Malaysian Journal of Computing, 3 (2): 119–137, 2018
Copyright © UiTM Press
eISSN: 2600-8238 online

META-MODELING CONSTRUCTS FOR REQUIREMENTS REUSE (RR): SOFTWARE REQUIREMENTS PATTERNS, VARIABILITY AND TRACEABILITY

Badamasi Imam Ya'u[1], Azlin Nordin[2,] and Norsaremah Salleh[2]

Department of Computer Science
Kulliyyah of Information and Communication Technology (KICT)
International Islamic University Malaysia, Gombak, Kuala Lumpur, Malaysia
[1]badamasi.imam@live.iium.edu.my, {[2]azlinnordin, [3]norsaremah}@iium.edu.my

ABSTRACT

Reuse is a fundamental activity, which increases quality and productivity of software products. Reuse of software artifacts, such as requirements, architectures, and codes can be employed at any developmental stage of software. However, reuse at a higher level of abstraction, for instance at requirements level, provides greater benefits in software development than when applied at lower level of abstraction for example at coding level. To achieve full benefits of reuse, a systematic approach and appropriate strategy need to be followed. Although several reuse approaches are reported in the literature, these approaches lack a key strategy to synergize some essential drivers of reuse, which include reusable structure, variability management (VM) and traceability of software artifacts. In line with this, we make our contribution in this paper by (1) presenting the concepts and importance of software requirements patterns (SRP) for reusable structure; (2) proposing a strategy, which combines three sub-disciplines of Software Engineering (SE) such as Requirements Engineering (RE), Software Product Line Engineering (SPLE) and Model-driven Engineering (MDE); (3) proposing a meta-modeling constructs, which include SRP, VM and traceability and; (4) Relationship amongst the three sub-disciplines of the SE. This is a novel approach and we believe it can support and guide researchers and practitioners in SE community to have greater benefits of reuse during software developments.

Keywords: *meta-model, Requirements reuse (RR), software requirements patterns (SRP), traceability, variability modeling (VM)*

1. Introduction

It is obvious that reuse is a SE practice, which is central to all software development activities (Franch, Palomares, Quer, Renault, & De Lazzer, 2010). It can be achieved through a number of approaches, such as component-based software development (Basha & Moiz, 2012; Ya'u, 2015), object-oriented and aspect-oriented software development (Nerurkar, Kumar, & Shrivastava, 2010) among others.

Nevertheless, reuse is not optimized as many software developers opportunistically apply in the lower abstraction levels for example, at design, runtime and implementation. A research shows that, when reuse is applied at highest level of abstraction (requirements analysis stage), which comprises of elicitation, analysis and documentation, all artifacts at subsequent stages related to the reusable requirements, such as test case, specification, design and codes are also reused and hence minimizes substantial effort (Goldin, Matalon-Beck, & Lapid-Maoz, 2010). This indicates that, reuse of software artifacts at the initial stage of development is far more advantageous than at any other stage of development (Bakar & Kasirun, 2014). The benefits of reuse is evident, especially in producing high quality product, perhaps with little modification (Liang, Avgeriou, & Wang, 2011). Consequently, applying reuse at requirements level can improve the quality of software, reduce development cost and shorten time to markets (Benitti & Silva, 2013; Chernak, 2012; Goldin & Berry, 2013; Hauksdóttir, Mortensen, & Nielsen, 2013).

SE is viewed as multi-disciplinary field, which connects a number of social and technological boundaries (Easterbrook, Singer, Storey, & Damian, 2008). It embodies the development, maintenance and management of quality software through cost-effective ways (Sjoberg, Dyba, & Jorgensen, 2007). These cost-effective ways may involve reuse of software artifacts (tangible by-products of software development, such as requirements, use cases, models among others), thus providing software development solutions and reduction of products time to markets (Chernak, 2012).

