Metamodeling Variability to Enable Requirements Reuse

Begoña Moros[1], Cristina Vicente-Chicote[2], Ambrosio Toval[1]

[1]Departamento de Informática y Sistemas
Universidad de Murcia, 30100 Espinardo (Murcia), Spain
{bmoros, atoval}@um.es

[2] Departamento de Tecnologías de la Información y las Comunicaciones
Universidad Politécnica de Cartagena, 30202 Cartagena (Murcia), Spain
Cristina.Vicente@upct.es

Abstract. Model-Driven Software Development (MDSD) is recognized as a very promising approach to deal with software complexity. The Requirements Engineering community should be aware and take part of the Model-Driven revolution, enabling and promoting the integration of requirements into the MDSD life-cycle. As a first step to reach that goal, this paper proposes REMM, a Requirements Engineering MetaModel, which provides variability modeling mechanisms to enable requirements reuse. In addition, this paper also presents the REMM-Studio graphical requirements modeling tool, aimed at easing the definition of complex requirements models. This tool enables the specification of (1) catalogs of reusable requirements models (modeling *for reuse* facet of the tool), and (2) specific product requirements by reusing previously defined requirements models (modeling *by reuse* facet of the tool).

Keywords: Model-Driven Software Development (MDSD), Requirements Engineering (RE), Requirements MetaModel (REMM), Requirements Reuse, Requirements Variability.

1 Introduction

Model-Driven Software Development (MDSD) aims at raising the level of abstraction at which software is conceived, implemented, and evolved, in order to help managing the inherent complexity of modern software-based systems. As recently stated in a Forrester Research Inc. study "*model-driven development will play a key role in the future of software development; it is a promising technique for helping application development managers address growing business complexity and demand*" [1].

In this context, requirements should be considered first class entities and they should be modeled in order to be integrated as part of the MDSD life cycle [2]. Modeling requirements involves selecting and adopting a requirements metamodel. This metamodel must include all the concepts (and the relationships existing between them) commonly appearing in a requirements specification process [3]. For instance, the OMG standard SysML (*Systems Modeling Language*) [4] metamodel includes a reduced set of concepts devoted to requirements modeling. In this line, some efforts have been made to integrate textual requirements into model-driven approaches [5, 6].

On the other hand, the benefits of reuse to improve the software development process productivity and the final product quality are very well known. Reuse can be introduced at different stages during the software development process, from requirements to design and implementation. However, the higher the level of abstraction at which reuse takes place, the larger its benefits [7, 8]. Requirements reuse can, therefore, provide significant gains in developmental productivity and in the quality of the resulting software products [8]. Moreover, in a recent study regarding current and future Requirements Engineering (RE) research trends [9], requirements reuse has been pointed out as one of the most pressing needs and grand challenges in RE research, which solutions are likely to have the greatest impact on Software Engineering research and practice.

However, requirements reuse is often carried out in a non-systematic way [7, 10], since requirements dependencies are not explicitly modeled in most cases. As a consequence, when a requirement is reused, those requirements (or, more generally speaking, those elements, e.g. stakeholders, test cases, documental sources, etc.) related to it, might not be properly taken into account at reuse time [10]. This problem could be easily overcome following a MDSD approach, whenever the adopted requirements metamodel:

- Allows designers to explicitly model inter-requirements relationships together with the traces existing between requirements and the rest of the elements involved in the RE process.
- Provides one or more explicit reuse mechanisms, allowing requirements engineers (1) to define assets of reusable requirements models (modeling *for reuse*), and (2) to build a certain product requirements specification by reusing some of the requirements previously defined (modeling *by reuse*). In the last case, when a requirement is reused in a product specification, all the elements related to it (those it depends on) can be easily incorporated to the specification. Besides, the definition of inter-requirements traces (in particular, incompatibility dependencies) makes it possible to identify and prevent many problems at reuse time.

Variability modeling is known to be essential for analyzing reuse strategies [11]. As a consequence, if a requirements metamodel needs to incorporate some kind of reuse mechanism it must deal with variability in some way. The most widely accepted approach for variability modeling in the RE field is the one based on feature modeling [12]. However, the variability information captured in feature models is not sufficient to represent, for instance, the variability related to requirements dependencies [13]. Thus, some authors prefer to model variability directly within requirements artifacts [14-16]. This is the approach we have taken in our proposal, that is, to integrate requirements variability directly into the proposed requirements metamodel. This metamodel, called REMM (*Requirements Engineering MetaModel*), is aimed at modeling both catalogs of requirements *for reuse* and product requirements *by reuse*. This metamodel, which will be later detailed in section 2, is a first step to integrate reusable requirements in a MDSD approach. A graphical modeling tool, called REMM-Studio, has been implemented on top of REMM to support requirements modeling, reuse, and validation.

Before entering into details, the following sections outline the main contributions of our proposal and some of our research background related to this work. Then, the rest of the paper is organized as follows. Firstly, the proposed Requirements Engineering MetaModel (REMM), defined to support requirements reuse, is presented in section 2. Section 3 presents the REMM-Studio graphical modeling tool, implemented to support the two facets of REMM, that is, defining catalogs of reusable requirements models (modeling *for reuse* facet), and defining specific product requirements by reusing predefined models (modeling *by reuse* facet). Then, section 4 presents some related works regarding requirements reuse and variability modeling in textual requirements. Finally, section 5 outlines some conclusions and future research lines.

1.1. Contributions of the Proposal

In order to make clear our contribution, we enumerate next the main achievements of our proposal, pointing out the differences with other approaches that will be later commented in the related work section.

