is the wide recognition that it is becoming increasingly difficult to manage the complexity and size of modern telecom networks [4]. To address these challenges, it was proposed that a DSML be developed to enable the modeling of complex networks delivering services to private subscribers via a range of different devices. This approach would provide Telefónica with a modeling language to capture the key features of these networks and services at a level of abstraction that enables the management of complex networks more efficient and more manageable. Some specific areas of requirement were identified as follows.

[1] http://www.modelplex-ist.org/

[2] The idea of language-driven development is providing developers with an integrated collection of semantically rich languages that specifically target their development needs.

By Telefónica's requirement, the Network DSML had to include, but was not necessarily limited to, the following concepts; a) network topology (e.g., sub-network addressing); b) device properties (e.g., interfaces, firmware version, etc.); and c) types of network traffic and service features (e.g., protocols, port ranges, QoS, etc.) In addition, a key requirement of the Network DSML tool was to allow modeling at different levels of abstraction, at least the following: a) device, showing internal device details, e.g. network interfaces; b) network topology, showing how devices connect to each other; and c) service, showing higher-level interactions and roles of whole sub-networks in the deployment of a service.

From these models, a wide range of artifacts could be generated, the first ones on the list being device configuration specifications to be fed, in the appropriate format, to Telefónica's OSS subsystems in charge of device configuration and monitoring.

Rather than develop these concepts from scratch, it was proposed by Telefónica that the Common Information Model (CIM) [5] would provide a useful starting point. This model provides many concepts useful to the modeling of networks and devices. A significant part of this model was identified as being relevant to the requirements of the project, and with some additional changes, this became the core model around which the DSML was based. CIM was also relevant as it is the underlying model in many COTS products dealing with management and instrumentation of network equipment, some of which are part of current Telefónica's OSS.

Finally, there were a number of generic features of the DSML tool, which were required in order to meet the needs of users of the tool. These included: a) a visual, user-friendly interface; b) scalability – enabling thousands of model elements to be managed; c) interoperability with other tools and standards; d) flexibility – enabling the rapid adaptation of the tool to support new abstractions (preferably done by the engineers themselves); and e) support for model validation and checking.

2.2 A Framework for Evaluating the DSML

Industrial participants in the MODELPLEX project have defined a set of research questions for evaluating the solutions. There, Telefónica has stated that, "We expect to develop a DSML that helps us create these models in a way that proves convenient for business experts with no technical background in modeling". The questions are:

> *"Can a telecommunication business expert model services by means of a DSML? How valid are the models in terms of completeness and usefulness for the generation of other artifacts?"*

As part of the MODELPLEX project we have also performed a state of the art analysis regarding evaluation of languages (in this case a DSML) and identified the stakeholders and their points of interest, as depicted in Fig. 1:

- *Language Engineers (LE)* are those developing the language. They evaluate a language based on whether the language features are easy to implement or whether it is easy to develop compilers or generators.
- *Language Users (LU)* are personnel using the DSML for modeling, which are interested in ease of use, increased productivity, etc.

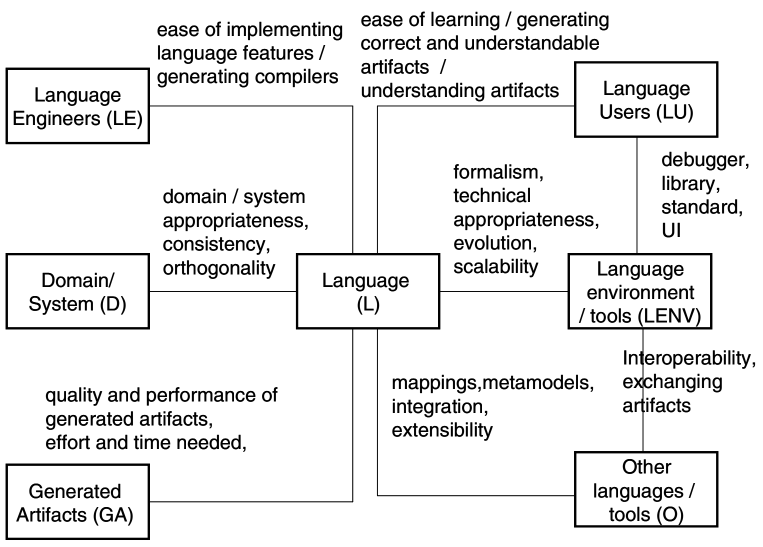
Language Engineers (LE)
ease of implementing language features / generating compilers
ease of learning / generating correct and understandable artifacts / understanding artifacts
Language Users (LU)
debugger, library, standard, UI
formalism, technical appropriateness, evolution, scalability
domain / system appropriateness, consistency, orthogonality
Domain/ System (D)
Language (L)
Language environment / tools (LENV)
Interoperability, exchanging artifacts
mappings,metamodels, integration, extensibility
quality and performance of generated artifacts, effort and time needed,
Generated Artifacts (GA)
Other languages / tools (O)