Due to increased demands from customers, there are unpredictable and frequent changes in businesses, marketplaces and competitors. Therefore, the way products are developed and projects are executed, also changed since, requirements are changed indefinitely to suit customers' needs (Gabriel, 1996). Requirements are the model of any system intended to be developed (Hoffmann, Kühn, Weber, & Bittner, 2004) and therefore provide the specifications of what should be implemented (Wiegers & Beatty, 2013). Because of the complexity of the increasing changes in business, requirements should be reused rather than reinvented from scratch (Hauksdóttir et al., 2013; Zhang, Nummenmaa, Guo, Mai, & Wang, 2011).

Because of this, the aim of this paper is to propose a systematic RR strategy, with the aid of a meta-model, which represents SRP, VM and traceability. Thus, our contributions are: (1) integration of SRP, VM and traceability in a meta-model (2) theoretically linking the concept of RE, SPLE and MDE to enhance systematic reuse.

It is discovered that, the reuse of software requirements is beneficial to developers, particularly during requirements elicitation, analysis, validation and documentation phases (Srivastava, 2013). In a previous research (Ya'u, Nordin, & Salleh, 2016a), we reported a number of approaches to RR, which include domain-specific, pattern-based, ontology-based and general approaches. Pattern-based are recognized in providing consistent and reusable structure for RR (Benitti & Silva, 2013; Franch et al., 2010). To promote RR, we propose the adoption of a meta-model strategy, which binds SRP, VM and traceability.

1.1. Motivation

Many domains such as insurance, banking, health, airlines, education, automotive and other consumer electronics deal with many sets of requirements within the same application domain or product families. Customers and end users are now in haste looking for latest, fast, and efficient interfaces, applications and products that can cater their social needs. For example, in software product families, utilization of family assets to produce subsequent products is emphasized. That is, no need for development from scratch; instead, new products are derived from the family assets (e.g. requirements) with little modification.

In this way, RR gives opportunity to build products in a consistent fashion with reduced time and frequency of error occurrences (Wiegers, 2005). RR therefore, has the potential to reduce the cost, effort and time to markets (Benitti & Silva, 2013; Chernak, 2012; Goldin & Berry, 2013; Hauksdóttir et al., 2013). This is due to its flexibility as it can be applied at any phase of RE lifecycle for instance, from requirements elicitation to documentation. It is presumed that the earlier reuse is applied, the greater benefit of reuse is realized (Benitti & Silva, 2013; Goldin et al., 2010; Velasco, Valencia-García, Fernández-Breis, & Toval, 2009). Therefore, the benefit of RR at phase affects the subsequent phases of the RE lifecycle (Bakar & Kasirun, 2014; Benitti & Silva, 2013; Goldin et al., 2010). To illustrate this statement, reuse of requirements in practice, involves reuse of other associated activities and knowledge, which include reuse of test cases, designs and analysis (Monzon, 2008). In addition, an increase in dependability, a reduction of risk and an increase in quality are also benefited from RR (Sandhu, Aashima, Kakkar, & Sharma, 2010).

The remainder of this paper is as follows: we discuss the importance of choosing an SRP for achieving RR in Section 2; Section 3 presents meta-model approaches for reuse; we discuss our proposed meta-modeling constructs in Section 4; relationships amongst RE, MDE and SPLE are presented in Section 5; Section 6 presents the discussion; and Section 7 presents the conclusion and future work of the paper.

2. Software requirements patterns (SRP)

Patterns appear to be prominent among many reuse approaches as each pattern describes a recurring problem together with the solution of this problem, which is applied over and over again (Franch et al., 2010). A requirement pattern is defined as a template and guidelines for writing a requirement (Palomares Bonache, Quer Bosor, Franch Gutiérrez, Guerlain, & Renault, 2012; Palomares, Quer, Franch, Renault, & Guerlain, 2013; Srivastava, 2013). The templates and the catalogs in which the requirements patterns are presented ensure a standardized structure for enhancing systematic RR (Ya'u et al., 2016a). Requirements patterns are described as reuse approach, which is similar to design pattern that can be applied in requirement specification (Konrad & Cheng, 2002). As such, SRP offers significant percentage of reuse for both functional and non-functional requirements (Srivastava, 2013). When SRP is applied in RE, it produces all software requirements related to the objectives of a particular pattern (Palomares Bonache et al., 2012). Although requirements patterns possess a generic form, their generic nature is restricted as RE overlaps with architectural design (Slavin, Shen, & Niu, 2012).

