A DSL for Configuration Management of Integrated Network Management System

Rosangela Pieroni[1,2] and Rosângela Aparecida Dellosso Penteado[1]
[1]*Department of Computing, Federal University of São Carlos, 13565-905, São Carlos, SP, Brazil*
[2]*CPqD (Research And Development Center In Telecommunications), 13086-902, Campinas, SP, Brazil*

Keywords: Model-Driven Development (MDD), Domain Specific Languages (DSL), Integrated Network Management.

Abstract: A management system of networks that takes all elements into consideration, regardless of the network technology, is one of the most emphasized requests by telecommunication companies. However, developing this system is not a trivial task. Furthermore, the software development process based on source code makes the task even more complex and requires a great effort of the developers to perform code update and maximize the reuse of software artifacts to insert a new network technology. In this paper, we propose a DSL approach to specify new network technologies into integrated network management system developed by a real company. An experiment was conducted according to all steps proposed by Wohlin (Wohlin et al., 2000) to validate our DSL approach versus specialization of classes with the purpose of increasing advantages with respect to time and number of errors inserted in the source code. Although the time spent to develop the application using the two approaches was not statistically different, all other results obtained such as code generated automatically without present errors and all comments reported by the participants regarding the ease of use of DSL, it encourages the development of new DSLs to others functions of the integrated network management system.

1 INTRODUCTION

The developing a management system of networks that takes all elements into consideration, regardless of the network technology, is one of the most emphasized requests by telecommunication companies. The advantage of an integrated management is to obtain the information of the network elements from a single software system. Once this is obtained, another advantage is to correlate the alarms of network elements and find the cause of the problem more assertively and, finally, fix it. This results in the provision of better quality services, which will be available in the telecommunication network to its users. The main customers of the integrated network management are those who already have a telecommunication network comprising elements of various technologies. Customers are interested in increasing assertiveness of the network problem search to make the necessary corrections without diminishing the quality of services provided. In this context, it may be necessary to add new network technologies into the system. However, developing an integrated network management system is not a trivial task and it is necessary to know each network technology. Furthermore, the software development process based on source code makes the task even more complex and requires a great effort of the developers to perform code update and maximize the reuse of software artifacts to insert a new network technology.

Model Driven Development (MDD) is an alternative to decreasing or minimizing these problems, considering the importance of models in software development process (Mellor; Clark; Futagami, 2003). In this paper, we propose a DSL approach to specify new network tecnologies. Thus, the system will be able to discover the new network elements belonging to a new network technology, and, from this discovery, these new elements will be managed by the system. To evaluate this approach, we will consider an integrated network management system developed by a company, whose name is withheld for reasons of confidentiality. Although the network management consists of five management areas (configuration, fault, security, performance and accounting), the system analyzed consists of only two of them, configuration and fault. Configuration management performs the discovery of the network elements belonging to technologies known by the system. The fault management handles events received from the discovered network elements. When these events are classified as alarms, correlation rules are applied in order to find out the root cause of the problem in the telecommunication network more assertively. Thus, the trouble ticket process to solve the problem and make the network operational and available again becomes more effective.

In this section we have described the general purpose of this paper. In Section 2 we will show the background study that resulted in this paper. In Section 3 we will present the DSL approach. In Section 4 we will describe the experiment done to validate the DSL. In Section 5 we will present the analysis of data gathered during the experiment. In Section 6 we will present the threats to the experiment's validity. In Section 7 we will present works related to the theme. In Section 8 we will present the limitations of the proposed solution. In Section 9 we will present the conclusion and intended future work. Finally, in Section 10 we will present the references.

2 BACKGROUND

Model Driven Development (MDD) is a technique for developing software systems in which models of high-level abstraction describe the system functions (Gronback, 2009; Lucrédio et al, 2008; Völter, 2008). These models are refined in other models until they are transformed into a source code. Thus, the higher level of abstraction model becomes part of the software.

In MDD, the functions are represented by models and correspond to the business domain rules of a system. All models are specified according to the language of its metamodel (meaning, the representation obtained by high level models of abstraction regarding the business domain rules). The models are incorporated as a part of the final software product by modeling techniques and automatic code generation (Durelli, 2011).

With the use of high level models of abstraction and automatic code generation, software developers are shielded from the complexities of the implementation platform (France; Rumpe, 2007). They may dedicate themselves more to the business domain (Hutchinson et al., 2011), because the complexity of the software is hidden by the artifact automatic generators. These artifacts reflect the solution expressed in models of high abstraction level (Schmidt, 2006). MDD may, therefore: provide software reuse more procedurally enables faster development, lower costs, make software production more flexible, and allow changes to be made more quickly (Antkiewicz; Czarnecki, 2006).

