A COMPARATIVE STUDY ON MODEL-DRIVEN REQUIREMENTS ENGINEERING FOR SOFTWARE PRODUCT LINES

David Blanes
dblanes@dsic.upv.es

Emilio Insfran
einsfran@dsic.upv.es

ISSI Research Group, Department of Information Systems and Computation
Universitat Politècnica de València, Spain.

Abstract: Model-Driven Engineering (MDE) and Software Product Lines (SPL) are two software development paradigms that emphasize reusing. The former reuse domain knowledge is represented as models and model transformations for product development, and the latter reuse domain knowledge is represented as core assets to produce a family of products in a given domain. The adequate combination of both paradigms can bring together important advantages to the software development community. However, how to manage requirements during a model-driven product line development remains an open challenge. In particular, the Requirements Engineering (RE) activity must deal with specific properties such as variability and commonality for a whole family of products. This paper presents a comparative study of eleven approaches that perform a MDE strategy in the RE activity for SPL, with the aim of identifying current practices and research gaps. In summary, most of the approaches are focused on the Domain Engineering phase of the SPL development, giving less attention to the Application Engineering phase. Moreover there is a lack of coverage of the Scoping activity, which defines the SPL boundaries. Several approaches apply some model transformations to obtain architectural and application requirements artifacts. Regarding the tool support for requirements specification and management, we found that most of the approaches use only academic prototypes. Regarding the validation of the approaches, the use of Case Studies as a *proof of concept* was the most commonly used method; however, there is a lack of well-defined case studies and empirical studies to improve the proposals.

Keywords: Requirement Engineering; Software Product Lines; Model-Driven Development.

1 INTRODUCTION

Reuse is a key factor in reducing costs and in improving the quality of software product properties such as security, reliability, performance, etc. On the one hand, in recent years, the Software Product Line (SPL) approach has emerged as a new paradigm to build software based on an intensive reuse policy. SPL Engineering has proven to be a very useful approach for developing diverse software products and software-intensive systems at lower costs, in a shorter time, and with higher quality (Pohl, Böckle, Van Der Linden, 2005). On another hand, Model-Driven Development (MDD) is a trend that is drawing attention in the software development community. MDD is an approach to software development that proposes the use of models at various levels of abstraction, and model transformations as it main artifacts. The use of models as the main citizens in product development gives many advantages to developers, for example by increasing the abstraction level. Moreover, model transformations allow the conversion of one source model into another target model, and thereby improves reuse in the software development process. Both of these paradigms are based on an intensive reuse strategy, so their appropriate combination can bring together important advantages to the software development community.

However, the combination of both MDD and SPL to produce software products requires the identification of new ways to define the stakeholder needs. Traditional software development methods are inadequate to address the challenges of rapid change and the growth of requirements. In addition, in the context of SPL, requirements must also capture specific properties such as variability, commonality or evolution. Moreover, the Requirements Engineering (RE) activity is applied to the different activities of the SPL development: Scoping, Domain Engineering, and Application Engineering. The *Scoping* is the activity that defines the SPL boundaries according to the potential products of interest in the domain. The result is the set of potential products that could be produced in the SPL, and the main features of the system. This information is used as input for the *Domain Engineering*, which is the activity that defines a Core Asset Base (a set of Core Assets associated with the SPL, to support a reuse strategy). This Core Asset Base is used in the *Application Engineering* activity, which creates a product by using a selection of Core Assets. A Core Asset is a reusable, software or non-software, artifact or resource that is used on one or more products. The reusable nature of a Core Asset implies a higher level of complexity with respect to the traditional RE activity due to the fact that a new property must be identified and defined in these core assets: the *variability* of the Core Asset.

In the last few years, several approaches have been proposed to identify and define this variability in the RE activity with the aim of dealing with these new specific needs. These approaches have many similarities and differences since they propose a different number of activities and artifacts in order to identify and model a requirements specification. The purpose of this paper is to study the use of RE techniques and the use of models and model transformations in the RE activity during SPL development. This analysis will help SPL developers to identify the strengths and weaknesses of the approaches regarding MDD characteristics. This paper is organized as follows: Section 2 introduces the related work; Section 3 presents a comparison of the RE approaches for SPL based on their support to the SPL activities, the RE tasks, the MDD coverage, the degree of automation with tools, and the type of validation performed. Finally, in Section 4 conclusions and further work are presented.

2 RELATED WORK

In the last years, several approaches have been proposed to deal with the RE activity of the SPL development. In this section we discuss recent works that analyze and compare some of these proposals.

Firstly, Kovačević *et al.* (Kovačević, Aférez, Kulesza, Moreira, Araújo, Amaral, 2007) present a survey about the state of the art in Requirements Engineering for SPL and Model-Driven RE. Two separate comparisons were performed. Firstly, MDD approaches were analyzed, focusing on differences between non-aspect and aspect-oriented approaches. Secondly, several RE approaches for SPL were analyzed. The authors defined common criteria for both comparisons (evolvability, verification, trade-off analysis, scalability, traceability, and tool support), and then specific criteria for each separate comparison (MDD properties and SPL properties). On the one hand, regarding the MDD properties we found: the language used to model the requirements models and the type of model transformations support used. On the another hand, regarding the SPL properties we found: the support for Domain Engineering, Application Engineering, the type of Adoption Strategy, and the Validation of the proposal. However, this survey does not analyze the RE proposals applied to the SPL paradigm using MDD techniques. Moreover, this work does not analyze which RE activities are covered by the approaches and the different types of models used. Finally, this work highlights the fact that most SPL approaches do not define a coherent and clear set of requirements and variation models and do not define the respective relationships between them remarking the need of well-defined traceability strategies.

Nicolas and Toval (Nicolás, Toval, 2009) performed a Systematic Literature Review (SLR) to study the generation of textual requirements specifications starting from models. The SLR was conducted with two research questions and assessed 30 papers from the last five years. The research questions were: 1) What value can be drawn from the literature with regard to the generation of requirements specifications (textual requirements and requirements documents) from software engineering models?; 2) What techniques have been addressed in this field?. This SLR was not focused on SPLs, however one section analyzes the product requirements derivation with a new research question: Which approaches take the requirements derivation from SPL models into account? In order to analyze this question 6 papers were selected; the analysis of these papers shows that the combination modes (approaches which propose algorithms, rules or patterns to generate textual requirements starting from models) were generative, the opposite to the integrative mode (studies which do not provide algorithms, rules or patterns to generate requirements from models, but rather a kind of open-ended guide to relate models and textual requirements). The study reveals that the papers have a requirements scope (approaches that deal with the generation of requirements or sets of requirements), but do not address the requirements documents in which the requirements should be placed; in contrast they do not present a documental scope (studies which concentrate on the manual, automatic or semi-automatic generation of requirements documents). The initial models were feature and variability models, and the target models were natural language and formal notations. Finally, the authors argue that the research community should pay greater attention to the product derivation process.