Fig. 1. Evaluating a language from multiple views

- *Language Environment (LENV)* is the tool (including editors and transformations) which is developed around the DSML. There are requirements such as whether the language is formal, evolvable or scalable. Besides, including debuggers and libraries, being standards-based and compatible with other tools, and having a pleasant User Interface (UI) all increase the value of the LENV.
- *Domain/System (D)* is the domain of interest. A language should be appropriate for the domain, the concepts should be consistent, etc.
- *Other languages / tools (O)* cover requirements for interoperability, mappings between languages, building future extensions, etc.
- *Generated artifacts (GA)* may have requirements regarding quality, performance, and effort or time needed for generation.

While the framework shown in Fig. 1 is developed with languages in mind, in MODELPLEX we also take advantage of an extended version of the Technology Acceptance Model (TAM) for evaluating tools and technologies. The original TAM, by Davies, is widely referenced [9] and used in information science research. It explains users' intention to use a new system through two beliefs, *perceived usefulness* and *perceived ease of use*. There are several extensions to TAM and we use the model described in [11] for evaluation of the DSML (and the base DSL) as depicted in Fig. 2 where we have also inserted requirements identified in the previous section and the stakeholders as defined above. We define these factors as:

- *Perceived Usefulness (PU)* is the degree to which a person believes that using a particular method or tool will enhance their job performance.
- *Perceived Ease of use (PE)* refers to the degree to which a person believes that using a particular method or tool would be free of effort.

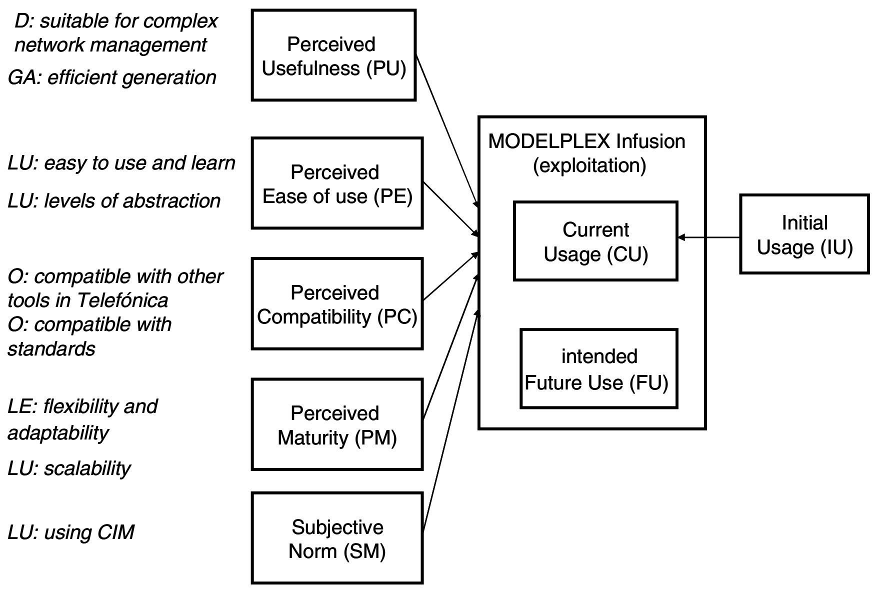
D: suitable for complex network management
GA: efficient generation
LU: easy to use and learn
LU: levels of abstraction
O: compatible with other tools in Telefónica
O: compatible with standards
LE: flexibility and adaptability
LU: scalability
LU: using CIM
Perceived Usefulness (PU)
Perceived Ease of use (PE)
Perceived Compatibility (PC)
Perceived Maturity (PM)
Subjective Norm (SM)
MODELPLEX Infusion (exploitation)
Current Usage (CU)
intended Future Use (FU)
Initial Usage (IU)