Domain Specific Language (DSL) is defined as a small and declarative computer language which is used in a particular domain in order to perform specific tasks (Fowler, 2005; Deursen et al., 2000). DSLs are used in MDD to facilitate modeling and transformations, since they are restricted to a domain language that allows the generation of the code proposed by MDD (Djukic et al., 2011).

The advantages of DSL are, as follows: it allows specific abstractions of a domain which are pre-defined and directly represent concepts of the application domain; it increases the level of abstraction; it generates concise codes; it prepares the codes to be reused; it generates just enough documentation for its use; it provides the stakeholders, as well as domain experts and developers with insights about the system. This way, developers can worry about other parts of the business system and validate and optimize only the domain level.

However, DSL requires modeling tools, processing and code generators that are more complex in comparison to those used in traditional development. Thus, when a DSL is developed, the team's experience in these tools should be evaluated and, where necessary, a learning curve shall be considered as an option. Moreover, when there is an expert in the domain problems in the development team it's possible to make the development more incisive, adding relevant knowledge of the domain. In classical development, there is not always the presence of an expert in the field of business on the development team to assist the work of the requirements analyst, who is responsible for the gathering of software requirements with the stakeholders.

3 DSL DEVELOPMENT

Evidence of the importance and necessity to apply software reuse techniques in a higher level of abstraction is derived from the characteristics of integrated network management system. We proposed applying MDD concepts to develop a DSL, which will facilitate the process of specifying a new network technology in the integrated network management system. To exemplify, once the new network technology is specified in the model and the automatic code generation has taken place, the latter will hide the complexity of development platforms from the developers, the software can be developed within the schedule targets and effort reduction and, consequently, it will be possible to meet the market's increasing demands, keeping up with quality standards of telecommunications services provision.

Eclipse Modelling Framework (EMF) (EMF, 2013) was the tool chosen to develop our purpose. EMF models are represented according to its metamodel, called Ecore. Acceleo (Acceleo, 2013) was the specific tool used to create the templates to transform Model to Code (M2C) in this approach.

To apply the concepts of MDD and develop DSL, a specific function was selected among the configuration management functions of the integrated network management system of a real company. This system has many functions (common and specific ones) and is continuously evolving due to the requests of other functions arising from various customers. Therefore, it was necessary to establish a baseline system and delimit the scope of the specification of a new network technology. Thus, the application of the concepts of MDD through the DSL development did not cover the entire integrated management system networks. The function selected was chosen because it was considered critical due to the variety of technologies that should be managed by the system and for being one of the features that most suffers change requirements.

The metamodel of the DSL for configuration management for the integrated network management system can be seen in Figure 1. The business rules to specify a new network technology are presented by this metamodel and basically define that the network technology could have zero or several configuration groups. These configuration groups can refer to a field type or a table type. The elements of these configuration groups are the attributes. These attributes are specific of network technology and must be presented in the MIB (Management Information Base) of the network element. The communication protocol used to perform discovery of the network elements is SNMP (Simple Network Management Protocol) therefore, the elements must have a MIB which is basically retrieved management information.

Part of the classes diagram of this system, which is responsible for specifying a new network technology, in classical development, can be seen in Figure 2. In this classes diagram, in order to perform the discovery of a new network element belonging

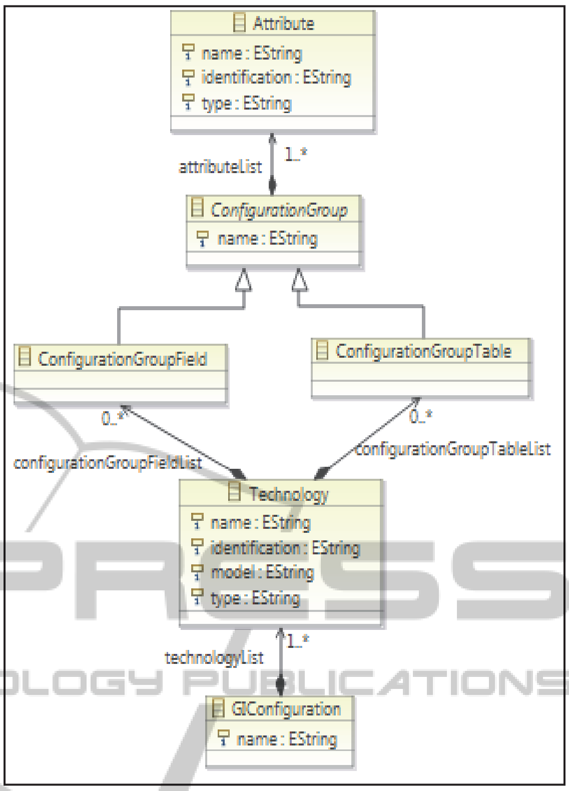


Figure 1: Metamodel for configuration management.