- The main goal of our research is to specify, validate and trace requirements models to other software models using a Model-Driven approach. We consider this a major contribution since most proposals linking models to requirements are rather Model-Based (e.g. based on use cases [17], based on i* models [18], etc.) than Model-Driven, as it follows from the very reduced number of publications related to MDSD approaches appearing in the most relevant RE journals and conferences.
- Our MDSD proposal revolves around a Requirements Engineering MetaModel (REMM) which, unlike other metamodels [5, 6, 19, 20], encompasses a comprehensive set of RE concepts (e.g. requirements, stakeholders, test cases, etc.), and relationships between them (e.g. extra- and inter-requirements traces), as detailed in section 2.
- REMM has been recently extended with two different variability mechanisms to enable requirements reuse. Firstly, REMM allows requirements engineers to define optional requirements by means of OR parent-child traces, similarly to OR decompositions of goals [21, 22]. Besides, REMM supports the definition of parameterized requirements, enabling the inclusion of variability into textual requirements specifications. Commonly, this kind of variability is modeled at requirements level using separate feature models [23]. The main contributions in this line, is that both variability mechanisms are provided by a unique metamodel (REMM), in the line of [24]. This makes it possible: (1) to keep all the information about the requirements and their variability points connected, and (2) to make explicit the relationships between the requirements parameters (variability points) by means of the explicit relationships existing between requirements. Thus, this approach avoids the problems derived from having to keep consistent two separated requirements and variability models.
- Finally, REMM is supported by a fully working tool, called REMM-Studio. This is also a major contribution since most proposals remain at a theoretical level. Furthermore, regarding current requirements management tools, they do not

support systematic requirements reuse [25, 26], and most of them neither requirements variability specification [27, 28].

Once the main contributions of our proposal have been highlighted, the following section briefly presents some of our research background in this line.

1.2. Research Background

To take advantage of the benefits of reuse at requirements level, our research group proposed a reuse-based RE method called SIREN [29]. This method could be considered both a document-based and a repository-based approach since it revolves around a repository of reusable requirements catalogs. Each of these catalogs includes a set of requirements specification documents, formatted according to the IEEE standard templates. These catalogs contain a set of related requirements belonging to the same *profile* –e.g. security [29], personal data protection [30], etc.– or to the same *domain* –e.g. tele-operated systems [31]. Note that the definition of a domain catalog is very close to the requirements specification of a software product line.

Requirements engineers may use the repository: (1) to improve the quality of its catalogs by adding new requirements or improving the existing ones or (2) to reuse the existing requirements in their current projects. The experience gained in the realm of requirements reuse led us to define the key issues that, in our opinion, should be taken into account for a reuse-based RE process to be successful [32].

On the other hand, due to the increasingly growing interest of the Software Engineering Community in MDD technologies, and given the major relevance of the RE process in software development, we are convinced of the great synergy that could arise from integrating requirements into a MDD approach, which would provide mutual benefits to the Software Engineering and to the Requirements Engineering Communities [33]. In this context, a preliminary version of the Requirements Engineering Meta-Model (REMM) and the supporting tool (REMM-Studio) were already proposed in [34]. In this paper, we present an improved version of both REMM and REMM-Studio, which now have been extended to support requirements variability modeling, enabling requirements reuse. Therefore, the document-based repository proposed in SIREN has evolved to a model-based repository, allowing designers to partially or completely reusing requirements models. For the sake of clarity and readability we have omitted some elements of the previous version of REMM to focus on the new concepts regarding requirements reuse.

2 Extending REMM to Enable Variability Modeling

As previously stated, the Requirements Engineering MetaModel (REMM) presented in this paper is an improved version of the one already introduced in [34]. REMM is extended here to enable requirements reuse, covering the following new aspects:

- Both the modeling *for reuse* and the modeling *by reuse* facets must be provided by REMM. This means that REMM must enable the construction of both reusable requirements models and specific product requirements models.

- In addition, reusable requirements shall provide some kind of variability modeling mechanism. To achieve this, we have provided REMM with the possibility of defining parameterized requirements when modeling *for reuse*. Requirements parameters shall be instantiated at reuse time when the parameterized requirement is selected for its inclusion in a new product specification.

The central concept included in REMM to support requirements reuse is the `Repository`. As it can be observed in Fig. 1, the repository contains `ReusableCatalog`s, which include a set of related `ReusableRequirement`s belonging to the same `CatalogType`, i.e. a `PROFILE` or a `DOMAIN`, according to the SIREN RE process (see section 1.2). These elements correspond to the *modeling for reuse* facet of the metamodel.

On the other hand, the key element supporting the *modeling by reuse* facet of REMM is the `ProductCatalog`. Product catalogs contain all the `ProductRequirement`s related to the specification of a new product. A product requirement could be product specific or it can be reused from some of the `ReusableCatalog`s available in the `Repository`. In this case, the product requirement keeps a trace to its source `ReusableRequirement` (*`reusedFrom`* association in the metamodel). This way, the original requirement specification can be consulted at any time.

In order to model requirements variability, the specification of parameterized requirements is allowed in `ReusableCatalog`s. A parameterized requirement is a `ReusableRequirement` which contains one or more `Parameter`s, characterized by a *`name`* and a *`type`*. The *`type`* of the parameter could be a number (`NumberType`), a string (`StringType`) or a value from an enumerated set of values (`EnumType`). This allows designers to specify variation points where variants can range from an infinite set of values (number or string types) to a finite one (enumerations). In case the parameter type is an `EnumType`, the *`minimum`* and *`maximum`* number of variants has to be established. By default, these attributes are initialized to 1 and thus, just one of the `EnumLiterals` (from the `EnumType)` has to be selected at instantiation time. The literals belonging to an `EnumType` could be `OPTIONAL` or `MANDATORY` (*`mandatoryLevel`*) meaning that the variant has to be included as part of any of the products defined from the reusable catalog or not.