Alves *et al.* (Alves, Niu, Alves, Valença, 2010) present a SLR about RE for SPL. The paper is focused on the assessment of research quality, the synthesis of evidence to suggest relevant implications for future practice, and the identification of research trends, open problems, and areas for improvement. This SLR was conducted with three research questions and assessed 49 studies dated from 1990 to 2009. The research questions were: 1) What SPL RE methods and/or tools are available to practitioners?; 2) How much evidence is available to support the adoption of the proposed methods?; and, 3) What are the limitations of current SPL RE methods?. This review exposes that most of the approaches have limitations in terms of the validity and credibility of their findings. Moreover, the study reveals a lack of tool support, and guidance for adoption of the proposed methods. However, this work did not analyze the use of requirements in the different SPL development activities in sufficient detail. It would have been interesting to have analyzed factors such as which techniques were used, or which kinds of models were employed during the different RE activities.

Finally, these related works were published in the last five years showing an increasing interest in RE approaches for SPL development by the software engineering community. In addition, there has also been an increased interest in the application of MDD techniques in RE. Currently there are no studies that focus on the analysis of the current degree of use of MDD approaches for RE activities in SPL development.

3 COMPARATIVE STUDY

In this section, we present a comparative analysis of the most important Requirements Engineering proposals that have been published to support the development of software products following the Software Product Line and MDD approaches. The purpose of this study is to collect together the current knowledge about RE techniques in MDD and SPL in order to identify common practices and research gaps with the aim of suggesting areas for further investigation. This has been summarized in the following research question: “What requirements engineering techniques have been employed in model-driven development approaches for Software Product Lines and what is their actual level of automation?” This research is focused on analyzing papers that present MDD approaches for SPL. We used three digital libraries as primary sources: IEEExplore, ACM Digital Library, and ScienceDirect (Elsevier), in addition we used the Google Scholar searcher. These digital libraries include specialized conferences and workshops in the area such as: Software Product Line Conference (SPLC), International Conference on Requirements Engineering (RE), Requirements Engineering: Foundation for Software Quality (REFSQ), and Model-Driven Requirements Engineering Workshop (MoDRE). As the inclusion criteria for relevant contributions, we have only considered for our study published papers that propose methods to cover the RE activity in SPL development using MDD techniques.

In order to analyze the selected papers, we have defined five criterions to perform the comparative study. The first criterion analyzes the support to the SPL development by the RE approaches. The second criterion is the Requirements Engineering tasks that where covered by the RE approaches, which artifacts where employed, the type of requirements (functional or non-functional), and what type of traceability was supported. The third criterion is the model-driven coverage, the desired purpose of this adoption and, if followed up, which model and input models, language and transformation type is used. The fourth criterion is the automatic support to the approach with tool. Finally, the last criterion analyzed is the type of validation provided by the approach. In this section there is a brief explanation of the results obtained from the comparison for each aspect involved. Following this, we introduce each of these criteria and the results obtained from them in detail.

a. Software Product Line Support

The aim of this criterion is to examine the given support to the SPL development by the RE approaches. We analyzed five sub-criterions: activities, adoption strategy, Scoping tasks, Domain Engineering tasks, and Application Engineering tasks.

We consider three activities of the SPL process for this study: Scoping, Domain Engineering, and Application Engineering. *Scoping* is the activity concerned with the establishment of the SPL boundaries and the reusability strategy. This activity covers the analysis of which products will be included in the SPL, based on cost and reusability analysis. The *Domain Engineering* aims to develop a requirements specification for the common PL and its related variability, whereas the Application Engineering aims to develop a requirement specification for a single product.

In the adoption strategy sub-criterion ways in which the software product development is supported by these approaches are analyzed. We use three *adoption strategies* proposed by Krueger (Krueger, 2001): the Proactive, Extractive, and Reactive approaches. In the *Proactive* strategy, the organization analyses, designs, and implements a SPL from scratch to support the full scope of products needed on the predictable horizon. In the *Reactive* strategy, the organization incrementally grows an existing SPL when the demand arises for new products or new requirements on existing products. In the *Extractive* strategy, the organization extracts existing products into a single SPL.

Regarding the *Domain Engineering tasks*, we consider: conceptual modeling, commonality and variability modeling, feature modeling and scenario modeling. The *Conceptual modeling* includes activities to identify, define, and organize the concepts that are relevant to the domain and their mutual relationships, in order to facilitate a precise and concise description of the domain. We consider *Commonality and Variability modeling* as activities to identify similarities and differences between the requirements. The *Feature modeling* includes activities to identify, study, and describe features relevant in a given domain. Finally, in the *Scenario modeling* we have found activities to describe and model the run-time behavior of members of the system family.

In the *Application Engineering* process we consider two tasks: derivation and delta identification. We consider that one approach supports the requirements *derivation*, if it provides a mechanism to obtain a requirement specification for a single product from the domain requirements specification. If the approach includes a mechanism to identify and model new requirements in the application requirements specification, then it supports the *Delta Identification*.

The last sub-criterion analyzed is the Scoping tasks. We consider three levels of *Scoping*: Product Portfolio Scoping, Domain Scoping, and Asset Scoping. The *Product Portfolio Scoping* aims to identify the particular products that ought to be developed as well as the features that they should provide. The *Domain Scoping* is the task of bounding the domains that likely to be relevant to the product line. The *Asset Scoping* aims at identifying the particular (implementation) components that should be developed in a reusable manner.

The results obtained are shown in Table I. The approaches mainly support the Domain Engineering activity as the Application and Scoping activities were considered only in a few approaches.