to a new technology it is necessary to implement two Java classes: Discovery<Technology>Session and Discovery<Technology>, where <Technology> is replaced with the name of the technology. The Discovery<Technology>Session class is responsible for registering the new technology in the list of technologies managed by the system. And the Discovery<Technology> class is responsible for defining the attributes of the network elements of the new technology to be managed by the system. In Discovery<Technology> class are encapsulated the particularities of the new technology to be managed by the system. Thus, when a new technology emerges and needs to be managed by the system, the developer must implement two new classes corresponding to the new technologies: Discovery<Technology>Session and Discovery<Technology>.

Acceleo tool (Acceleo, 2013) is used to transform the specified model into Java code using templates. The template developed corresponds to the two Java classes. Thus, the code must to be written by the developers to specify a new network technology in a classic development process is automatically generated when the DSL approach is used.

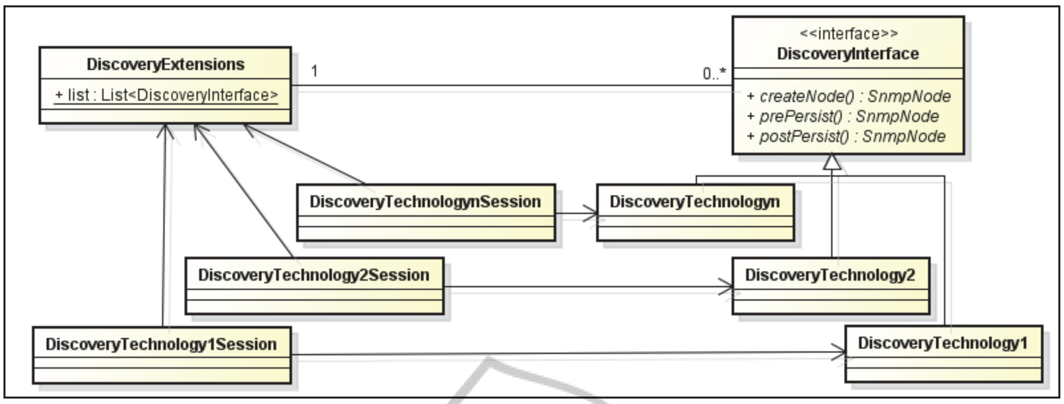
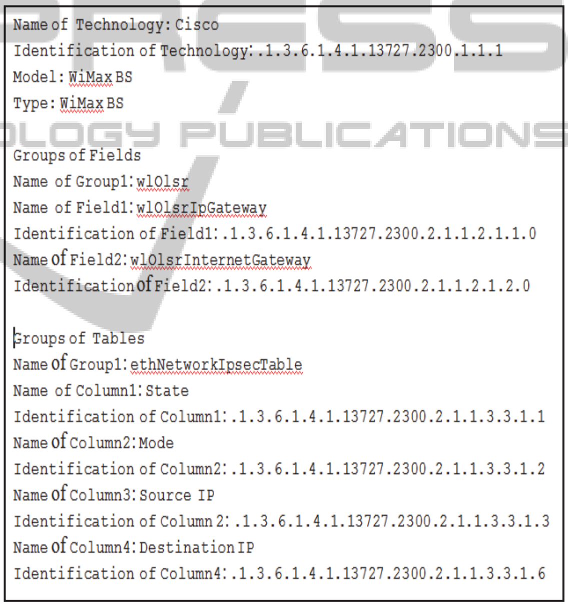

DiscoveryExtensions
+ list : List<DiscoveryInterface>
1
0..*
<<interface>>
DiscoveryInterface
+ createNode() : SnmpNode
+ prePersist() : SnmpNode
+ postPersist() : SnmpNode
DiscoveryTechnologynSession
DiscoveryTechnologyn
DiscoveryTechnology2Session
DiscoveryTechnology2
DiscoveryTechnology1Session
DiscoveryTechnology1

Figure 2: Network technology classes diagram.

The specification of the new network technology using the model and making the transformation model to code replaces the work of the developer of a real company that, currently, needs to conduct the specialization of classes manually for each new network technology that must be managed by integrated network management system.

The specification of a new network technology depends exclusively on the requirements that the network expert sets. This specification is a requirement document used by the developers to create the model. An example can be seen in Figure 3. In this specification, the technology is called Cisco, and its MIB identification (OID) is .1.3.6.1.4.1.13727.2300.1.1.1. This technology has a configuration group of field type called wlOlsr. This configuration group has two attributes: wlOlsrIpGateway and wlOlsrInternetGateway, which have the MIB identification .1.3.6.1.4.1.13727.2300.2.1.1.2.1.1.0 and .1.3.6.1.4.1.13727.2300.2.1.1.2.1.2.0, respectively. This technology has also a configuration group of table type called ethNetworkIpsecTable. This configuration group has four attributes: ethNetworkIpsecTable, mode, source IP and destination IP, which have the MIB identification .1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.1, .1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.2, .1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.3 and .1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.6, respectively.