Fig. 2. The model used for evaluating DSM

- *Perceived Compatibility (PC)* is the degree to which an innovation is perceived as being consistent with existing practices, standards and tools, and the past experience of potential adopters.
- *Perceived tool Maturity (PM)* is the degree to which tools are perceived as mature and suitable for the tasks in hand.
- *Subjective Norm (SM)* is the degree to which software developers think that others who are important to them think that they should use that particular method or tool.

For evaluating each factor, a set of questions is defined from the stakeholders' viewpoint and their interest in the developed language in the context of the company. These questions are listed in Section 5 together with the answers. Originally, the evaluation was intended as a questionnaire. However, we performed a qualitative analysis by three engineers form the MODELPLEX research team at Telefónica instead, since the DSML was not used by a significant number of developers then.

3 Design and Implementation of the DSM

For this project we used the Eclipse Graphical Modeling Framework (GMF) [6] plug-in to develop the DSML. Our reasons for choosing GMF were its relative popularity and maturity and the fact that it is open source and based on Eclipse (one of the key platforms mandated by MODELPLEX by virtue of its interoperability and openness). GMF provides the ability to develop a working tool for the graphical representation of data, based entirely on an EMF (Eclipse Modeling Framework) [7] model and complying to all the relationships and constraints specified in that model.

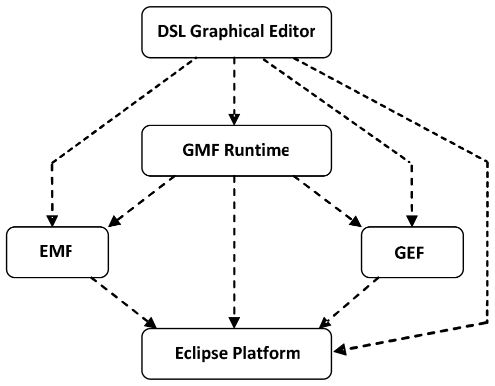

Fig. 3. Relations between the developed DSML editor and the Eclipse components

The process of creating a graphical model editor in GMF requires the use of other Eclipse components such as the Eclipse Graphical Editing Framework (GEF). So when trying to understand the relationship between GMF and EMF it is also important to take into account their relationship to the Eclipse Platform, on which they are built. Fig. 3 is a representation of that relationship. As we can see, a GMF-based DSML graphical editor depends on the GMF runtime component but also makes direct use of EMF, GEF and the Eclipse platform.