Among existing patterns are requirement patterns particularly those introduced in (Withall, 2007) as a suitable way of writing software requirements with less effort and greater precision. Withall (2007), defines requirements patterns as an approach to specifying a requirement. He presents 37 reusable patterns, including templates and examples as a framework for writing general software requirements. Withall's requirements patterns are therefore regarded as more detailed and complete compared to other pattern catalogues (Benitti & Silva, 2013). In general, requirements patterns enable organizations to reuse requirements knowledge from previous projects instead of starting from the scratch.

SRP can be used at different phases of RE, which include elicitation, analysis, validation and specification (Franch et al., 2010; Srivastava, 2013). It has been considered as an artefact that fosters RR (Palomares Bonache et al., 2012; Palomares et al., 2013). SRP is therefore viewed as advantageous in software development lifecycle as it: offers RR through guidelines; improves the quality and consistency of requirements through uniform style; improves requirements management through traceability (Palomares et al., 2013). In addition, SRP facilitates reuse at design and code levels since, implementations are indexed with their requirements patterns (Konrad & Cheng, 2002).

In summary, SRP promotes RE process through reuse of requirements, production of quality requirements and traceability amongst reusable requirements (Palomares Bonache et al., 2012; Palomares et al., 2013).

However, realizing the benefit of reuse at the early stage of software development also requires an adequate framework to affirm the structure of the reusable artifacts as well as appropriate tool that facilitates the reuse process (López, Laguna, & Peñalvo, 2002b). Furthermore, management of different models and notations for RR can be achieved by using high level of abstraction such as meta-modeling, which can describe a formalization of concepts, relations and common features of these models. In line with this, concept and importance of meta-modeling is presented in the next section.

3. Meta-model approaches for reuse

Software reuse improves software productivity, especially when reuse of software artifacts is applied at the early stage of software development (Benitti & Silva, 2013; Goldin et al., 2010; López et al., 2002b; Velasco et al., 2009). RR in particular can empower and make software development lifecycle more profitable. However, research has shown that, availability of different notations, formats and granularity of requirements contribute in making RR challenging, particularly its core activities such as representation, classification, storage, selection and modification of reusable assets (López et al., 2002b; Seman et. al., 2010).

Table 1**:** Meta-model approaches for reuse

Study	**Scope**	**Notation**	**Application**	**Meta-model**			
				Traceability?	***VM?***	***SRP?***	***Tool?***
(Moros et al., 2008)	General purpose	Object models	Variability modeling	No	Yes	No	Yes
(López et al., 2002a)	General purpose	Semi-formal diagrams	requirements model	No	No	No	Yes
(Franch et al., 2010)	General purpose	Natural language	SRS	Yes	No	Yes	No
(Bachmann et al., 2003)	SPL	UML	Variability modeling	Yes	Yes	No	No
(Gomaa & Shin, 2002)	SPL	UML	Variability modeling	No	Yes	No	Yes
(Cavalcanti et al., 2011)	SPL	UML	Variability & Traceability	Yes	Yes	No	Yes
(Goknil et al., 2008)	MDE	SysML	SRS	No	No	No	Yes
(Goknil et al., 2013)	MDE	Product-line/ SysML	SRS	No	No	No	Yes
(Cerón et al., 2005)	SPL/ MDE	UML	Software Process	Yes	No	No	Yes
(Navarro et al., 2006)	General Purpose	UML	SRS	Yes	Yes	No	Yes
(Moon et al., 2007)	SPL	UML	Variability Management	Yes	Yes	No	No
Our proposed Meta-Modeling Constructs for RR	RE/SPL/ MDE	Natural language/ UML	Requirements Analysis & SRS	Yes	Yes	Yes	Yes