The model for this new network technology specified using our proposed metamodel can be seen in Figure 4. The new network technology is represented by a tree. GIConfiguration is the root of tree, the technology Cisco has two configuration groups, and each group has its attributes. Each attribute has a property window where it is possible to inform the MIB identification and the name such as it is in the specification of a new network technology. For instance, the ethNetworkIpsecTable attribute can be seen in Figure 5.

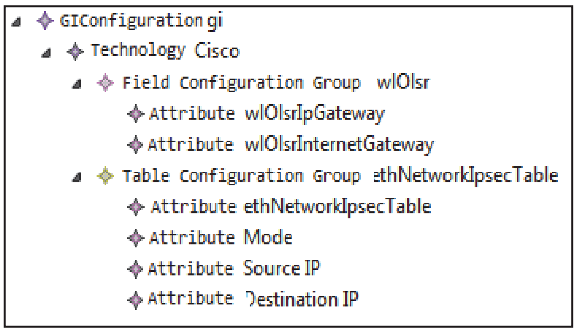

```
Name of Technology: Cisco
Identification of Technology: .1.3.6.1.4.1.13727.2300.1.1.1
Model: WiMax BS
Type: WiMax BS

Groups of Fields
Name of Group1: wlOlsr
Name of Field1: wlOlsrIpGateway
Identification of Field1: .1.3.6.1.4.1.13727.2300.2.1.1.2.1.1.0
Name of Field2: wlOlsrInternetGateway
Identification of Field2: .1.3.6.1.4.1.13727.2300.2.1.1.2.1.2.0

Groups of Tables
Name of Group1: ethNetworkIpsecTable
Name of Column1: State
Identification of Column1: .1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.1
Name of Column2: Mode
Identification of Column2: .1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.2
Name of Column3: Source IP
Identification of Column 2: .1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.3
Name of Column4: Destination IP
Identification of Column4: .1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.6
```

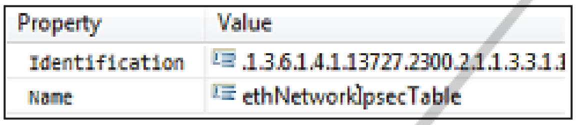
Figure 3: Specification of a network technology.

After the developer specifying the model for a new network technology, the code generation corresponding to the model can be requested. In turn, the code of two classes Discovery<Technology> and Discovery<Technology>Session is generated completely autonomously, i.e., the developers do not have to write any code. For instance, the code generated to the field configuration group is the code shown in Figure 6. The code to the table configuration group is the code shown in Figure 7.



```
◢ ✦ GIConfiguration gi
   ◢ ✦ Technology Cisco
      ◢ ✦ Field Configuration Group wlOlsr
           ✦ Attribute wlOlsrIpGateway
           ✦ Attribute wlOlsrInternetGateway
      ◢ ✦ Table Configuration Group ethNetworkIpsecTable
           ✦ Attribute ethNetworkIpsecTable
           ✦ Attribute Mode
           ✦ Attribute Source IP
           ✦ Attribute Destination IP
```

Figure 4: Network technology model.



Property	Value
Identification	.1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.1
Name	ethNetworkIpsecTable

Figure 5: Attributes property window.

4 EVALUATION OF THE APPROACH

In this section we present an experiment conducted using our DSL approach. It, has been compared with the previous method, which is the specialization of classes, currently used for the definition of a new network technology.

This experiment was conducted according to all steps proposed by Wohlin (Wohlin et al., 2000). It was defined as: (i) analysis of the DSL developed and described in Section 3, (ii) with the purpose of increasing advantages, (iii) with respect to efficiency (time) and easiness (number of errors inserted in the source code), (iv) from the points of view of ten developers who develop integrated networks management system, (v) in the context of the real company.

4.1 Planning

The planning phase was divided into six steps described in the following subsections:

4.1.1 Context Selection

The experiment has been performed in real company, whose name was omitted for purposes of confidentiality. It involved the participation of ten developers who know the integrated network management system.