Any reusable requirement stored in a repository could be selected at reuse time. Nevertheless, we should pay special attention to those requirements defined with a `MANDATORY` *`priority`*, which in the case of domain catalogs can be considered to be the common requirements to be included in the specifications of all the products belonging to the corresponding application domain.

Both reusable and product catalogs include not only requirements, but also the `Trace`s (relationships) existing between them. In this case, only two kind of traces have been depicted in REMM: `DependenceTrace` (which *`type`* can be `REQUIRES`, `EXCLUDES`, `DEPENDS`), and `ParentChildTrace` (which *`type`* can be `AND`, `OR`). Readers can find a deeper explanation about these and other types of traces considered in REMM in [34]. Here, we will just focus on the three modeling improvements included in the new version of REMM regarding traces, which can be summarized as follows:

- Requirements traces have been reorganized in a hierarchical way, i.e. all their common aspects have been factorized in the abstract `Trace` class.

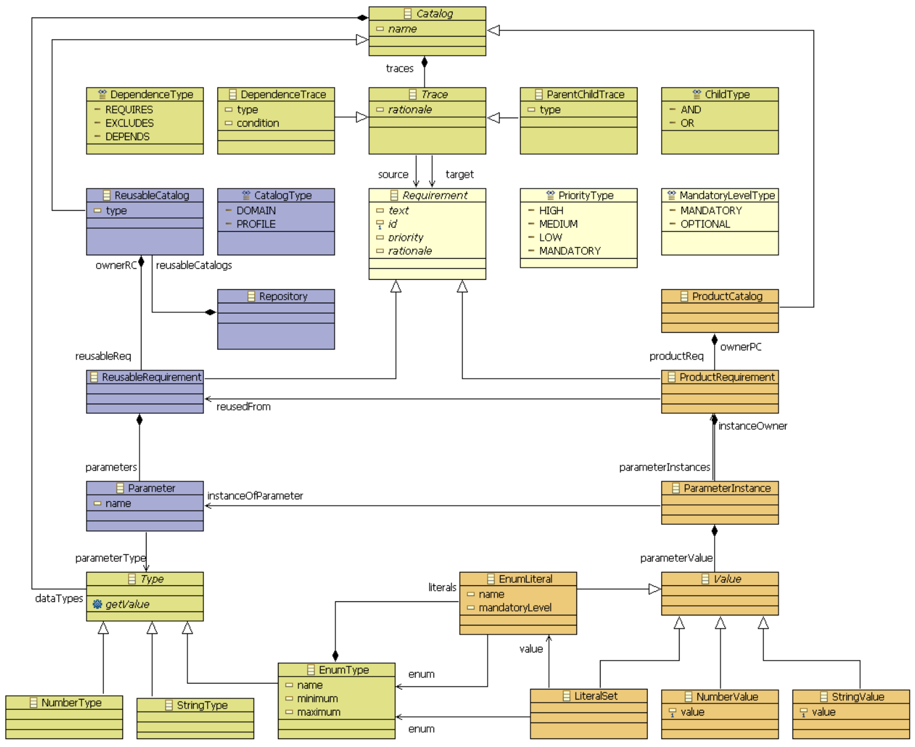


Fig. 1. REMM (*Requirements Engineering MetaModel*).

- Traces now include a `rationale` attribute aimed at specifying the reason for including them into the catalog. This information could be very useful at reuse time to decide whether a trace should be reused or not.
- `DependenceTrace`s now include a new attribute called `condition` that enables the specification of the condition under which the trace will exist. This condition is especially relevant in the case of dependence traces between parameterized requirements, since it helps establishing the relationship between the values of the parameters.

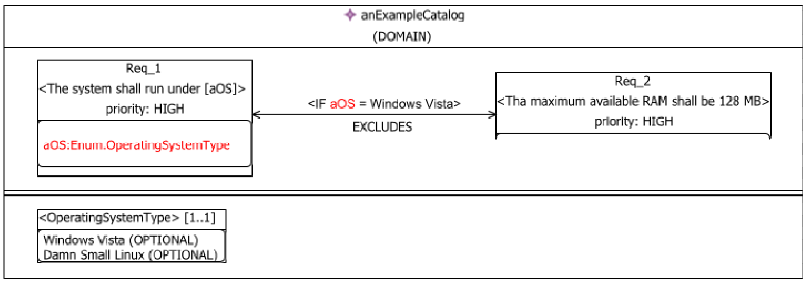

anExampleCatalog
(DOMAIN)
Req_1
<The system shall run under [aOS]>
priority: HIGH
aOS:Enum.OperatingSystemType
<IF aOS = Windows Vista>
EXCLUDES
Req_2
<Tha maximum available RAM shall be 128 MB>
priority: HIGH
<OperatingSystemType> [1..1]
Windows Vista (OPTIONAL)
Damn Small Linux (OPTIONAL)

Fig. 2. Conditional statement included in a `EXCLUDES` dependence trace.

For instance, as shown in Fig. 2, the condition included in the EXCLUDES dependence trace between Req_1 and Req_2, establishes that if the system runs Windows Vista, then the maximum available RAM can not be restricted to 128 MB.

The example shown in Fig. 3, presents a conditional dependence trace, which involves the parameters defined both in the source and in the target requirements. In this case, the condition states that if the selected operating system is Windows Vista, then the system must have at least 500 MB of RAM available.