A meta-model provides a specification which a modeling process should fulfill, through definition of the epistemology and design foundation of the modeling process, which consists reasoning processes, proofs, logic, rules, constructs, axiom of validity among others (Van Gigch, 2013). Meta-modeling is a component of every system design problems and neglecting it incurs a major flaw of many system designs (Van Gigch, 2013). As such it is imperative to adopt meta-modeling as part of a framework to support and empower RR.

As we pointed out in our previous work (Ya'u, Nordin, & Salleh, 2016b), there are very little work reported in the literature on meta-modeling for the enhancement of RR despite their reported importance in software development. Table 1 summarizes meta-modeling approaches for reuse according to certain criteria, such as the scope of reuse, different notations used, area of application and the major constructs of the meta-model. Among dozens of RR approaches available in the literature, we could not find many studies, which glaringly address RR in form of a meta-model. Furthermore, from these meta-modeling approaches, it can be seen that 3 are general purpose meta-modeling approaches from which two (López, Laguna, & Peñalvo, 2002a; Moros, Toval, & Vicente Chicote, 2008) do not use SRP in their proposals. For instance, in (Moros et al., 2008), the main focus is on modeling variability in requirements models to enable RR from two facets, that is, modeling for reuse and with reuse. In that approach, traceability of requirements was facially discussed. If we look at (López et al., 2002a), beside exclusion of SRP in their approach, neither traceability nor variability management (VM) was applied. They took advantage of meta-modeling concept and focused on the integration of semi-formal diagrams for achieving RR. Two approaches, (Bachmann et al., 2003; Gomaa & Shin, 2002) are domain-specific; they use SPL principles, which focus on VM, using UML notation; none of these two approaches explicitly address RR. The approach (Franch et al., 2010), proposes a meta-model for SRP, which describes a form of requirements traceability. However, VM is not addressed in this approach.

A meta-model approach for supporting variability and traceability was presented in (Cavalcanti et al., 2011). The aim was to coordinate activities in SPL by managing and maintaining traceability and variability among different artifacts in various phases of software development. The meta-model in that approach provides support in different aspects such as scoping, requirements, tests, and project and risk management. The approach is supported by a web tool, using Django framework. The meta-modeling aspect does not focus on RR nor does it use SRP in the requirements analysis.

In (Goknil, Kurtev, & van den Berg, 2008), a meta-model for requirements model called core meta-model and an approach for customizing this core meta-model were presented. The core meta-model enables reasoning on requirements, thus allowing detection of implicit relations and inconsistencies within the requirements based on formalization of concepts and relations defined in the core meta-model. The approach applied web ontology language (OWL) technique and aimed at providing an avenue for reusing tool such as reasoners. The approach therefore, combines RE and MDE methodologies. Nonetheless, SRP, traceability and variability were not treated as part of the meta-modeling aspect.

In an extension for (Goknil et al., 2008), an additional feature for reasoning on requirements and their relation in multiple requirements modeling approaches was presented in (Goknil, Kurtev, & Millo, 2013). The idea is to use requirements meta-model as a core meta-model specialized for different requirements modeling approaches and notations such as product-line and SysML. The specialization allows the use of the same semantic (given in first order logic) and reasoning mechanism for the core meta-model for multiple requirements modeling approaches, thus enables change impact analysis. A Tool for Requirements Inferencing and Consistency Checking (TRIC) was developed and requirements, their relations and properties are mapped to OWL during the tool implementation.