Another commonality observed is that the approaches do not usually describe the adoption strategy followed. Generally a Proactive strategy was followed by the approaches, implying that the approaches typically start a SPL from scratch without considering existing assets or legacy systems. For this reason, the Proactive strategy is considered the most expensive and risk-prone (Krueger, 2001). An interesting alternative is to combine the Proactive with a Reactive strategy, which allows the addition of new products to an existing product-line, and the identification and integration in the Domain Requirements Specification of deltas potentiate this strategy. Moreover, the approaches that provide model transformations permit this strategy, with the adaption of an existing SPL to the new requirements. In only one approach was there not sufficient support for the Proactive strategy, this is the case of Dhungana *et al.*, where is it assumed the need of an existing SPL, so it supported the Reactive strategy only. Other special cases are those of Corriveau *et al.* (Corriveau, Bashardoust, Radonjic, 2011), and Guelfi & Perrouin (Guelfi, Perrouin, 2007), which are approaches focused on testing and requirements analysis, so they do not follow an adoption strategy to develop a SPL.

TABLE I
SOFTWARE PRODUCT LINE SUPPORT

Approach	Activities	Adoption Strategy	Domain Engineering Tasks	Application Engineering Tasks	Scoping Tasks
Alférez *et al.*	Domain, Application	Proactive, Extractive	C&V, FM, SM	Derivation	None
Bragança & Machado	Domain, Application	Proactive, Reactive	C&V, FM, SM	None	None
Coelho & Batista	Domain	Proactive, Reactive	None	None	None
Corriveau *et al.*	Domain	Not applicable	None	None	None
Dhungana *et al.*	Domain	Reactive	None	Delta ident.	None
DREAM	Domain, Scoping	Proactive, Extractive	CM, C&V, SM	Derivation	Portfolio, Domain
Guelfi & Perrouin	Domain, Application	Not applicable	SM	Derivation	None
I-GANDALF	Domain	Proactive, Reactive	C&V, FM	None	None
OVM-A	Domain, Application, Scoping	Proactive, Reactive, Extractive	C&V, FM	Delta ident.	Portfolio, Domain, Asset
SIRENspl	Domain	Proactive, Reactive	CM, C&V, FM, SM	None	None
SREPPLine	Domain, Application, Scoping	Proactive, Reactive, Extractive	C&V, SM	Delta ident.	Portfolio, Domain, Asset

C&V: Commonality & Variability Modeling, FM: Feature Model, SM: Scenario Modeling

Regarding the Domain Engineering tasks, most of the approaches support the Commonality & Variability Analysis. Among these papers, most of the approaches use Feature Modeling to identify common and variable requirements. The Orthogonal Variability Model (OVM) is an alternative to the C&V modeling used in OVM-A (Pohl, Böckle, Van Der Linden, 2005) and SREPPLine (Mellado, Fernández-Medina, Piattini, 2007). These approaches use Variation Points and define traceability from the requirement models to the OVM. The use of the OVM representation has the advantage of keeping the functional requirement and the variability separate.

Regarding the Application Engineering, we only found five approaches that supported this activity. Alférez *et al.* (Alférez, Kulesza, Weston, Araujo, Amaral, Moreira, Rashid, Jaeger, 2008) and DREAM (Moon, Yeom, Chae, 2005) allow the creation of Application Requirement Specifications from the Domain Requirements Specification. However, they do not capture requirement deltas. Guelfi and Perrouin provide a product derivation technique based on an analysis model based on UML, OCL, and Use Cases, that implicitly defines the product line variability and the SPL boundaries by means of constraints that forbid undesired products. Dhungana *et al.* (Dhungana, Seyff, Graf, 2011) propose a solution based on the tool DOPPLER, where a simulation is created from simple final-user decisions. This simulation allows developers to identify new requirements in an existing product line. OVM-A and SREPPLine provides the most complete solution, since they include the analysis and identification of requirement deltas.

Finally, we found only three papers that integrated the scoping activity in the proposals. This fact evidences a lack of integration between scoping and the requirement engineering proposals.

DREAM start with an elaboration of a Domain Terminology model. This model contains the main terms used in the SPL and is a way to do the Domain Scoping. This model is used to identify the Primitive Requirement (PR), which is a new concept defined by DREAM as a transaction that has an effect on an external actor. Its granularity is in between that of a Use Case and an atomic operation of a Use Case; the purpose of a PR is to make the domain requirements more concrete and to discover the variability and rationale of the domain requirements.

OVM-A supports Product Portfolio, Domain Scoping, and Asset Scoping. The Portfolio Analysis allows a systematic evaluation of the Product Portfolio. During the analysis, each product (or product type) is rated according to two variables and thereby its location in a two-dimensional matrix is determined. The Asset Scoping and Domain Scoping are accomplished with the commonality and variability analysis.

The last approach that supports the Scoping is SREPPLine. This approach starts with the identification of security features, which can be considered as a way to do Product Portfolio Scoping. Once the security features are defined, Security Asset Scoping is performed. In this activity the security assets for each security feature, and the dependences between assets are identified. After this the security threats scoping and risk assessment are executed. In order to derive security requirements, each pair of asset and security objectives were analyzed for their possible relevance, together with their related threats which imply more risk, so that suitable security requirements, or the suitable package of security requirements, that adequately mitigate these threats at the necessary levels with regard to the result of the risk assessment activity, were identified.

We conclude that there is a need for the integration of Scoping activity and Requirements Engineering. For example, between the analyzed approaches only SREPPLine provides one solution to integrate risk assessment. Other authors like John (John, Eisenbarth, 2009) pointed out this problem, where it is said that although scoping is commonly related to requirements engineering for product lines, it is often executed in a solitary and upstream development step. The integration of activities like risk assessment or cost estimations in the RE approaches could help to set properly the SPL boundaries.

b. Requirements Engineering Support

The goal of this criterion is to analyze the coverage of the Requirement Engineering process. We analyzed four sub-criterions: RE tasks, RE artifacts, if these artifacts represent functional or non-functional requirements, and RE traceability.

The first sub-criterion is the analysis of the different *RE tasks*. We use the classification proposed by (Cheng, Atlee, 2007) that categorize the requirements tasks in: elicitation, modeling, analysis, validation, verification, and management.

Elicitation refers to the activities performed to enable the understanding of the goals, objectives, and high-level functions necessary for the proposed software system. *Modeling* allows requirements to be expressed in terms of one or more models that document the user needs and constraints clearly and precisely. The *Analysis* consists of evaluating the quality of the requirements captured and specified in the elicitation and modeling tasks. The requirements *Validation* is supported if the approach provides mechanisms to check if the stakeholder needs are satisfied in the requirement specification. Requirements *Verification* is the process of ensuring that the system requirements are complete, correct, consistent, and clear, whereas requirements *Management* is the process of scheduling, coordinating, and documenting the RE.