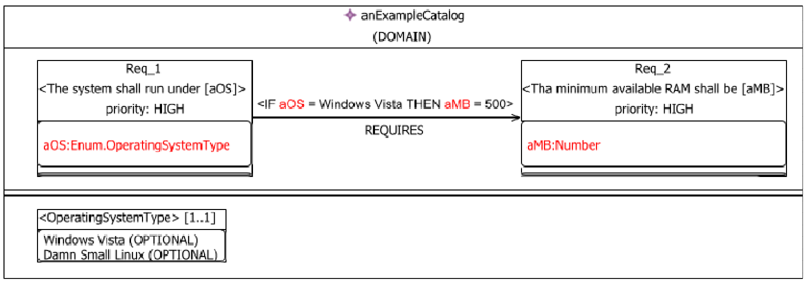

anExampleCatalog
(DOMAIN)
Req_1
<The system shall run under [aOS]>
priority: HIGH
aOS:Enum.OperatingSystemType
<IF aOS = Windows Vista THEN aMB = 500>
REQUIRES
Req_2
<Tha minimum available RAM shall be [aMB]>
priority: HIGH
aMB:Number
<OperatingSystemType> [1..1]
Windows Vista (OPTIONAL)
Damn Small Linux (OPTIONAL)

Fig. 3. Conditional statement included in a REQUIRES dependence trace.

3 Requirements Reuse in REMM-Studio

The first version of the REMM-Studio tool, already presented in [34], was aimed at helping requirements engineers: (a) to build graphical requirements models, (b) to validate them against REMM and against a set of additional OCL constraints, and (c) to automatically generate navigable Software Requirements Specification (SRS) documents from their models using the IEEE 830 template. For the sake of simplicity, this paper will only address the improvements included in REMM-Studio to support the new extended modeling capabilities of REMM regarding requirements reuse.

In this vein, the improved REMM-Studio tool presented here, provides two new model editors, each one supporting one of the two facets or views included in REMM, that is, (1) the definition of catalogs of reusable requirements models (modeling *for reuse* facet), and (2) the specification of new product requirements by reusing previously defined models (modeling *by reuse* facet). In order to illustrate the modeling capabilities of these two model editors, a case study inspired in the requirements specification for a *HOme LIghting automation System* (HOLIS2000) presented in [35], will be used.

3.1 Tool Supporting the Modeling *for Reuse* Facet of REMM

As already explained in section 2, in REMM, the Repository is defined to contain ReusableRequirements models organized in Catalogs, which store requirements belonging to the same DOMAIN or PROFILE. Please note that REMM DOMAIN catalogs can now provide full software product line requirements specifications, since parameterized requirements are now available in the new version of the metamodel. In this vein, the example repository shown in Fig. 4, presents an example `Catalog` of reusable requirements models (some of them parameterized), all belonging to the *HOme LIghting automation Systems* [35] `DOMAIN` (*HOLIS Catalog*). The repository depicted in Fig. 4, also contains a Normative `PROFILE Catalog`, which gathers requirements regarding copyrighting, licensing, legal policies, etc. (*Normative Catalog*).

Variation points can be introduced in reusable requirements models by including any number of `Parameters` in their specification (*`text`* attribute). Parameters must be written into square brackets to help the tool automatically detect them and create the corresponding parameter declarations, which will include the name of the parameter (as declared in the requirements description) and also its type. For instance, the [aNumber] parameter included in the description of Req_2 in the HOLIS Domain Catalog, is declared to be of type Number, while the [formatTime] parameter included in the description of Req_7 is declared to be of an enumerated type Enum.FormatTimeType, which can take values {12h, 24 h} (see Fig. 4).

It is worth noting that for all parameters of an `EnumType` type, the minimum and maximum number of `EnumLiterals` must be established. For instance, as shown in Fig. 4, the [onLevel] parameter in Req_9 can take only one value from the set {20%, 30%, 40%}, while the [formatTime] parameter in Req_7 can take one or the two values included in the set {12h, 24h}.