4.1.2 Formulation of Hypotheses

```
extField = new NodeExtensionField();
extField.setFiledSetName("wlOlsr");
extField.addFieldValue("wlOlsrIpGate
way",session.getOID(".1.3.6.1.4.1.13
727.2300.2.1.1.2.1.1.0"));

extField.addFieldValue("wlOlsrIntern
etGateway",session.getOID(".1.3.6.1.
4.1.13727.2300.2.1.1.2.1.2.0"));
ne.addExtensionField(extField);
```

Figure 6: Code generated to the field configuration group.

```
NodeExtensionTable extTable = new
NodeExtensionTable();
extTable.setFiledSetName("ethNetwork
IpsecTable");
TableValues table = new
TableValues();
table.addColName(".1.3.6.1.4.1.13727
.2300.2.1.1.3.3.1.1",
"ethNetworkIpsecTable");
table.addColName("1.3.6.1.4.1.13727.
2300.2.1.1.3.3.1.2 ", "Mode");
table.addColName(".1.3.6.1.4.1.13727
.2300.2.1.1.3.3.1.3 ", "Source IP");
table.addColName(".1.3.6.1.4.1.13727
.2300.2.1.1.3.3.1.6", "Destination
IP");
session.getTable(table);
```

Figure 7: Code generated to the table configuration group.

The questions, metrics and hypotheses defined for this experiment were:

- Question 1 (Q_1): Is the definition of a new network technology using DSL approach more efficient than using specialization of classes approach?
   - ○ Measure 1 (M_1): Time (t) spent by participants to define a new network technology
   - ➢ Null Hypothesis (H_{10}): There is no significant time difference in the definition of a new network technology using DSL or specialization of classes approach.
   - ➢ Alternative Hypothesis (H_{11}): The time spent by participants to define a new network technology is shorter when DSL approach is used.
   - ➢ Alternative Hypothesis (H_{12}): The time spent by participants to define a new network technology is shorter when specialization of classes approach is used.

- Question 2 (Q_2): Does the DSL approach reduce the participants chance of making mistakes in the source code when defining a new network technology?
    - Measure 2 (M_2): Number of problems (p) found in the source code used to define a new network technology
    - Null Hypothesis (H_{20}): There is no significant difference in the number of problems found in the source code when the DSL or the specialization of classes approach is used.
    - Alternative Hypothesis (H_{21}): The number of problems found in the source code is smaller when the DSL approach is used.
    - Alternative Hypothesis (H_{22}): The number of problems found in the source code is smaller when the specialization of classes approach is used.

4.1.3 Independent Variables

This experiment had the following independent variables: 1) the integrated network management system; 2) Eclipse environment; 3) the way the development of integrated network management system in a real company is done today, which is specialization of classes to define a new network technology; 4) Java programming language; 5) DSL developed; 6) specifications of two new network technologies.

4.1.4 Dependent Variables

The dependent variables are, as follows: 1) efficiency, which is related to the time taken to define a new network technology and 2) the number of problems found in the source code generated due to mistakes made by the participants during the definition of a new network technology.

4.1.5 Experiment Design

The experimental model used was factor two paired treatments (Wohlin et al., 2000). In this experiment, the factor was the use of an approach to define a new network technology, while the treatments were the applied approaches – DSL and specialization of classes.

4.1.6 Instrumentation

Participants received the following materials for the execution of the experiment: one document specifying a new network technology; Eclipse EE configured with the integrated network management system; Eclipse Modeling with DSL plug-ins; one guidebook on how to create the DSL model and to request to generate code from the model; one guidebook on how to specialize the classes and Data Collection Form. In this classes specialization guidebook, classes had been implemented previously. Therefore, the participants only had to specify the network technology related. Also, in Data Collection Form, the participants had to report the time spent to develop the application, the interruptions due to doubts and errors, and their opinions and suggestions about the use of DSL and specialization of classes.

4.2 Operation

After defining and planning the experiment, its operations were carried out in two steps: Preparation and Execution.

4.2.1 Preparation

At first, the participants filled a form reporting their experiences with the concepts and technologies used in the experiment. Afterwards, the participants were trained in specialization of classes and in our approach to define a new network technology. After training, the participants were able to carry out the experiment tasks.

4.2.2 Execution

Before starting the execution of the experiment, the participants were positioned into the blocks and received the materials referent to Task 1. Each participant had access to an individual computer equipped with the tools required for the application development.

The tasks of the two blocks can be seen in Table 1. In task 1, while block 1 developed application 1 applying specialization of classes, block 2 applied DSL to the same application. Then, in task 2, while block 1 developed application 2 applying DSL, block 2 applied specialization of classes to the same application.

Table 1: Tasks of the blocks.