The *requirement artifacts* criterion analyzes the proposed software artifacts projected by the approaches. For example, many works employ goals, scenarios, and Use Cases as a conceptual framework to identify user requirements.

The nature of these artifacts can be *functional or non-functional*. A functional requirement is a requirement that specifies a function that a system or system component must be able to perform. A non-functional requirement is a requirement that defines restrictions on a system or system component that it must satisfy, such as reliability, security, usability, etc.

Finally, we analyze the *traceability*, which refers to the ability to follow the life of a requirement either back to its origin or forward to its transformation into a design artifact. We use the classification proposed in Kovačević *et al.* (Kovačević, Aférez, Kulesza, Moreira, Araújo, Amaral, 2007) as a way of understanding two types of traceability: vertical and horizontal.

We consider *vertical* traceability as the ability to relate requirements from domain specific requirements to product specific requirements. The *horizontal* traceability is the ability to relate domain requirements to the domain architecture. It includes the mapping between variation points of these artifacts.

TABLE II
REQUIREMENT ENGINEERING TASKS

Approach	RE Tasks	RE Artifacts	Functional / Non-functional Req.	Traceability
Alférez *et al.*	Elicitation, modeling, management	Feature Model, Use Cases, Activity Diagrams, Table of Trace links	Functional	Vertical
Bragança & Machado	Elicitation, modeling	Use Case Diagram, Feature Model, Activity Diagram	Functional	Both
Coelho & Batista	Modeling	PL-AOVGraph model	Both	Horizontal
Corriveau *et al.*	Analysis, verification	User Requirement Notation	Both	None
Dhungana *et al.*	Elicitation, analysis	Decision model	Functional	None
DREAM	Elicitation, modeling, management	Domain Terminology, PR-Context matrix, PR-Use Case matrix, Domain Use Cases.	Functional	Horizontal
Guelfi & Perrouin	Analysis	UML, OCL, Textual Use Cases	Functional	Vertical
I-GANDALF	Elicitation, modeling	Goal Model, Feature Model	Both	Horizontal
OVM-A	Elicitation, modeling, management, validation, verification	Application-Requirements Matrix, Orthogonal Variability Model	Both	Both
SIRENspl	Elicitation, modeling, validation	Feature Model, Use Cases, NFR templates, Textual Requirements	Both	Horizontal
SREPPLine	Elicitation, modeling, validation	Security Use Cases, Misuse Cases, Orthogonal Variability Model	Both	Both

Table II shows the results obtained from the RE covered tasks comparison. Clearly most of the approaches cover the elicitation and modeling tasks. We found only three exceptions: that of Corriveau *et al*, which. is an approach with the goal of analyzing a requirement specification, and to perform requirements verification by using a model-driven strategy. That of Dhungana *et al.* only partially supports the requirement elicitation with a new technique based on tool assistance. In this approach, the users enter their needs in a natural language. Based on the text entered the tool identifies relevant decisions within the variability model and it displays corresponding questions to the end-user, followed by the performance of a simulation based on the decisions. This simulation is used to help the user to identify new requirements. Finally, Guelfi & Perrouin provides an analysis model, based on UML, OCL, and Use Cases, that implicitly defines SPL variability and SPL boundaries by means of constraints that forbid undesired products.

Regarding validation, OVM-A and SREPPLINE propose techniques for requirement validation; SIRENspl (Nicolás Ros, 2009) (Toval Álvarez, Nicolás, Moros, Garcia, 2002) mention this activity but do not propose any specific techniques to perform it. In OVM-A, the authors propose the ScenTED technique (Scenario based TEst case Derivation) (Reuys, Kamsties, Pohl, Reis, 2005), which is a model-based, reuse-oriented technique for test case derivation in the system test of software product families. SREPPLine covers the requirements validation with the inspection technique. Another way to check a requirement specification is to perform a verification. OVM-A can offer requirements verification with an extension that proposes a technique to check consistency based on model of checking techniques (Lauenroth, Pohl, 2008). SIRENspl propose a requirements validation activity in its guidelines, nevertheless no concrete technique is proposed. Despite these proposals, there is a lack of integration of requirement validation and verification techniques in RE approaches for SPL.

Analyzing the artifacts used, we usually found the creation of Variability Models. One widely adopted notation of the Variability Model is the Feature Model. The Feature Model represents externally visible product characteristics in a domain. We mainly found two typical notations that represented the Feature Models: FODA-based (Kang, Cohen, Hess, Novak, Peterson, 1990) or UML-based notations. Other alternatives to represent Variability Models are the Orthogonal Variability Models proposed in OVM-A (Pohl, Böckle, Van Der Linden, 2005).

Nevertheless, the variability modeling is not limited to Variability Models. Other classical notations such as Uses Cases and Goal Notations were adapted to the software product lines. For example, DREAM uses Use Cases to model functional requirements and variability. However, the inclusion of functionality and variability in the same model could produce an overload, which might have a negative effect in the legibility. A Goal-oriented notation is used in approaches like Coelho & Batista's (Coelho, Batista, 2011) (Batista, Bastarrica, Soares, Fernandes, 2008), or I-GANDALF (González-Baixauli, Navarro, Laguna, Sampaio do Prado Leite, 2009). Here a Goal Model is used as an early requirement artifact and guides the creation of the Feature Model. Other approaches combine both notations, for example in the case of Corriveanu *et al.*, where Use Cases and Goal-models are combined in the User Requirements Notation (URN). URN (Amyot, Mussbacher, 2011) (International Telecommunication Union, 2008) offers an international standard combining a goal-oriented requirements language and a scenario-based notation.

Regarding the representation of functional and non-functional requirements, we observe that the inclusion of Goal-based models allows the representation of functional and non-functional requirements. We have found other ways to represent the NFR in SPL, for example, quality templates as in SIRENspl, or security requirements as in SREPPLine. DREAM has support in its meta-model to the concept NFR; however the method is focused on scenarios that model functional requirements and its variability. OVM-A provides a meta-model that relates variations with requirement artifacts. In this context, a requirement artifact can be a functional or quality requirement, so in this case, we can consider non-functional requirement support.