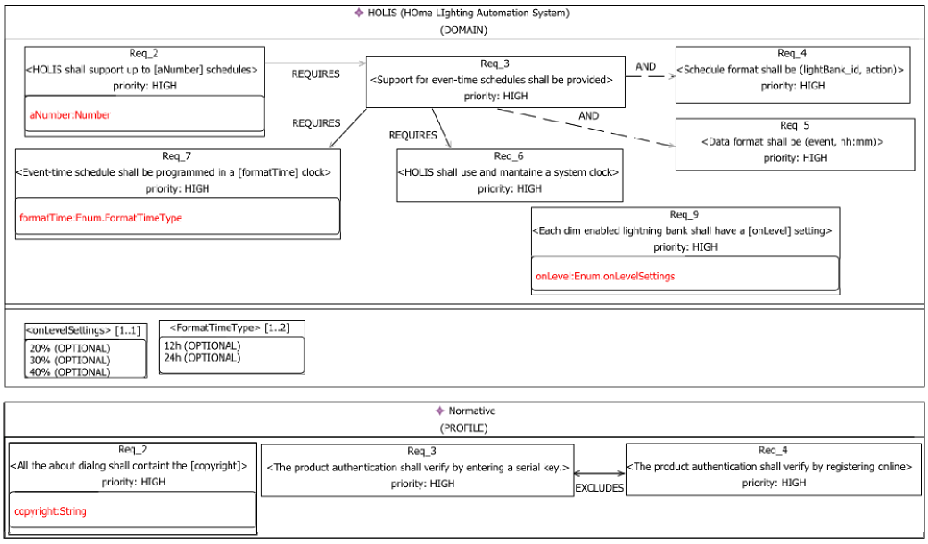

HOLIS (HOme LIghting Automation System)
(DOMAIN)
Req_2
<HOLIS shall support up to [aNumber] schedules>
priority: HIGH
aNumber:Number
REQUIRES
Req_3
<Support for even-time schedules shall be provided>
priority: HIGH
AND
Req_4
<Schedule format shall be (lightBank_id, action)>
priority: HIGH
REQUIRES
AND
REQUIRES
Req_5
<Data format shall be (event, hh:mm)>
priority: HIGH
Req_7
<Event-time schedule shall be programmed in a [formatTime] clock>
priority: HIGH
formatTime:Enum.FormatTimeType
Rec_6
<HOLIS shall use and mantaine a system clock>
priority: HIGH
Req_9
<Each dim enabled lightning bank shall have a [onLevel] setting>
priority: HIGH
onLevel:Enum.onLevelSettings
<onLevelSettings> [1..1]
20% (OPTIONAL)
30% (OPTIONAL)
40% (OPTIONAL)
<FormatTimeType> [1..2]
12h (OPTIONAL)
24h (OPTIONAL)
Normative
(PROFILE)
Req_2
<All the about dialog shall containt the [copyright]>
priority: HIGH
copyright:String
Req_3
<The product authentication shall verify by entering a serial key.>
priority: HIGH
EXCLUDES
Rec_4
<The product authentication shall verify by registering online>
priority: HIGH

Fig. 4. REMM-Studio *for reuse* facet. Repository including an excerpt of the reusable requirements models included in the HOLIS and the Normative Catalogs.

When defining reusable requirements models, inter-requirements `Traces` must be defined. Examples of `Dependence` (`REQUIRES` and `EXCLUDES`) and `ParentChild` (`AND` type) traces are shown in Fig. 4. The meaning of these traces is widely described in [34], and their importance at reuse time will be further discussed in the following section.

On saving the repository, requirements models are validated against REMM and also against some extra constraints, some of them implemented as OCL rules, and others as java boolean methods. For instance, the validation process assures that: all requirements have unique identifiers; all parameters included in a reusable requirement have a valid type and a unique name; all parameters included in a requirement description have been specified in the parameter declaration section, and *vice versa*, etc.

3.2 Tool Supporting the Modeling *by Reuse* Facet of REMM

When developing a new product, it seems quite a good idea to take a look at previously developed solutions to find out if they provide any useful resource which can be reused. This particularly applies when specifying new products requirements since, as previously stated, the sooner reuse is applied during the development process, the greater the benefits. Thus, in order to specify the requirements of a new product, which will be stored in a `ProductCatalog`, the first step should be to find out if our repository provides some suitable reusable requirements. To achieve this, the tool supporting the modeling *by reuse* facet of REMM, allows requirements engineers: (1) to load the repository and thus, to import any of the `ReusableRequirements` stored in its catalogs, and (2) to depict new project specific `ProductRequirements` from scratch.

Users can load the requirements models stored in the repository using the contextual popup menu associated to the canvas of the modeling *by reuse* tool, which represents the current product catalog (see Fig. 5).

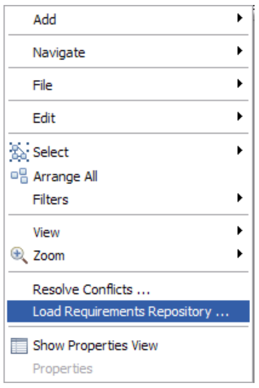


Fig. 5. Contextual popup menu for the current product catalog.

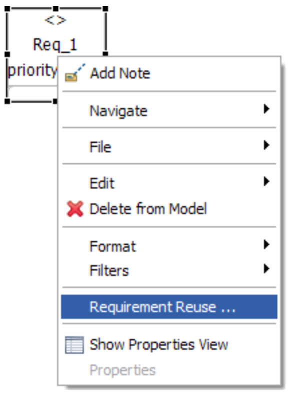
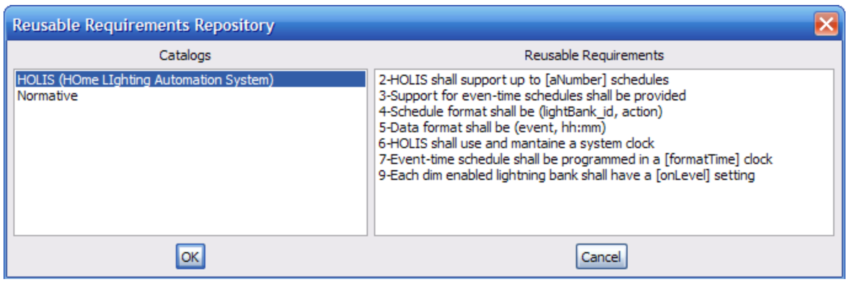


Fig. 6. Contextual popup menu for product requirements.

New product requirements can be created (1) from a reusable requirement specification imported from the already loaded repository, or (2) from scratch. In the first case, the requirements engineer must select the reusable requirement that will be used as the basis for the new product requirement specification (see Fig. 6). Then, a dialog showing the content of the repository is opened (see Fig. 7), enabling the selection of the reusable requirements contained in each catalog.



Reusable Requirements Repository

Catalogs	Reusable Requirements
HOLIS (HOme LIghting Automation System)	2-HOLIS shall support up to [aNumber] schedules
Normative	3-Support for even-time schedules shall be provided
	4-Schedule format shall be (lightBank_id, action)
	5-Data format shall be (event, hh:mm)
	6-HOLIS shall use and mantaine a system clock
	7-Event-time schedule shall be programmed in a [formatTime] clock
	9-Each dim enabled lightning bank shall have a [onLevel] setting

Fig. 7. Reusable requirements repository dialog at reuse time.