A meta-model for RE in Systems Family context was presented in (Cerón, Dueñas, Serrano, & Capilla, 2005). The work discussed the important issues of System Family Engineering (SFE) in requirements modeling, which contains common and variable parts; functional and non-functional aspects. Capability Maturity Model Integration (CMMI) was used as a base to improve software process. A meta-model, which covers several of the specific needs of SFE concerning requirements management and traceability, was presented. The approach emphasized on feasibility of adopting RR in SFE. Although the approach combines RE, SFE and MDE, application of SRP and variability management in RE and SFE respectively were not covered.

In (Navarro, Letelier, Mocholi, & Ramos, 2006), a meta-modeling approach for integration and scalability of RE concepts was presented. The approach combined RE and MDE, thus claimed management of traceability and variability in the RE concepts with the aid of MetaEdit+, a tool support for modeling and meta-modeling. Employment SRP to structure the requirements artifacts was the focus of the approach.

A meta-modeling approach for tracing variability between requirements and architecture was presented in (Moon, Chae, Nam, & Yeom, 2007). At the first stage, two meta-models for representing domain requirements and domain architecture with variability were presented. In that stage, trace relationships between requirements and architecture with respect to variability was described. In the second stage, another meta-model, a variability trace meta-model was defined as a means of realizing and coordinating the interrelationships of the two meta-models, the domain requirements and the domain architecture meta-models. The approach did not use SRP for structuring the requirements nor did it present tool for automating for tracing the variability between the requirements and the architecture.

To promote systematic RR and fully exploit the potential benefit of RR in software development processes, we need to integrate both traceability, VM and SRP in a meta-model as we propose in Section 4.

4. Methodology

This section presents the methodology employed for the proposed approach, that is, the section describes from the beginning where we started our research on RR to the current stage according to the following steps depicted in Figure 1.

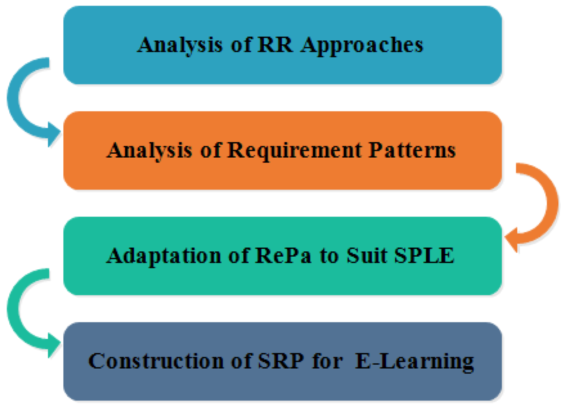
Analysis of RR Approaches
Analysis of Requirement Patterns
Adaptation of RePa to Suit SPLE
Construction of SRP for E-Learning

Figure 1: Methodology steps for RR

4.1 Analysis of RR approaches

Reuse of software artefacts has been interesting topic of research for decades. However, reviews from the literature reveal that the current state of reuse practice needs to be revolutionized to meet customers and organizational needs. This motivated us to investigate in detail what RR reuse approaches exist hitherto and what RR challenges so far reported in the literature. As one of our objectives, the result and details of the existing RR and the challenges were published in our previous research (Ya'u et al., 2016a). In a null shell, we have discovered that amongst the popular RR approaches in the literature include domain-specific, pattern-based, ontology-based and general approach. In the other hand, the major challenge of RR reuse has been a systematic reuse structure, awareness and tool support.

4.2 Analysis of requirement patterns

In line with the result published in (Ya'u et al., 2016a), we have discovered that pattern-based approach has the higher potential to leverage RR in terms of consistent structure. One of the major challenges of RR is systematic structure for reuse. To fill this gap, we analyzed the capabilities of requirement patterns reported by many researchers in the literature as we discuss in Section 2 especially the work published in a book by Withal (Withall, 2007). After scrutinizing various pattern templates, we found that requirements pattern provides a structure in which detailed information required to specify and reuse requirements is logically organized. The anatomy of requirements pattern includes pattern author, related pattern, applicability, problem and solution to mention a few.