Regarding traceability, the adoption of one model-driven strategy potentiates traceability support by the above approaches. If there is a vertical traceability then a relationship between domain and application requirements is defined, for example as reported by Alférez *et al.*, who proposed traceability between requirement models and the Feature Model. This traceability information is used to obtain the requirements for a specific application. Guelfi & Perrouin support vertical traceability from domain artifacts to application artifacts. The adoption of this traceability strategy is to analyze the possible configurations from the domain requirement specification. Other approaches were focused on store traceability from requirements to architectural artifacts. Coelho & Batista and I-GANDALF follow the traceability from requirements to Feature Model and, subsequently, to architectural artifacts. In DREAM the traceability is defined based on a meta-modeling approach, which traces variability from requirements to the architecture. This metamodel extends the Reusable Asset Specification (Reusable Asset Specification, 2012) proposed by the OMG. SIRENspl covers the traceability from the domain models to the textual requirements. Finally, the approaches that give support to both horizontal and vertical traceability have been described by Bragança *et al.* & Machado, OVM-A, and SREPPLine. Bragança *et al*. (Bragança, 2007) (Bragança, Machado, 2007) store information from Use Cases to Feature Models, from Feature Models to Configurations, and from a Configuration implicitly in the transformation rules. This strategy allows application requirements from the Domain Use Case models to be obtained. OVM-traces information between the Orthogonal Variability Model to domain, and application requirements; SREPPLine supports traceability in a similar way, but between the Orthogonal Variability Model and the security requirements.

c. *Model-Driven Coverage*

The goal of this criterion is to compare the use of the MDD techniques in the approaches. We include in this criterion the following sub-criterion: type of coverage, purpose, input model, output model, transformation language, and type of transformation.

The first sub-criterion is the *type of coverage* to the MDD. We consider two options: the paper proposes a meta-model, or the paper proposes model transformations. Many approaches could provide a meta-model with the aim to provide a Domain Specific Language. Considering a model as an abstraction of a real world phenomenon, the meta-model is the abstraction where the model properties are reflected. Other approaches could provide a transformation between models. Otherwise the definition of model transformations allows a target model to be obtained from a source model in an automatable way.

The second sub-criterion analyzes the aimed *purpose* with a MDD strategy. For example most of the proposals use the model transformation as refinement from the source model to the target model. Other proposals provide only a meta-model to represent the concept in a specific domain (i.e., SREPPLine defines a meta-model to represent security requirements for a SPL).

The third and fourth sub-criterion analyzes the *input and output* models used for the approaches to propose model transformations.

Another sub-criterion used to define the transformations is the *language*. In the last years, several specialized languages have been proposed in order to specify model-driven transformations (e.g., the QVT language proposed by the OMG (OMG, 2012)).

The last sub-criterion in this section analyzes the *type of model transformations*. We distinguish two types, based on the classification proposed by Mens *et al.* (Mens, 2006). When the source and target model are expressed in the same language and in the same abstraction level; the transformation is *endogenous*. When different modeling languages and abstraction levels are used to express source and target models then the transformation is *exogenous*. Moreover, we distinguish them based on the abstraction level between: *horizontal*, when the source and target model reside at the same abstraction level; and *vertical* transformations, when the source and target model reside at different levels. Additionally, we can also find *bidirectional* transformations (form source to target models, and from target to source models) or transformations in just one direction (*unidirectional* transformations).

TABLE III
MODEL-DRIVEN COVERAGE

Approach	Type of coverage	Purpose	Input	Output	Language	Type of transformation
Alférez *et al.*	Transformation	Obtain Application Requirements	Domain Use Cases, Feature Model	Application Use Cases	VML4RE	Endogenous, vertical, single direction
Bragança & Machado	Transformation	Refine requirements to software architecture artifacts	Use Cases	Architectural requirements	QVT operational	Endogenous, vertical, single direction
Coelho & Batista	Transformation	Refine requirements to software architecture artifacts	Feature Model	PL-AOV Graph	Graph transformation, ATL	Endogenous, vertical, bidirectional
Corriveau *et al.*	Transformation	Obtain a testable specification	URN (User Requirement Models)	Testable Requirement Model	Textual transformation language	Endogenous, horizontal, single direction
Dhungana *et al.*	Meta-model	Meta-model to represent Asset and its variability	None	None	None	None
DREAM	Meta-model	Meta-model for domain requirements	None	None	None	None
Guelfi & Perrouin	Transformation	Product derivation following OCL constraints	Domain requirements model, OCL restrictions	Application requirements	Imperative textual language	Endogenous, horizontal, single direction
I-GANDALF	Transformation	Refine requirement specifications	Goal Model	Feature Model	QVT Relations	Endogenous, vertical, single direction
OVM-A	Meta-model	Model the SPL variability	None	None	None	None
SIRENspl	Transformation	Obtain a textual requirement specification from domain models	Domain Model	Textual requirements specification	QVT Relations	Endogenous, horizontal, bidirectional
SREPPLine	Meta-model	Provide a repository of secure artifacts	None	None	None	None

Table III shows the results obtained for this criterion. Most of the proposals present model transformations between the models. Four proposals do not provide model transformations although they provide their meta-model specifications. These meta-model specifications allow developers to specify transformations in the future. In this line, Dhungana *et al.* supports adapting a core meta-model to different domains and to define the variability of core assets. In this way, DREAM (Moon, Chae, Yeom, 2006) provides a meta-model for representing domain requirements. Domain requirements are divided into functional and non-functional requirements. OVM-A provides a meta-model to express the variability of a SPL and SREPPLine provides a meta-model to support a security requirement repository. This meta-model is integrated with a prototype tool. Other authors present the specific model transformations. One of the uses of these transformations is to obtain application requirements from the domain requirements specifications. This is the case described by Alférez *et al.*, Coelho & Batista, and I-GANDALF. SIRENspl use the model transformations as a refactoring from Domain Models to Textual Requirements specifications. Other proposals use the model transformations to obtain application requirements from domain requirements. In Alférez *et al.* the functional requirement models and a configuration model are used to obtain the functional requirements for one specific application automatically. Guelfi & Perrouin propose a solution to obtain application requirements combining transformations and OCL constraints; another use of applying model transformation is to obtain test cases. Corriveau *et al.* generate testable models from an URN requirements specification. Regarding the transformations, we have found heterogeneous proposals. VML4RE proposes a language to compose different requirement models with a Feature Model. Some approaches follow the Model-Driven Architecture initiative proposed by the Object Management Group (OMG), specifically the QVT language. There are two implementations of the QVT language: the QVT operational, adopted in Bragança & Machado; and the QVT Relations, used in I-GANDALF and SIRENspl. These languages are quite similar; however the declarative nature of QVT Relation allows bidirectional transformations. This advantage is applied in SIRENspl, where a script is provided to make transformations in both directions. Other proposals use the Atlas Transformation Language (ATL) for example as described by Batista *et al.* Finally, we have found other experimental languages such as that presented by Corriveau *et al.* (ACL contract language), or Guelfi & Perrouin (which sketch a language to restrict the product derivation).