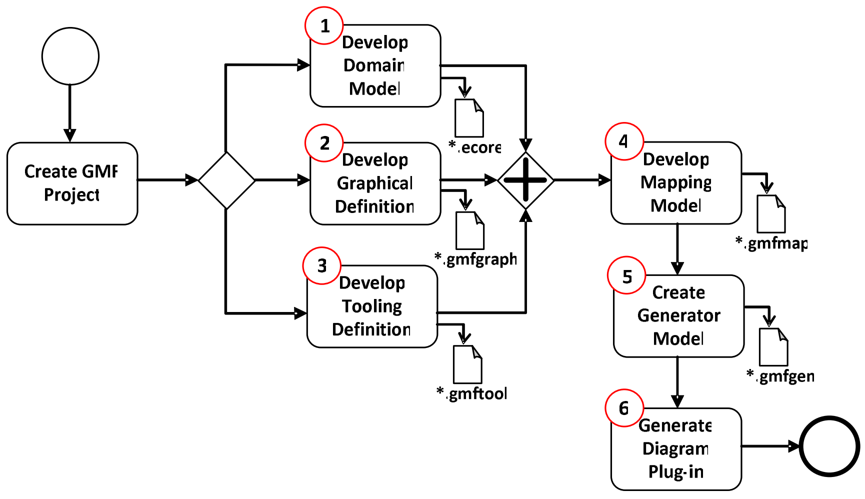

Fig. 4. Steps in creating the DSML editor

The first step in the development of this tool was the creation of an EMF model (or metamodel). From this model specification, a set of Java classes are produced which can later be used as the foundation for our tool. The CIM metamodel was initially transformed into Ecore by Xactium and then modified by Telefónica to the needs of their specific domain. A significant challenge of this stage of development is the size of the CIM metamodel which contains over 1500 concepts, hence a reduced subset was used in the first implementations, consisting of more than 200 concepts.

The next step was to take our (CIM) metamodel and begin the development of the Eclipse graphical modeling editor. The basic components of the GMF model we developed are depicted in Fig. 4 and described below:

1. The domain model defines the non-graphical information managed by the editor (this can be generated directly from our EMF model).
2. The graphical definition model contains information related to the graphical elements that will appear in a GEF-based runtime, but has no direct connection to the domain models for which they provide representation and editing.
3. An optional tooling definition model is used to design the palette and other periphery (menus, toolbars, etc).
4. The diagram mapping model defines mappings/relationships between domain model elements and graphical elements.
5. Once the appropriate mappings are defined, GMF can produce a generator model from which the code could be generated.

This process was repeated over a number of iterations, to produce a fully working tool. An example of the tool in action is shown in Fig. 5.

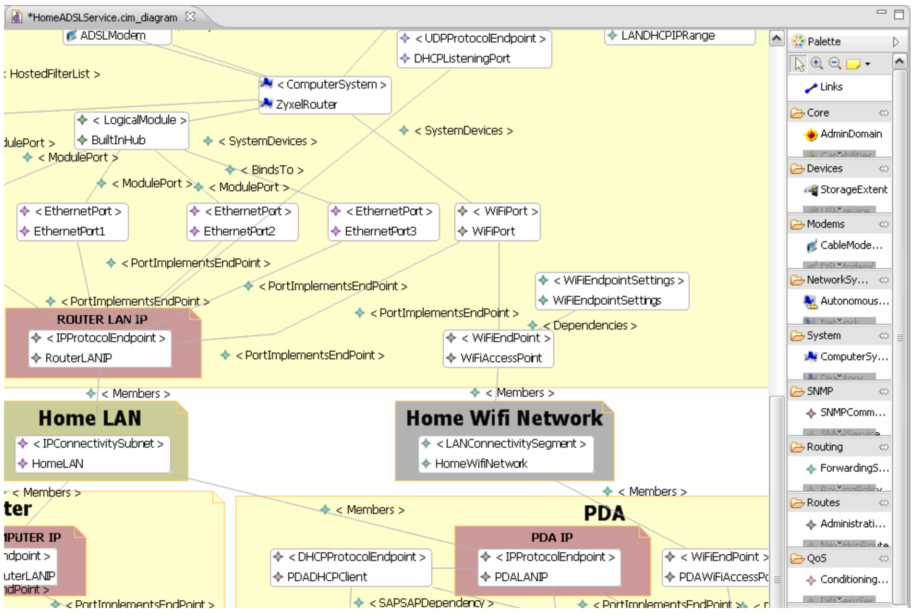

Fig. 5. A fragment of a model in the network DSML tool

4 Experiences with GMF/EMF

One of the most challenging aspects of this DSML was the large number of modeling abstractions and relationships in the CIM model. As a result, it was decided to develop the tool as a graph editor, with each node in the graph representing a class instance and each edge in the graph representing a relationship between classes. As an example, a device would be represented as a node, labeled "Device", and its relationships (both direct and inherited from parent classes) as edges.