When a reusable requirement is selected, all those related to it by means of a `Dependence.REQUIRES` or a `ParentChild.AND` trace, are also automatically included in the current product catalog. Thus, inter-requirements relationships are explicitly taken into account at reuse time. For instance, if Req_2 (with text *"HOLIS shall support up to [aNumber] of schedules"*) is selected from the excerpt of the HOLIS Catalog shown in Fig. 4, then all the other reusable requirements included in this excerpt, except Req_9, will be also reused and included in the resulting product catalog (see the result of this operation in Fig. 8). Please note that the new product requirements built by reusing those defined in the HOLIS catalog keep a reference to the original ones, but now have a different ID. The values of the rest of the attributes are copied from the reusable requirement.

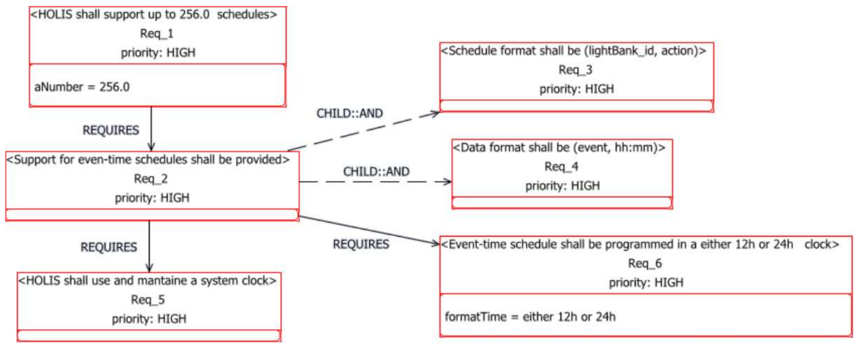

<HOLIS shall support up to 256.0 schedules>
Req_1
priority: HIGH
aNumber = 256.0
REQUIRES
CHILD::AND
<Schedule format shall be (lightBank_id, action)>
Req_3
priority: HIGH
<Support for even-time schedules shall be provided>
Req_2
priority: HIGH
CHILD::AND
<Data format shall be (event, hh:mm)>
Req_4
priority: HIGH
REQUIRES
REQUIRES
<Event-time schedule shall be programmed in a either 12h or 24h clock>
Req_6
priority: HIGH
formatTime = either 12h or 24h
<HOLIS shall use and mantaine a system clock>
Req_5
priority: HIGH

Fig. 8. Requirements model reused when selecting Req_1.

Parameterized reusable requirements must be instantiated when imported into a product catalog. As shown in Fig. 8, the parameter [aNumber] in Req_1, corresponding to the number of supported schedules, has been fixed to 256, and the parameter [formatTime] in Req_6, corresponding to the clock format, takes the two values in the set {12h, 24h}.

This tool also provides some model validation facilities on saving the product catalog models. As before, the final product catalog is validated against REMM and against some extra constraints. For instance, the validation process assures that: all parameterized requirements have been instantiated; product catalogs do not contain `Dependence.EXCLUDES` traces, neither product requirements created by reuse from others related in the original reusable catalog by this kind of traces, since this means they are incompatible for the same product. If incompatible reusable requirements are detected, more detailed information about the problem can be obtained selecting the *Resolve Conflicts* option available in the product catalog contextual menu (see Fig. 5).

4 Related Work

This section describes some related research regarding variability modeling at requirements level and the supporting tools. As stated in the introduction, all reuse strategies should take variability into account. At requirements level, variability is commonly tackled by means of one of the following approaches: (1) using feature diagrams [23], (2) using goal-oriented models [21, 22], or (3) directly including the variability into the textual requirements specifications [16, 36-38].

Regarding the inclusion of variability into textual requirements, the simplest way to achieve it is to add certain keywords or phrases (e.g. *either…or*) into the requirement statement. However, since this strategy does not prevent from ambiguous requirements specifications [38], explicit variability modeling mechanisms have to be provided. In this vein, the general approach consists of augmenting the textual requirement specifications [16], either with textual constructs (e.g. framed with "<<" ">>" [36], or XML tags [37, 38]), or simply extending the adopted requirements specification template (e.g. the IEEE 830). These approaches have two main drawbacks: (1) adding textual extensions negatively affect the readability of the requirements specifications, and (2) there is not a direct and clear way to specify that the selection of a requirement variant may have some influence on other requirements. These problems could be overcome using a MDSD approach, since the relationships between requirements, and also between requirements variants, could be made explicit without contaminating the requirements specifications with any kind of additional information.

As previously stated in the introduction, the use of feature diagrams [23] to model requirements variants is not sufficient [13]. We agree with the authors who argue that variability affects all the development stages and thus, it can be considered an orthogonal concern [39]. However, although modeling variability in a separate way may report some benefits, there are also some drawbacks, mainly: (1) keep all the information consistent, and (2) weave the variability with other software artifacts (and, in particular, with requirements), when needed. Explicitly modeling variability as part of the requirements specification, as offered in REMM, avoids these problems.

Goal-oriented models enable the description of variability by capturing the alternative ways by which stakeholders achieve their goals [21, 22]. As stated in [40], *goals can be ultimately represented or decomposed as regular requirements*. As a consequence, we have decided not to include them in REMM in order to keep the metamodel as simple as possible without loosing expressiveness.

Given that our proposal is in the MDSD context, and that we aim at including variability into our requirements metamodel, it is necessary to analyze whether there exists a variability reference model or not. Unfortunately, there is not such a standard variability model currently available, although some attempts towards its definition, such as the so called *Consolidated Variability Metamodel* (CVM) [41], or the very widely used *Orthogonal Variability Model* (OVM) [39], provide a good starting point.