d. Tool support

This criterion analyzes if the approach gives tool support. The idea of this criterion is to analyze the type of tools provided by the approaches rather than to analyze all available tools. In many cases, only academic prototypes are presented (e.g. plug-ins for the Eclipse platform). If the tool is used in an industrial environment or with a commercial use, then it is considered as an industrial tool (e.g. IBM Rational DOORS (Telelogic AB, 2004)).

TABLE IV
TOOL SUPPORT

Approach	Type of provided tool (none, academic prototype, industrial)
Alférez *et al.*	Academic prototype
Bragança & Machado	Academic prototype
Coelho & Batista	Academic prototype
Corriveau *et al.*	Academic prototype
Dhungana *et al.*	Academic prototype
DREAM	Academic prototype
Guelfi & Perrouin	None
I-GANDALF	Academic prototype
OVM-A	Academic prototype
SIRENspl	Academic prototype
SREPPLine	Academic prototype

The results obtained are shown in Table IV. Alférez *et al.* provides a tool to support product derivation of requirements models in SPL by using a domain-specific language. It also supports trace link generation from features to requirements model elements, for further analysis. Bragança & Machado present a prototype based in the Eclipse Modeling Framework (EMF) and SmartQVT. The EMF project is a modeling framework and code generation facility for building tools and other applications based on a structured data model. SmartQVT is a tool set for model-to-model transformations that implements the QVT relations language in a Java language. Coelho & Batista propose a tool called MaRiSa (Mapping Requirements to Software Architecture) that implements bidirectional transformations between the requirements (represented as AOV-Graphs) and the architecture (represented in AspectualACME). The models are represented as XML files and the transformation engine is codified in Java. Corriveaunu *et al.* propose a prototype that takes a requirements model with variability (i.e., a domain requirements model) and enables the generation of a member's requirements model from it in such a way that the member's requirement model (necessarily without variability) can then be bound and tested against the actual behavior of this member's corresponding implementation. Dhungara *et al.* provides a prototype EuReCuS (End-user Requirements Elicitation and Customization of Services), which enables end-users to enter a user story using natural language text and presents relevant questions, as the user enters their story. EuReCuS is currently utilizing the product line variability modeling capabilities of the DOPLER tool suite. DREAM provides a tool that supports the management of the commonalities and variability of domain requirements and customizes the requirements of individual systems from these domain requirements. The Domain Use Case modeling relies on external third-party modeling tools, such as Rose XDE by IBM or Together Control Center by Borland. DREAM supports the export and import of domain Use Case models to/from XMI files. The use of this format allows the importation of entities from repositories and providing connectivity to other tools. The MORPHEUS tool gives support to the I-GANDALF approach. MORPHEUS allows the description of metamodels customized according to the project's semantic needs. Moreover, MORPHEUS is able to execute QVT transformations by using external engines. OVM-A presents a prototype as an extension to the DOORS tool. This prototype provides features such as determining overlaps and differences between the variability of two product lines, retrieval of all product lines offering a certain variant, retrieval of all variants common for all product lines or retrieval of all variants defined for a given variation point. SIRENspl has a tool called SirenSPLTool, which is a graphical editor based on Eclipse that supports traceability from Feature Models and Uses Cases and textual requirements. Finally SREPPLine has the REPPLineTool. This tool provides the management and the visualization of the artifacts variability and traceability links and the integration of the security standards, as well as the management of the security reference model proposed by SREPPLine.

We observe that many proposals have provided a prototyping tool with limited examples. This is a good first step, however in order to gain a broader acceptance the authors should test their tools in industrial environments with much more complex projects.

e. Validation

We consider five levels of validation for the analyzed studies. From lower to higher degree the type of validation is categorized as follows: simple example; academic case study; industrial-based case study; empirical controlled experiment with a control group; and industrial case study. Many papers provide only an example to illustrate the proposed method. The case study examines a phenomenon or unit, collects data, and analyzes the results in a single case. If the case study is applied in an academic context then we consider that it is an *academic case study* (e.g., a University). If the case study is applied in an industrial environment (e.g., automotive projects) then it is considered an *industrial-based case study*. In an empirically controlled experiment, the goal is to validate the hypothesis based on an analysis of the result differences between the control and the experimental group. Finally, the highest degree of validation is to apply an empirical experiment in an industrial environment.

TABLE V
VALIDATION

Approach	Validation
Alférez *et al.*	Academic case study
Bragança & Machado	Academic case study
Coelho & Batista	Academic case study
Corriveau *et al.*	Example
Dhungana *et al.*	Example
DREAM	Industrial-based case study
Guelfi & Perrouin	Academic case study
I-GANDALF	Academic case study
OVM-A	Industrial-based case study
SIRENspl	Industrial-based case study
SREPPLine	Industrial-based case study

Table V contains the results obtained. Two approaches only provide examples to illustrate their proposals Corriveanu *et al.* provides an example to generate Variability Contract. Dhungana *et al.* argues that model-driven approach to requirements verification in the presence of variability is entirely feasible. However, the proposals provide only a simple example.

Other approaches use case studies to illustrate their feasibility. Alférez *et al.* use a case study based on a Smart Home SPL to illustrate their proposal. In (Bragança, 2007) it is applied to a case study to derive the architectural requirements of a product line, based on a library product line. Coelho & Batista apply a case Study that models an E-Commerce system. This system represents business transactions via the internet. Guelfi & Perrouin illustrate the proposal with an Automatic Teller Machines (ATM) case study. I-GANDALF applies the proposal with a case study in the e-commerce system domain.