When developing the mapping model it was apparent that there was an issue in managing containment relationships. In GMF, containment relationships by default map to containment structures in the diagram model. However, given the large number of containment relationships, it was not practical to make special cases for each diagram mapping as many of these relationships would need to be represented in different ways, e.g. as a sub-node or sub-diagram. Moreover, for many concepts, it did not make sense to treat their diagram representation as a container in any case.

To address this issue we adopted a GMF development technique called phantom (or shadow) nodes. These are simply nodes without their containment feature set. The use of this technique was necessary but we knew this would cause problems at a later stage since, when using it, the top-level nodes should still have a containment relationship to the canvas and this was not acceptable in our model. This issue was solved by making changes to the generated code, which involved editing the create function for each of the nodes on the diagram to give them a containment relationship to the canvas when they are drawn on the diagram. Due to the use of shadow nodes, all nodes could be given a containment relationship of a different type by the user.

Another challenge was that of making the tool as usable as possible, which involved changing the tooling definition. Again, the large number of abstractions was an issue which had to be addressed as simply as possible. To do this, we grouped classes into groups and then also grouped diagram components to a tool based on their types; this reduced the number of palette elements and increased usability significantly. We made further changes to the tooling palette as well by changing the *Icons* in the *.edit* file by grouping tools with icons.

When using the editor we discovered that the automatically generated popup menus and connection handles were more of a hindrance than help, due to the complexity of the model. The popup menu was overly large due to the number of creation tools for each component in the diagram, also connection handles produced an incorrect output due to the scope of the model that the tool is built on. To solve this issue we removed the popup menus and connection handles by adding some code to the *diagramEditPart* of each diagram element (including the nodes and the canvas).

One of the key requirements of the DSML tool was that it provided sufficient flexibility to enable rapid changes to the metamodel, thus enabling the tool to adapt to changing modeling requirements. Unfortunately, this was not supported well by GMF. Even small changes to the model require repeating the code generation steps and there was always significant risk that errors would creep into the generated code. Furthermore, any changes required someone with strong technical expertise in GMF.

Another important aspect missing from GMF is the provision of a facility to encapsulate levels of abstraction through the use of components or product line concepts. For example, in the case of a 'router' concept, it could be thought of as being composed of a collection of more primitive elements. Ideally, the tool needs to provide an easy to use mechanism for creating abstractions as patterns of more fundamental elements. Again, the facility should not be reliant on the re-generation of the tool, but should provide the ability to create new abstractions dynamically.

5 User Experience with the DSML Tool

Here we present the opinion of the users of the tool to a set of questions related to the TAM evaluation criteria presented in Section 2.2.

Table 1. Results of the DSML tool evaluation

Perceived Usefulness	
1. Is the CIM metamodel suitable for modeling network management in Telefónica?	Yes, they are suitable for this purpose but need constant revision and extension to keep up with the evolution of the domain and the standard of reference (CIM).
2. Do the DSML and the artifact generation capabilities affect quality, performance and productivity of the work?	Yes, the DSML has the potential to improve productivity and quality but additional work and training, as well as other tools like model transformation languages, etc., are needed to achieve those objectives.
Perceived Ease of use	
1. Is the DSML tool easy to use? Is the UI acceptable?	Not enough, largely due to the sheer size of the metamodel which resulted in having to add a large number of connection and node tools.
2. How can the abstraction layers improve models and their understanding?	Abstraction layers are necessary in cases such as this and can improve greatly the understanding and usefulness of the models. The problem is that abstraction layers are not supported in GMF and, even with the addition of a model composition framework, the level of integration achieved was not sufficient.
Perceived Compatibility	
1. Is the DSML compatible with the standards?	Yes, using CIM provides such compatibility but brings problems due to its size.
2. Is the DSML compatible with other tools?	Many tools used in the network management domain are based on CIM, but as the DSML transforms the CIM metamodel into EMF, this leads to compatibility issues with CIM-based off-the-shelf products that need to be resolved.
Perceived Maturity	
1. Is the solution scalable?	GMF does not scale well because of some shortcomings in the implementation that have been already discussed.
2. Is the solution flexible?	The same applies to flexibility. A more dynamic, meta-model-driven tool generation approach is needed.
Subjective Norm	
1. How would others judge our use of the DSML? Do we think that it improves our reputation and image as innovative?	Yes, the image and reputation of innovation can be greatly improved by the use of tools and approaches such as the one presented herein.