4.3 Adaption of RePa to suit SPLE

RePa is an International Workshop on requirements patterns, which was organized to provide a standard requirement patterns template for specifying requirements (Chung, Paech, Zhao, Liu, & Supakkul, 2012). Since many specific requirements patterns exist in the literature, the common template was designed to provide uniformity for requirements patterns cataloging. The template consists of three sections, the required, optional and custom. In our approach, we use the custom section to complement the work of Withal by adding 'Consideration for Design' sub-section. This sub-part is of utmost important when designing software development process especially when dealing with the discrepancies between problem and solution domains. This also harmonizes development processes and sub-processes in SPLE for instance reuse of software artefacts from domain requirement engineering through design sub-process to testing sub-process. Because of the vast and detailed information required to manage commonality and variability of requirements and the complexity due to the size of the scope of the product line, we believe that requirement pattern approach can play a vital role in orchestrating different types of requirements.

4.4 Construction of SRP for e-learning

Having considered and adapted RePa template for SPLE, we explore from various sources of software requirements specification or documents in the literature. However, due to intellectual property right. And other constraints, it was difficult to retrieve as many SRS as we intended to find. Nonetheless, e-learning, mobile, insurance (Takaful) medical records requirements were retrieved. Because of the quantity and authenticity of the sources, we selected to use e-learning domain as an example to evaluate our proposed requirements pattern template. As it can be seen,

Table 2 presents the details of *Inquiry* requirement pattern, which also includes the custom section we mentioned earlier. This guides requirement engineering or developer to understand in deep what requirement of this type constitutes. To implement and reuse the requirement of this type,

Table 3 describes the solution section of the *Inquiry* requirement pattern, which comprises of pattern goal, primary/ main requirement, common and variable requirements and variability model information.

Table 2: Inquiry requirement pattern

Section			**Description**	
Pattern ID			RP5	
***Pattern Name**			Inquiry	
Also Known As			NA	
Authors			Stephen Withal	
Date Created			2017-01-01	
***Context/ Applicability**	****RE Activity***		Specification	
	****Pattern Type***		Product	
	Business Domain		E-learning	
	Organization Environmental Factors		Teaching and Learning Environment	
	Stakeholders	***Role***	Students, Instructors, Teachers, Administrators	
		Goal	To use e-learning application in running and delivering their organization responsibilities	
***Problem AKA Intent and Objective**			Poor security measures to protect unauthorized access to information system	
***Force**			A cutting-edge e-learning security facility to protect teaching and learning applications	
***Solution**			Refer to "Solution" Section	
***Application and Example**			***Application:*** This pattern is applied in the events where inquires on information stored in a database are displayed to the user	
			Example: The system shall display information on the screen for all inquiry on the database.	
***Known Uses**			Web-based and desktop applications.	
Cataloguing:	***Classification***	*Type*	Functional	
		Default Value		
		Purpose	This indicates whether the functionality of this requirement that shall be provided by the system is satisfied	
		Audience Role	Software and requirement engineers	
		Audience Goal	Software requirement specification for SPL	
		Allowed Value	True	
	Related Pattern	*ID*	RP6	
		Name	Report	
		Relation Type	***Extends***	Yes
			Refers	No
Custom Section	***Consideration for Design***	*Description*	This describes the aspect of the design that should be considered for the requirements of this type.	

		Purpose	This highlights the reason why the design for the implementation of the type of requirement is considered
		Constraint	This provides with those design constraints a software designer should consider.
		Design Pattern	This lists the name of the design pattern that corroborates with this requirement pattern.
		Design Guide	This highlights a step by step guide for designing the implementation of requirement of this type.
	Consideration for Development	*Description*	This describes the needs for considering the development of the functionality of requirement of this type.
		Purpose	This details the purpose for considering the implementation of requirement of this type
		Constraint	This clearly shows the kinds of constraints that affect the implementation of requirement of this type
		Development Guide	For this type of pattern, the following have to be considered; 1. Check the availability of information 2. Find out whether there are potential performance concerns 3. If display is refreshed automatically, how easy is it to achieve that in the prospective user interface environment?
	Consideration for Testing	*Description*	This describes the needs for testing the functionality of requirement of this type.
		Purpose	This states the reasons for considering the testing for the functionality of the requirement of this type.
		Constraint	This describes the constraints for testing the requirement of this type.
		Test Type	This part states the type of testing executed for the function of the requirement of this type
		Test Guide	To be satisfied with the Inquiry requirements, test it by displaying the inquiry to verify that it shows what is intended to show. For example, identify all types of information that must be viewable including all database tables.