Finally, many proposals were applied in industrial environments. DREAM was applied to a case study of e-Travel Systems (e-TS). The case study was developed in cooperation with the Electronics and Telecommunications Research Institute, which is supported by the Korean national government, and Daewoo Information Systems Corporation, one of the top five IT and software companies in Korea. OVM-A documents experiments inside a project in the automobile industry. The goal of the project was the development of a sophisticated way to reuse requirements for electronic control units (ECUs) among different vehicle lines. After the application of the approach, the resulting variability model described around 40 variation points with approximately 150 variants for the climate control. SIRENspl was applied in TeleOperated Systems for ship hull maintenance (TOS). TOSs are robotic systems used to perform ship maintenance tasks, such as cleaning or painting a ship's hull. The research was done by applying a qualitative research method (Action Research) (Barkerville, 2001). SREPPLine provides a case study to the Public Registry Online Product Line of a Spanish Public Administration.

In conclusion, most of the approaches use the Case Study as "proof of concept" instead of using it as an evaluation method. Furthermore, the approaches should improve their validations with rigorous design experimentations (e.g., randomization, replication of the studies etc).

4 CONCLUSION

In this study, we analyzed eleven Requirement Engineering approaches that use MDD techniques for SPL development. We used five comparison criteria: SPL activity support, RE covered tasks, MDD strategy support, the degree of automatic support with a given tool, and the type of validation of the proposals. This comparison provides information about the advantages and disadvantages of the approaches included in the study. The results obtained from this comparison have allowed us to identify commonalties among the different approaches as well as several research gaps. Firstly, most of the research is focused on the Scoping and Domain Engineering activities, but the Application Engineering is the less supported. The approaches should provide better mechanisms to incorporate new deltas to the SPL (SPL evolution) instead of just producing derivations based on the Domain Requirements specifications and extending the product requirements specification at the Application Domain. Regarding the development strategy, the Proactive strategy adoption was the most common. However, this strategy is the most expensive and it assumes a higher level of risk over other strategies. It would be interesting to combine a Proactive strategy with the Reactive or the Extractive strategies to avoid these disadvantages. Another observation was that most of the approaches give support to the Commonality & Variability modeling, however, in many cases this variability is modeled with the functional requirements (e.g., Uses Cases extended with Variation Points). This can produce model overloading and maintainability problems, which can be avoided with a clear separation between the variability and the functional requirements. This is the case of OVM-A, where the variability is stored in a separate model. Another interesting solution is proposed by Alférez *et al.*, where the requirement models and variability are modeled in different models but are composed with the VML4RE language. This solution has the advantage of allowing the relation of multiple requirement models with the Feature Model. A further problem is the lack of integration with the scoping activity; this problem was reported in other reviews such as (John, 2009). The integration between both of these activities could be improved to set the SPL boundaries aligned with the requirements established in the Domain Engineering.

Regarding the RE tasks, most of the approaches usually cover the elicitation and modeling of requirements. Regarding the elicitation, Dhungara *et al.* offers a tool where partial elicitation and requirement analysis can be performed, with the simulation in components based on end-user decisions. Guelfi & Perrouin also cover the requirements analysis with a technique to derive requirements and check OCL constraints. The requirements validation is mentioned in OVM-A, SIRENspl and SREPPLine, however only OVM-A proposes a technique called ScenTED (Scenario based TEst case Derivation): a model-based, reuse-oriented technique for test case derivation. Regarding the requirements verification, we found two proposals: Corriveau *et al.* and OVM-A. Corriveau *et al.* offers an interesting technique based on a MDD strategy to generate test cases. OVM-A proposes a requirements verification based on model-checking techniques. Including these activities, requirement validation and verification, in the SPL development could allow us to check whether or not the used artifacts satisfy the stakeholder needs. Regarding the employed artifacts, we found some approaches that adapted traditional notations such Use Cases in the SPL requirements. For example, DREAM extends traditional Use Cases to model functionality and requirements variability. However, the use of both functional and requirements aspects in the same model can produce legibility problems. Another notation used is the Goal Model notation (i.e., the Intentional Modeling of I-GANDALF). This notation has the advantages of covering both functional and non-functional requirements and its variability. Dealing with non-functional requirements in the Domain Engineering could help to improve the quality of the products obtained from the product line. The system variability was usually modeled with Feature Models using the FODA-notation or extended UML class models to represent variation points. A widely used notation is the Orthogonal Variability Model, which represents the variability in a separate model and has traceability relationships with other models. This notation has the advantage of explicitly setting up the traceability relations to architectural and application artifacts. We only found three approaches that supported both vertical and horizontal traceability. A well-defined traceability strategy could improve the quality of the software products in the product family.

Regarding the Model-driven coverage, we found two common ways to use model-driven transformations in the SPL development. Many authors provide refinements from requirements to model artifacts; others provide transformations from domain requirements to application requirements to perform an automatic derivation of the requirements of a product. Another interesting application of the MDD strategy is the refactoring of Uses Cases and Feature Models into textual requirements in SIRENspl, or to obtain testable requirement models as has been presented in the approach by Corriveau *et al.* We found several academic prototypes in the proposals; however there is a clear need to test these tools in industrial environments. With respect to the validation, most of the approaches only provide simple examples to illustrate their proposals. There is common use of the Case Study as "proof of concept" instead of using it as a well-defined validation method. The use of empirical studies with practitioners and experienced subjects could help to improve their methods and tools.

All these issues provide a clear motivation for further research on RE for SPL development following MDD approaches. Our future work will include the tailoring of a specific RE approach for dealing with scoping in conjunction with the domain requirements engineering, and the study of approaches to deal with the evolution of SPL when new products and requirements arise.

ACKNOWLEDGMENTS

This research is part of the MULTIPLE project (with ref. TIN2009-13838).

REFERENCES

Alférez, M., Kulesza, U., Weston, N., Araujo, J., Amaral, V., Moreira, A., Rashid, A., Jaeger, M. C., A (2008). Metamodel for Aspectual Requirements Modeling and Composition. Technical report. Universidade Nova de Lisboa, Portugal.

Alves, V., Niu, N., Alves, C., Valença, G. (2010). Requirements Engineering for Software Product Lines: A Systematic Literature Review. In: *Information and Software Technology* 52 (8), (pp. 806 – 820), Elsevier.