		Block 1	Block 2
Task 1	Application 1	Specialization of classes	DSL
Task 2	Application 2	DSL	Specialization of classes

When all participants were commanded to execute Task 1, they started measuring the time. Block 1 used Eclipse IDE to implement source-code to a new network technology and block 2 used the DSL developed to create a model and generate code automatically to a new network technology. When they finished the implementation, they stopped measuring the time. Then, they executed the implementation test to verify whether it was developed as expected. If the test showed a problem, the participants had to report it in the Data Collection Form and reinitialize to measure the time used to fix the problem found. This way, the time measured corresponded to the time spent to implementation correctly and not to test the application. Task 2 was performed in a similar way to Task 1, but now block 1 used the DSL developed and block 2 used Eclipse IDE. In the end, the participants commented the difficulties and advantages of applying each approach.

5 ANALYSIS OF DATA

The experiment data is presented in Table 2. In general, the groups developed the tasks satisfactorily and the collected data was within the expected limits. This means that the treatments were executed correctly and in accordance with the planning. The analysis of data is divided into two subsections: Descriptive Statistics and Hypotheses Testing.

5.1 Descriptive Statistics

In Table 2, it can be seen that the participants of the DSL approach spent less time to develop an application in comparison to the use of specialization of classes approach, i.e., approximately 45.5% against 54.5% (it can be seen in the last line of the Table 2). Analyzing the time spent for each participant both in the specialization of classes and in the DSL approach, four of them took the same time both approaches and six of them took less time using the DSL approach. According to the feedback provided by the participants in a form, this result is due to the fact that, in the DSL approach the code is generated correctly and automatically. On the other hand, when the participants developed the application applying the specialization of classes approach, they spent more time fixing the problems found. Moreover, most of the participants reported that they found it easier to specify the new network technology using the DSL approach that the model created using the DSL approach provides a broader view of the new network technology and that it is necessary to know Java language to use specialization of class approach.

Regarding problems, 100% of them occurred in the specialization of classes approach. The problems reported by the participants were two: typing mistakes in specifications of new network technology and Java code error.

5.2 Hypotheses Testing

The objective of this section is to verify any degree of significance, that is, whether it is possible to reject the null hypothesis based on the data set obtained. As we had defined two null hypotheses, this section is divided into two: Hypotheses Testing Time and Hypotheses Testing Problems.

5.2.1 Hypotheses Testing Time

Since some statistical tests are only applicable if the population follows a normal distribution, we applied the Shapiro-Wilk test to verify whether the experiment data departs from linearity before choosing a proper statistical test. All the tests were executed with the support of Action Tool (Estatcamp, 2013), which is the software for statistical analysis, integrated with MS Excel.

When the result of the Shapiro-Wilk test on a data set is smaller than 0.05 that means the chance of data following a normal distribution is less than 5%. Thus, it is considered, statistically, that the data did not follow a normal distribution. The result of the Shapiro-Wilk test considering the time spent using specialization of classes approach was 0.6726 and 0.3606 using DSL approach. Since the result was greater than 0.05 using both approaches, it can be stated with a confidence level of 95% that the experiment data related to the time spent in application development is normally distributed.

Once the data regarding time collected in the experiment is normalized, we decided to apply the Paired T-Test to the experiment data to verify the hypotheses of the experiment Q_1. According to Paried T-Test, when the result is less than 0.05, it means that the chance of both sets of data being statistically similar is less than 5%. So in this case, the null hypothesis should be rejected. The result of this test was 0.1474. Since 0.1474 is greater than 0.05, with a confidence level of 95%, we can state there is no difference between the time spent using the specialization of classes approach and the DSL approach. Therefore, the null hypothesis (H_{10}) must be accepted.

Table 2: Experiment data set.

Task	Specialization of Classes			DSL		
	Participant	Time(min)	Problems	Participant	Time(min)	Problems
1	S1	17	2	S6	6	0
	S2	16	0	S7	15	0
	S3	13	1	S8	11	0
	S4	17	0	S9	15	0
	S5	12	0	S10	16	0
	Average	15	0,6	Average	12,6	0
2	S6	10	1	S1	13	0
	S7	15	0	S2	14	0
	S8	11	1	S3	9	0
	S9	15	1	S4	13	0
	S10	22	4	S5	12	0
	Average	14,6	1,4	Average	12,2	0
Average		14,8	1		12,4	0
Percentage		54,5	100		45,5	0

5.2.2 Hypotheses Testing Problems

Similarly, we used the Shapiro-Wilk test, and its result on the data of the number of problems in the source code was 0.0068 using the specialization of classes approach; and it was impossible to calculate it using the DSL approach because all data is equal to zero. Since the result was less than 0.05, it can be stated, with a confidence level of 95% that the data related to the number of problems using specialization of classes approach does not follow a normal distribution.