Currently, Requirements Management Tools (RMT) do not support systematic requirements reuse [26, 32], and most of them neither requirements variability specification [27, 28]. Some approaches propose extending commercial RMT, e.g. DOORS with variability mechanism [42], or RequisitePro, with reuse capabilities [32]. Commonly, RMT focus on a single-project scope and the only reuse mechanism they provide is some kind of "copy and paste". However, just copying requirements is not enough for a systematic reuse approach, since it does not take into account the inter-requirements relationships, and the impact that selecting a certain requirement variant may have into other requirements.

5 Conclusions and future work

This paper has presented a systematic requirements reuse approach based on a Requirements Engineering Metamodel (REMM), which provides explicit variability modeling mechanisms. As a result, variability information can be directly included in the requirements models defined in terms of REMM, overcoming the limitations of other approaches which deal with textual requirements.

Similarly to the *Consolidated Variability Model*, which separates variability specification from resolution, REMM separates parameterized requirements definition from product requirements instantiation. On the other hand, REMM incorporates all the variability concepts defined in the *Orthogonal Variability Model*. Thus, OVM variability points correspond to REMM parameterized requirements, while OVM variants correspond to EnumType values defined in REMM. Correspondingly, variability dependences (optional, mandatory) in OVM are specified in REMM for each EnumLiteral of an EnumType, and the alternative choice in OVM corresponds to the range {minimum…maximum} in REMM EnumTypes.

The REMM-Studio tool has been implemented to support the two modeling facets of REMM, that is, its modeling *for reuse* facet and its modeling *by reuse* facet. As a consequence, REMM-Studio supports requirements variability specification and also requirements reuse. Unlike most current commercial requirements management tools, which only support a "copy and paste" requirements reuse policy from one project to another, REMM-Studio takes requirements dependences into account at reuse time and supports variability specification by means of parameterized requirements.

We are currently working in the definition of model queries to help selecting requirements from the repository. We are also working in a solution to provide product catalog reuse, i.e. converting catalogs of product requirements (product catalogs) into catalogs of reusable requirements (reusable catalogs). Finally, the main goal of our research around REMM and REMM-Studio is to provide the means to integrate requirements with other (high level) software development artifacts, that is, to integrate requirements into a complete MDD approach. To achieve this we plan to enrich REMM with some kind of forwards traces, probably to some domain specific models.

Acknowledgments

This work has been partially funded by the Spanish CICYT projects DEDALO (TIN2006-15175-C05-03, University of Murcia) and MEDWSA (TIN2006-15175-C05-02, Technical University of Cartagena).

References

1. Lo Giudice, D.: The State Of Model-Driven Development. OMG Technical Meeting, Brussels, Belgium (2007)
2. Champeau, J., Rochefort, E.: Model Engineering and Traceability. UML 2003. SIVOES-MDA Workshop, San Francisco. California (2003)
3. Vicente-Chicote, C., Moros, B., Toval, A.: REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting. Journal of Object Technology **6, no. 9** (2007) 437-454
4. OMG: OMG Systems Modeling Language (OMG SysML^{TM}) Specification. (2006)
5. Conrad, M., Fey, I., Buhr, K.: Integration of Requirements into Model-based Development. AuRE'04 (in conjuntion with RE'04), Nanzan University, Nogoya, Japan (2004)
6. Schätz, B., Fleischmann, A., Geisberger, E., Pister, M.: Model-Based Requirements Engineering with AutoRAID. Informatik 2005, Bonn, Germany (2005) 511-515
7. Shehata, M.S., Eberlein, A., Hoover, H.J.: Requirements Reuse and Feature Interaction Management. ICSSEA'02, Paris, France (2002)
8. Cybulsky, J., Reed, K.: Requirements Classification and Reuse: Crossing Domains Boundaries. 6th International Conference on Software Reuse (ICSR'2000). Springer, Lecture Notes in Computer Science, Viena (2000) 190-210
9. Cheng, B.H.C., Atlee, J.M.: Research Directions in Requirements Engineering. ICSE'07, Minneapolis, USA (2007) 285-303
10. Knethen, A.v., Paech, B., Kiedaisch, F., Houdek, F.: Systematic Requirements Recycling through Abstraction and Traceability. RE'02, Essen, Germany (2002) 273-281
11. Liaskos, S., Jiang, L., Lapouchnian, A., Wang, Y., Yu, Y., Leite, J.C.S.d.P., Mylopoulos, J.: Exploring the Dimensions of Variability: a Requirements Engineering Perspective. In: Pohl, K., Heymans, P., Kang, K.-C., Metzger, A. (eds.): First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS), Limerick, Ireland (2007)
12. Gomaa, H.: Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Addison Wesley Professional (2004)
13. Bühne, S., Lauenroth, K., Pohl, K.: Why is it not Sufficient to Model Requirements Variability with Feature Models? : International Workshop on Automotive Requirements Engineering (AuRE 2004) co-located at RE04, Nazan University, Nagoya, Japan (2004)
14. Bühne, S., Lauenroth, K., Pohl, K.: Modelling Requirements Variability across Product Lines. 13th IEEE International Requirements Engineering Conference, Paris, France (2005)
15. Tavakoli-Kolagari, R., Reiser, M.-O.: Reusing Requirements: The Need for Extended Variability Models. International Symposium on Fundamentals of Software Engineering (FSEN 2007), Tehran, Iran (2007)
16. Capilla, R.: Variability Description in Requirements for Product Family Support. International Workshop on Requirements Reuse in System Family Engineering. Co-located with International Conference on Software Reuse, Madrid, Spain (2004)
17. Berenbach, B., Gall, M.: Toward a Unified Model for Requirements Engineering. International Conference on Global Software Engineering (ICGSE 2006). IEEE Computer Society, Costão do Santinho, Florianópolis, Brazil (2006)