AMPLE Project Research Group, Available: http://ample.di.fct.unl.pt

Amyot D., Mussbacher G. (2011). User Requirements Notation: The First Ten Years, The Next Ten Years. In *Special Issue: Selected Papers of the IEEE International Conference on Computer and Information Technology*, 6(5), (pp.747-768), Academy Publisher.

Barkerville, R. (2001). Conducting Action Research: High Risk and High Reward in Theory and Practice. *Qualitative Research in Information Systems*, (pp. 192-218). Idea Group Publishing.

Batista, T., Bastarrica, M.C., Soares, S., Fernandes, L. (2008). A Marriage of MDD and Early Aspects in Software Product Line Development. In: *Early Aspects Workshop at 12th International Software Product Line Conference (SPLC)*, (pp. 97–104). Limerick, Ireland. IEEE.

Bragança, A. (2007). Methodological Approaches and Techniques for Model Driven Development of Software Product Lines. PhD thesis. Universidade do Minho, Portugal.

Bragança, A., Machado, R. J. (2007). Automating Mappings between Use Case Diagrams and Feature Models for Software Product Lines. *Proceeding of the 11th International Software Product Line Conference (SPLC)*, (pp. 3-12). Kyoto, Japan. IEEE.

Cheng, B. H. C., Atle, J. M. (2007). Research Directions in Requirements Engineering. *Proceeding of Workshop on the Future of Software Engineering held on International Conference on Software Engineering*, (pp. 285-303). Minneapolis, MN, USA. IEEE.

Coelho K., Batista T. (2011). From Requirements to Architecture for Software Product Lines. *In Proceedings of the 9th IEEE/IFIP Working Conference on Software Architecture (WICSA)*, (pp. 282 - 289). Boulder, Colorado, USA. IEEE Computer Society.

Corriveau, J.P., Bashardoust S., Radonjic, V.D. (2011). Requirements Verification in the Presence of Variability. *Proceedings on Model-Driven Requirements Engineering Workshop (MoDRE)*, (pp.74-78). Trento, Italy. IEEE.

Dhungana, D., Seyff, N., Graf, F. (2011). Research Preview: Supporting End-User Requirements Elicitation Using Product Line Variability Models. *Proceeding on the 17th International Working Conference of Requirements Engineering: Foundation for Software Quality*, (pp. 66-71), Essen, Germany Springer.

Gomaa H. (2004). Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Addison-Wesley.

International Telecommunication Union *(2008)*, User Requirements Notation (URN) – Language definition. *Recommendation Z.151 (11/08)*, Available: http://www.itu.int/rec/T-REC-Z.151/en.

González-Baixauli, B., Navarro, E., Laguna, M.A., Sampaio do Prado Leite, J.C. (2009). Goals to Features: a Model Driven Proposal in I-GANDALF. GIRO Technical Report 2009. University of Valladolid, Spain.

Guelfi N., Perrouin G. (2007). A Flexible Requirements Analysis Approach for Software Product Lines. Proceedings on 13th International Working Conference REFSQ, (pp. 78-92). Trondheim, Norway. Spinger

John I., Eisenbarth, M. (2009). A Decade of Scoping – A Survey. *Proceeding of the 13th International Software Product Line Conference*, (pp. 31-40). San Francisco, California, USA.ACM.

Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, S. (1990). Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21. Software Engineering Institute, Carnegie Mellon University, USA.

Kovačević, J., Aférez, M., Kulesza, U., Moreira, A., Araújo, J., Amaral, V. (2007). Survey of the state-of-the-art in Requirements Engineering for Software Product Line and Model-Driven Requirements Engineering. AMPLE Deliverable D1.1. Available: http://ample.holos.pt/gest_cnt_upload/editor/File/public/Deliverable%20D1.1.pdf

Krueger, C. W. (2001). Easing the Transition to Software Mass Customization. *Proceedings of the 4th International Workshop on Software Product-Family Engineering* (PFE 2001), (pp. 282-293). Bilbao, Spain. Springer.

Lauenroth K., Pohl K. (2008). Dynamic Consistency Checking of Domain Requirements in Product Line Engineering. *Proceedings of the IEEE International Requirements Engineering Conference*, (pp. 193–202). Barcelona, Spain. IEEE.

Pohl, K., Böckle, G., Van Der Linden, F. (2005). Software Product Line Engineering: Foundations, Principles, and Techniques. Springer

Reusable Asset Specification (2012) .Object Management Group. Available: http://www.omg.org/technology/documents/formal/ras.htm

Reuys, A., Kamsties, E., Pohl, K., Reis, S. (2005). Model-Based System Testing of Software Product Families. *Proceeding of the Advanced Information Systems Engineering (CAiSE)*, (pp. 519-534). Porto, Portugal. Springer.

Mellado, D.. Fernández-Medina, E., Piattini, M. (2007). SREPPLine: Towards a Security Requirements Engineering Process for Software Product Lines. In: *Proceedings of the 5th International Workshop on Security in Information Systems (WOSIS)*, (pp. 220-232). Funchal, Madeira, Portugal. INSTICC Press.

Mens T., Van Gorp P. (2006). A Taxonomy of Model Transformation. *Proceedings of the International Workshop on Graph and Model Transformation (GraMoT5)*, (pp. 125–142). Tallinn, Estonia. Mendeley.

Moon, M., Yeom, K., Chae, H. S. (2005). An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability Analysis in a Product Line. In: *IEEE Transactions on Software Engineering*, 31 (7), (pp. 551-569), Piscataway, NJ, USA. IEEE.

Moon M., Chae H. S., Yeom K. (2006). A Metamodel Approach to Architecture Variability in a Product Line. *Proceeding of the 9th International Conference on Software Reuse (ICSR)*, (pp. 115-126). Turin, Italy. Springer.

Nicolás Ros, J. (2009). Una Propuesta de Gestión Integrada de Modelos y Requisitos en Líneas de Productos Software. PhD thesis. Departamento de Informática y Sistemas, Universidad de Murcia.

Nicolás, J., Toval, A. (2009). On the generation of requirements specifications from software engineering models: A systematic literature review. In: *Information and Software Technology* 51 (9), (pp. 1291- 1307), Butterworth-Heinemann Newton, USA. Elsevier.

Toval Álvarez J. A., Nicolás J., Moros B., García F., (2002). Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach. *Requirements Engineering* 6 (4), (pp. 205-219). Springer.

Object Management Group, OMG, Available: http://www.omg.org/

Telelogic AB (2004). DXL Reference Manual, DOORS 7.1