6 Challenges for DSML Technologies

We recognized two main challenges (or shall we say obstacles) during developing the DSML solutions: a) developing a DSML in an environment such as Eclipse requires high language expertise and tool expertise, which make developing DSMLs out of reach of domain experts with some IT expertise; and b) the resulting DSML is not changeable or flexible enough. We describe these in more detail below with outline of solutions.

a) More user-centric development environment

Understand that DSMLs are a business solution, not a technical one. While the community of users of Eclipse and GMF is growing, and there are many examples of DSML tools that have been developed using the technology, there is too much emphasis on GMF as a technical solution rather than a business solution. This is widely reflected by the large number of academic conferences on the subject of DSMLs, the relative lack of involvement of business users in the development and use of DSMLs, and the fact that the development of DSL technologies is largely being driven by programming experts and IT groups. Until this is addressed, DSML tools, certainly in the case of those built using Eclipse, will not achieve critical mass for business users, and will largely remain the domain of academic interest and IT research departments. Such lack of critical mass poses a significant issue to large companies like Telefónica, who have to take into account the costs of supporting and maintaining non-mainstream technologies in the long term.

Enable DSMLs to be developed by end users. A problem encountered with GMF was the significant technical expertise required to develop DSML tools, even simple ones. This is a significant challenge for users who are not technically minded as, in practice, they will have the best understanding of their domain. It seems particularly strange that although a key objective of DSMLs is to provide a more targeted and flexible domain solution, the ability to create DSMLs is only accessible to experienced programmers.

Address abstraction zoom-in and zoom-out, to help simplifying the models and allow reuse. During this project we have identified a specific example of flexibility which is an important requirement for modeling (certainly in the telecom domain). Because many telecom systems are built up of components, which are themselves composed of more granular components, there is a requirement to be able to create pre-defined combinations of components which can be combined together in new ways. While it is possible to create component models in UML, there is an advantage of being able to generically combine different DSML concepts into reusable components. Whilst this capability is not available 'out of the box' with GMF and other DSML tools, we are examining how the Eclipse Reuseware [10] initiative might address this.

Support multi-user, multi-tenancy DSML tools. Once a DSML has been developed and is in use, a significant challenge is to scale its use to multiple users. While models created in Eclipse can be exported to others users, there is no simple mechanism for ensuring changes to models by one user are kept in sync with changes made by other users. Data can soon become out of step, and the effort to resolve changes becomes prohibitive for successful commercial use. In other areas of business software, for example, database applications, such issues have been recognized and addressed through multi-user support, while multi-tenancy solutions address critical issues of managing and upgrading data when underlying changes to the database are made. This challenge is not specific to DSMLs but to modeling tools in general.

b) Need for flexible solutions:

Enable non-programming customization and adaptability. As identified above, a problem encountered in the development and use of the CIM DSML tool was the problem of adapting it to new requirements. These adaptations were primarily around the changes to the underlying metamodel, including changes to properties, relationships and the addition of new domain concepts. However, they could also include changes to diagrammatical representations, and also hiding and showing of user relevant information, for example fields. GMF completely failed in this regard due to the fact that any changes (whether simple or complex) required re-generation of the code.

Support dynamic management of data and metadata. Another key requirement for adaptability is the ability to accommodate changes to the metamodel without making existing model data redundant. While this was not tested fully, in a number of cases, models became corrupted due to changes in the DSML tool and could not be reused without significant modification of the underlying XML file. The alternative, of creating a model-to-model mapping to transform the data would again necessitate significant programming expertise. We will investigate other solutions to this problem in the MODELPLEX project but we fear it is a complex issue to solve.