18. Maiden, N.A.M., Manning, S., Jones, S., Greenwood, J.: Generating requirements from systems models using patterns: a case study Requirements Engineering Journal **10** (2005) 276-288
19. Marschall, F., Schoenmakers, M.: Towards Model-Based Requirements Engineering for Web-Enabled B2B Applications ECBS'03, Huntsville, AL, USA (2003)
20. Vogel, R., Mantell, K.: MDA adoption for a SME: evolution, not revolution - Phase II. ECMDA 2006, Bilbao, Spain (2006)
21. Liaskos, S., Lapouchnian, A., Yu, Y., Yu, E., Mylopoulos, J.: On Goal-based Variability Acquisition and Analysis. 14th IEEE International Requirements Engineering Conference (2006)
22. Bennasri, S., Souveyet, C., Rolland, C.: Modelling Variability in Requirements with Maps ADVIS2004. LNCS 3261 (2004) 523-532
23. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, A.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, (1990)
24. Siegmund, N., Kuhlemann, M., Rosenmüller, M., Kaestner, C., Saake, G.: Integrated Product Line Model for Semi-Automated Product Derivation Using Non-Functional Properties. Second International Workshop on Variability Modelling of Software-intensive Systems (VAMOS 2008), Essen, Germany (2008)
25. Toval, A., Moros, B., Nicolás, J., Lasheras, J.: Eight Key Issues for an Effective Reuse-based Requirements Process. Computer Systems Science and Engineering ((accepted; publication pending))
26. Matulevicius, R.: Process Support for Requirements Engineering. A Requirements Engineering Tool Evaluation Approach. Department of Computer and Information Science. Faculty of Information Technology, Mathematics and Electrical Engineering, Vol. Doktor Ingeniør. Norwegian University of Science and Technology, Trondheim (2005)
27. Beuche, D., Birk, A., Dreier, H., Fleischmann, A., Galle, H., Heller, G., Janzen, D., John, I., Kolagari, R.T., Maßen, T.v.d., Wolfram, A.: Using Requirements Management Tools in Software Product Line Engineering: The State of the Practice. SPLC'07 (2007)
28. Bass, L., Clements, P., Donohoe, P., McGregor, J., Northrop, L.: Fourth Product Line Practice Workshop Report. CMU/SEI (2000)
29. Toval, A., Nicolás, J., Moros, B., García, F.: Requirements Reuse for Improving Information Systems Security: A Practicioner's Approach. Requirements Engineering Journal. Springer **6** (2002) 205-219
30. Toval, A., Olmos, A., Piattini, M.: Legal Requirements Reuse: A Critical Success Factor for Requirements Quality and Personal Data Protection. In: Press, I.C. (ed.): IEEE Joint International Conference on Requirements Engineering (ICRE'02 and RE'02), Essen, Alemania (2002) 9-13
31. Nicolás, J., Lasheras, J., Toval, A., Ortiz, F.J., Álvarez, B.: A Collaborative Learning Experience in Modelling the Requirements of Teleoperated Systems for Ship Hull Maintenance. LSO+RE 2006, Hannover. Germany (2006)
32. Toval, A., Moros, B., Nicolás, J., Lasheras, J.: Eight key issues for an effective reuse-based requirements process. Paper accepted in the International Journal of Computer Systems Science and Engineering, publication pending (2007)
33. Moros, B., Vicente-Chicote, C., Toval, A.: A Model-Driven Engineering Approach to Requirements Engineering: How These Disciplines May Benefit Each Other. 2nd International Conference on Software and Data Technologies (ICSOFT 2007), Barcelona, Spain (2007)
34. Vicente-Chicote, C., Moros, B., Toval, A.: REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting. Journal of Object Technology (JOT) **6** (2007)

35. Leffingwell, D., Widrig, D.: Managing Software Requirements: A Use Case Approach, Second Edition. Addison Wesley (2003)
36. Schmid, K., John, I.: A Practical Approach To Full-Life Cycle Variability Management. International Workshop on Software Variability Management (SVM 2003), Portland, Or. (2003)
37. John, I., Muthig, D.: Product Line Modeling with Generic Use Cases. SPLC-2 Workshop on Techniques for Exploiting Commonality Through Variability Management, Second Software Product Line Conference, San Diego (2002)
38. Pohl, K., Weyer, T.: Documenting Variability in Requirements Artefacts. In: Pohl, K., Böckle, G., Linden, F.v.d. (eds.): Software Product Line Engineering. Foundations, Principles and Techniques. Springer (2005)
39. Lauenroth, K., Pohl, K.: Principles of Variability. In: Pohl, K., Böckle, G., Linden, F.v.d. (eds.): Software Product Line Engineering. Foundations, Principles and Techniques. Springer (2005)
40. Kaindl, H., Smialek, M., Svetinovic, D., Ambroziewicz, A., Bojarski, J., Nowakowski, W., Straszak, T., Schwarz, H., Bildhauer, D., Brogan, J., Mukasa, K., Wolter, K., Krebs, T.: Requirements Specification Language Definition - Defining the ReDSeeDS Languages, Institute of Computer Technology, Vienna University of Technology (2007)
41. Bayer, J., Gerard, S., Haugen, O., Mansell, J., Moller-Pederson, B., Oldevik, J., Tessier, P., Thibault, J.-P., Widen, T.: Consolidated Product Line Variability Modeling. In: Timo Käköla, J.C.D. (ed.): Software Product Lines. Research Issues in Engineering and Management. Springer (2006)
42. Schmid, K., Krennrich, K., Eisenbarth, M.: Requirements Management for Product Lines: Extending Professional Tools. SPLC'06 (2006)