Table 3: Solution section of inquiry

Solution ID	PS5.1		
Pattern Name	Inquiry		
Goal	Display inquiry		
Description	This pattern for is for specifying inquiry requirements from the system.		
Requirement	***ID***	RQ5.1.1	
	Name	Financial transaction	
	Type	Functional	
	Description	The system shall provide a function of a requirement of this type to enable a user to make an inquiry for the information stored in the system	
	Priority	High	
Common Requirement Form	***ID***	CR5.1.1.1	
	Description	This form establishes inquiry requirements for information stored in the database of the system	
	Constraints	Fixed part (1) Extended part: 1. Selected customer 2. Selected data range	
	Fixed Part	*Form Text*	There shall be an inquiry that shows the details of financial transaction.
	Extended Part	*Form Text*	The system shall allow an inquiry of financial transaction of a customer based on the following: 1. selected customer 2. selected data range
Variable Requirement Form	***ID***	VR5.1.1.1	
	Description	This form shows variable requirements for specifying different variation points of Inquiry requirement pattern	
	Constraints	Fixed part (1) Variable part: 1. by credit 2. by cash	
	Fixed Part	*Form Text*	The system shall display an inquiry of financial transaction either made by credit or cash
	Variable Part	*Variation Points (VP)*	Inquiry
		Variants (V)	1. credit 2. cash
Variability Model Form	***Description***	This form establishes the need to use orthogonal variability model to show and trace the level of variations in different requirements artefacts.	
	Constraints	Focus on orthogonal variability models	
	Model (s)	Textual requirements, feature models, traditional requirement model, UML models	

5 Our proposed meta-modeling constructs

In this section, we present our proposed meta-modeling constructs, which we believe they can support in realizing a systematic RR. The proposed approach is a general framework to systematic RR, comprising both design facets (design for reuse and design with reuse) and can be applied to any software product family. To achieve our aim, our approach combines three sub-fields of SE: RE, SPLE and MDE. RE covers various phases of software developments such as elicitation, identification, analysis, modeling etc. Our proposed area of application includes requirements analysis (elicitation, analysis and documentation) and writing of SRS. Because, these activities usually commence at the initial stage of software development, which offers more reuse benefits as discussed earlier. Through RE processes, we can exploit potential benefits of SRP. SRP enables uniform development of system specifications, thus making understanding and maintenance of these specifications easier (Konrad & Cheng, 2002). From their capability of capturing proven knowledge, requirements patterns are thought to be a powerful tool for streamlining RE processes (Mahendra & Ghazarian, 2014). Concept and benefits of using SRP to promote RR in software development are previously discussed in Section 2.

SPL is a popular and successful reuse approach in software development for systems of family, which is known in commonality and variability management of reusable software artifacts in the product families (Sinnema, Deelstra, Nijhuis, & Bosch, 2004). The ultimate objective of product line engineering is improvement of productivity such as reduction of development time and cost as well as increasing quality of products (Royer & Arboleda, 2013). However, SPLE demands a mature SE, planning and reuse, adequate practices of management and development as well as having capability to manage organizational issues and architectural complexity, which require the support of auxiliary methods and tools (Cavalcanti et al., 2011).