7 Conclusions and Future Work

The purpose of this paper is to report on our experiences with a widely used, open source framework for building DSML tools- the Eclipse GMF framework. While we do not wish to claim that the challenges identified in its use are applicable to all DSML technologies, we do believe the challenges we have identified are an important consideration when evaluating and developing DSML technologies (particularly when assessing ROI). One particular issue with regard to GMF, was the need for a more user-centric tool development process that would enable end users and domain experts to participate more fully with the tool design process. A second issue is the need to encapsulate levels of abstraction, again provided in a user friendly way. We view both of these challenges as an essential requirement for the wider commercial uptake of DSML technologies. Whilst there may be existing technologies available which overcome these challenges, we believe our focus on GMF is important as it is one of the leading technologies in the marketplace.

We also believe that a more flexible approach to DSML development is required, which would support the dynamic creation of DSMLs as opposed to the generative approach taken by GMF.

Solving the above issues would enable DSML tools to be created by domain experts rather than software developers, thus providing a more interactive and user-centric approach to DSML development. Some key features of this approach must be; a) the use of metadata to configure and customize the resulting tools on the fly; b) a user friendly interface for customizing the editors – probably via a DSML tool; c) the ability to easily upgrade model data without the need for complex transformations; d) the ability to easily customize the resulting tool, for example, in terms of look and feel, icons, etc; and finally e) the ability to represent patterns of concepts of higher level abstractions, which can themselves be reused in the tool.

We are at the moment exploring ways of adapting the existing GMF technologies to provide a DSML tool generation engine. This would provide a way of dynamically creating tools by loading the tool model into the engine (rather than generating the code). Another contribution of the work has been developing a framework for evaluating DSML solutions which will be reused in future work. We hope to report on all this as part of the MODELPLEX project.

Acknowledgments. This work has been done in the MODELPLEX project (IST-FP6-2006 Contract No. 34081), co-funded by the European Commission as part of the 6th Framework Program.

References

1. Mohagheghi, P., Dehlen, V.: Where is the Proof? A Review of Experiences from Applying MDE in Industry. In: Schieferdecker, I., Hartman, A. (eds.) ECMDA-FA 2008. LNCS, vol. 5095, pp. 432–443. Springer, Heidelberg (2008)
2. Baker, P., Loh, P.S., Weil, F.: Model-Driven Engineering in a Large Industrial Context - Motorola Case Study. In: Briand, L.C., Williams, C. (eds.) MoDELS 2005. LNCS, vol. 3713, pp. 476–491. Springer, Heidelberg (2005)
3. Mohagheghi, P., Fernandez, M., Martell, J.A., Fritzsche, M., Gilani, W.: MDE Adoption in Industry: Challenges and Success Criteria. In: ChaMDE Workshop at MoDELS 2008, To be publised in the Proc. of Workshops at MoDELS 2008 (2008), `ftp://ftp.umh.ac.be/pub/ftp_infofs/2008/ChaMDE-report`.pdf
4. Wong, D., Ting, C., Yeh, C.: From Network Management to Service Management – A Challenge to Telecom Service Providers. In: Proc. 2nd International Conference on Innovative Computing, Information and Control (ICICIC 2007) (2007)
5. DMTF's Common Information Model Website, `http://www.dmtf.org/standards/cim/`
6. Eclipse Graphical Modeling Framework (GMF), `http://www.eclipse.org/gmf/`
7. Eclipse Modeling Framework (EMF), `http://www.eclipse.org/emf/`
8. Kelly, S., Tolvanen, J.-P.: Domain-Specific Modeling- Enabling Full Code Generation. IEEE Computer Society Publications, Los Alamitos (2008)
9. Davis, F.: Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly 13(3), 318–339
10. Reuseware Composition Framework, `http://www.reuseware.org/`
11. Dybå, T., Moe, N.B., Mikkelsen, E.M.: An Empirical Investigation on Factors Affecting Software Development Acceptance and Utilization of Electronic Process Guides. In: Proc. 10th International Symposium on Software Metrics (Metrics 2004), pp. 220–231 (2004)