Thus, as the number of problems seen in the data collected in the experiment is not normalized, we decided to apply the Wilcoxon Signed Rank test to the experiment data to verify the hypotheses of the experiment Q_2. According to Wilcoxon Signed Rank test, when the result is less than 0.05 it means that the chances of both sets of data are statistically similar – less than 5%. Therefore, the null hypothesis should be rejected. The result of this test was 0.0235. Since result is less than 0.05, with a confidence level of 95%, the null hypothesis (H_{20}) was refuted and the H_{21} was accepted. Therefore, statistically, we can conclude that the DSL approach reduces the number of problems compared to the specialization approach.

6 THREATS TO VALIDITY

6.1 Internal Validity

Experience level of participants: different levels of knowledge of the participants could affect the collected data. To mitigate this threat, we divided the participants into two balanced blocks according to their knowledge regarding Java programming, design patterns, MDD and DSL. All participants had prior experience in specialization of classes and they were trained in the DSL approach.

Number of participants: it can be argued that the experiment was applied with few participants. However, they are developers who work in a real company, participating directly or indirectly in the development of integrated network management system. Therefore, they are able to conduct the experiment and make pertinent comments regarding the two approaches applied.

Environment of the experiment: different computers and installations could affect the recorded timings. However, the participants used the same hardware configurations, operation system and software configurations.

6.2 External Validity

Interaction between configuration and treatment: the exercises performed in the experiment were an example of the specification of a new network technology. Only two specifications were developed and both had similar complexity. To mitigate this threat, the tasks were designed considering a real network technology.

6.3 Conclusion Validity

Measure reliability: it refers to metrics used for measuring development effort. To mitigate this threat we only have the time spent, which was captured in forms filled by the participants.

Low statistic power: the ability of a statistic test in revealing reliable data. To mitigate it, we applied two tests: T-Tests to statistically analyze the time spent to develop the application, and Wilcoxon Signed-Rank test to statistically analyze the number of problems found in the application.

7 RELATED WORKS

Santosh and Madanagopal, 2009, proposed a Generic Proxy Agent framework to facilitate management of heterogeneous network elements. This framework consists of two components, namely an SNMP Proxy Agent and Mediation device. The mediation device is a separate software module which actually communicates to Network Elements by converting SNMP requests into proprietary protocol messages and vice versa. A proxy agent performs the translation of SNMP requests into non-SNMP requests and vice versa. The developed Generic SNMP Proxy Agent framework operates with any type of Network Element (NE) from different vendors, equipped with its proprietary protocols. It also provides the flexibility to add them in the network. The developed framework assures independent communication and management of multiple heterogeneous NEs. The framework maintains the list of MIBs and allows listed MIBs and its OIDs to be registered in a limitless quantity as required by the Proxy Agent.

Ma-kun Guo et al., 2010, proposed a kind of network management system based on Extensible Markup Language (XML) which has advantages in platform-free, efficiency and flexibility to solve problems related to extensibility configuration management and efficient application development processes. This approach proposes one kind of model of SNMP MIB to transform it in XML view and a common network management system. Technology of the system can be used as network management and software system development reference. The rapid development of the network, of SNMP scalability, efficiency, safety and other issues of defects, XML-based network management technology turns into a great opportunity for development.

Our approach was developed in an integrated network management system developed by the real company. This system aims at managing different types of networks and technologies such as: IP, IP/MPLS, WiMAX, GPON, transport networks (WDM, SDH, PDH, optical modems and SHDSL) and plant telecommunications infrastructure. Using the DSL approach it is possible to specify new network technologies in a high level of abstraction, i.e., in a model.

8 LIMITATIONS

Only one function of the integrated network management system was chosen to integrate the DSL developed. Thus, the DSL developed does not cover the entire system. DSL has been developed in Eclipse platform, therefore it is necessary that the developer knows the Eclipse development environment.

9 CONCLUDING REMARKS AND FUTURE WORK

In this paper we proposed an approach that uses DSL to facilitate the specification of a new network technology in the integrated network management system, and it was used by a real company. In this approach the new network technology is defined in a model and the corresponding code is generated automatically. Our approach promotes the reusing everything from the model up to code generation. The code is generated from a template and the template, in turn, allows for the specification of the model, which was made according to the rules defined in the metamodel.

In addition, since the models are created accordingly to the metamodel, the rules have already been validated at the time of creation of the model. The possibility of mistakes existing in the Java code generated from the model is smaller than when implemented manually because the templates used in the generation of the code have been tested previously.

The reports of the participants of the experiment emphasized the easiness of use of DSL, both in the specification of new network technology and in a broader view of the requirement that DSL provides, as in the generation of Java code from the specified model without having the concern with writing Java code.

The results obtained in the experiment were analysed in descriptive statistics and hypotheses testing. According to the descriptive statistics is possible to observe that the time spent to develop the application using DSL approach was shorter when specialization of classes approach was used. However, in the hypotheses testing time according to Paried T-Test the null hypothesis was accepted. This hypotheses testing time result can indicate that there were few participants in the experiment and the time data set did not show difference.

Regarding problems, in both the descriptive statistics and hypotheses testing, the errors in the source code did not occur when the DSL approach was used.

Although the null hypothesis was accepted in the hypotheses testing time, all other results obtained such as the descriptive statistical analysis, code generated automatically without present errors and all comments reported by the participants regarding the ease of use of DSL, it encourages the development of new DSLs to others functions of the integrated network management system.

In future works we intend to analyze the functions of integrated network management systems and propose the development of other DSLs. And other modelling resources and MDD technology could be useful tools to allow a more graphical interface.

REFERENCES

Acceleo. 2013. Available in: http://www.eclipse.org/acceleo/ and in: http://www.acceleo.org/>.

Antkiewicz, M., Czarnecki, K., Stephan, M., 2009. Engineering of Framework-Specific Modeling Languages. *IEEE Transactions on Software Engineering,* v. 35, n. 6, p. 795-823.

Antkiewicz, M., Czarnecki, K. Framework-specific modeling languages with round-trip engineering. In: NIERSTRASZ, O. et al. (Ed.). Model Driven Engineering Languages and Systems (MoDELS 2006). [S.l.]: Springer Berlin / Heidelberg, 2006. (Lecture Notes in Computer Science, v. 4199/2006), p. 692–706.

Deursen, A. V., Klint, P., Visser, J., 2000. Domain-specific languages: An annotated bibliography. SIGPLAN Notices - ACM Press, v. 35, n. 6, p. 26–36.

Djukic, V., Lukovic, I., Popovic, A., 2011. Domain-Specific Modeling in Document Engineering. In: *Federated Conference on Computer Science and Information Systems* – FedCSIS. Szczecin, Polônia. Proceedings… Washington: IEEE Computer Society. p. 817-824.

Durelli, R. S., 2011. Uma abordagem apoiada por linguagens específicas de domínio para criação de linhas de produtos de software embarcado, UFSCar.

EMF. 2013. Eclipse Modeling Framework. Available in: <http://www.eclipse.org/modeling/emf/>.

ESTATCAMP, 2013. Portal Action. Available in: HTTP://www.portalaction.com.br.

Fowler, M., 2005. Language Workbenches: The Killer-App for Domain Specific Languages? [S.l.]: martinfowler.com. Available in: <http://www.martinfowler.com/articles/languageWorkbench.html>.

Fowler, M. et al., 1999. Refactoring: Improving the Design of Existing Code. [S.l.]: Addison Wesley.

France, R., Rumpe, B., 2007. Model-Driven Development of Complex Software: A Research Roadmap. In: *29th International Conference on Software Engineering – Future of Software Engineering – ICSE*. Minneapolis, USA. Proceedings… Washington: IEEE Computer Society 2007.p.37-54.

Gronback, R. C. Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit. 1. ed. Addison-Wesley Professional, 2009. 736 p.

Hutchinson, J., Whittle, J., Rounccefield, M., Kristoffersen, S., 2011. Empirical Assessment of MDE in Industry. In: *33RD International Conference on Software Engineering – ICSE*. Honolulu, HI, EUA. Proceedings…New York: ACM. p. 471-480.

Lucrédio, D., Fortes, R., Almeida, E., Meira, S. Performing Domain Analysis for Model-Driven Software Reuse. In: *International conference on Software Reuse: High Confidence Software Reuse in Large Systems*, 10th, 2008, Berlin. Proceedings… 2008. p. 200-211.

Ma-kun Guo, Yi-min Yu, Min Wang, Qi Yu, Research and Implementation of Network Management System Based on XML View. 2010. In: *International Conference on Logistics Engineering and Intelligent Transportation Systems (LEITS)*, IEEE.

Mellor, S. J., Clark, A. N., Futagami, T., 2003. Model-Driven Development. IEEE Software, v.20, n.5, p. 14-18, Setembro 2003.

Santosh, S., Chavan and R. Madanagopal. 2009. Generic SNMP Proxy Agent Framework for Management of Heterogeneous Network Elements. In: *First International Conference on COMmunication Systems And NETwork*s. Pages 331-336. IEEE Press Piscataway, NJ, USA.

Schmidt, D. Guest Editor's Introduction: Model-Driven Engineering. IEEE Computer, v.39, n. 6, p. 25-31, 2006.

Völter, M. MD Best Practices, 2008. Available in: <http://www.voelter.de/>.

Wohlin, C. et al., 2000. Experimentation in Software Engineering: An Introduction. [S.l.]: Kluwer Academic Publishers.