The future trend in SPLE is to automate its production plan. A successful technique for defining an executable tool chain is MDE (Royer & Arboleda, 2013). In the context of SPL therefore, modeling is seen as a mechanism to define and represent variability involved in a family of products (Cavalcanti et al., 2011). In this case, a meta-model can help tremendously in capturing variability and commonality in SPLE (Royer & Arboleda, 2013).

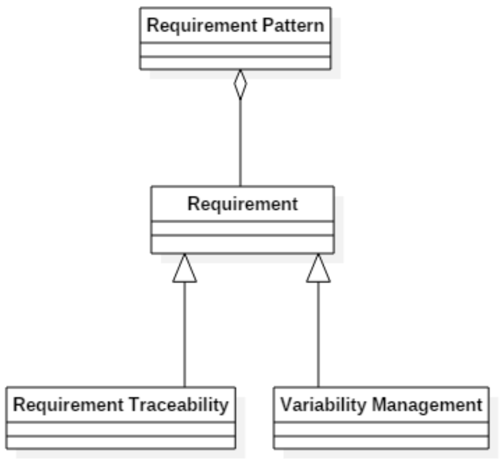

Figure 2: **Proposed meta-modeling constructs**

Figure 2 presents a meta-model comprises of the 3 proposed constructs: (1) SRP (2) VM and (3) Traceability. These together provide a systematic approach to RR in software development.

We include variability modeling in our proposal as being regarded an essential task during analysis phase and also a crucial activity in developing SPL (von der Maßen & Lichter, 2002). The modeling aspect of variability helps developers to have deep understanding of commonalities and variabilities in SPL and as well supports product derivations (Czarnecki, Grünbacher, Rabiser, Schmid, & Wąsowski, 2012; Sinnema & Deelstra, 2007). VM is also described as explicit representation of variability in software product families (Bachmann et al., 2003; Sinnema et al., 2004). This happens by treating the introduction, use and evolution of variability. That is, the ability to modify a system or artifacts in a specific context (Sinnema & Deelstra, 2007).

6 Relationship concerning RE, SPLE and MDE

We discover some good relationships combining RE, SPLE and MDE in our proposed approach. Their relations can help achieve the software productivity we mentioned earlier. RE, deals with all aspects of obtaining quality requirements, which include requirements gathering, analysis, negotiation and documentation. Some activities to improve the quality, structure and consistency of requirements, such as SRP and traceability are therefore required. In addition, requirements are considered mostly as textual artifacts, whose structure often not explicitly specified (Goknil et al., 2008). Since requirements are one of the initial system models, it is important to represent requirements description as models. This can in fact keep the continuum of models in MDE, where every artifact is treated as a model. Representing requirements descriptions as model can only be achieved by employing a meta-model for requirements (Goknil et al., 2008).

Since the main aim is to achieve systematic RR in product families, SPLE is a core domain in reuse enhancement, which brings benefits in terms of costs and productivity. SPLE therefore, encompasses both domain and application engineering phases, which deal with management of requirements commonalities and variabilities respectively. The two engineering processes of SPLE, domain and application are also referred as development for reuse and development with reuse respectively (Royer & Arboleda, 2013).

Furthermore, MDE techniques and tool also have the potential to improve the quality and productivity of SE processes. MDE paradigm emphasizes on three main concepts, which are models, meta-models and model transformation. It uses software modeling as primary document, which consists of requirements, feature model, use cases, unified modeling language, architectures among others (Royer & Arboleda, 2013). MDE provides automation to SE processes at every stage of development. In relation with SPLE, MDE is considered a promising discipline, which provides uniformity and abstraction for software artifacts and processes within SPLE (Royer & Arboleda, 2013).

From the Venn diagram shown in Figure 3, the three sub-disciplines are interrelated with the constructs of our proposed meta-modeling approach, in particular traceability. In RE, traceability helps in describing, following and understanding the life of requirements and their impact on other artifacts (Champeau & Rochefort, 2003); it also assist in identification of pairs during verification and validation (Winkler & Pilgrim, 2010; Liew et. al., 2010).

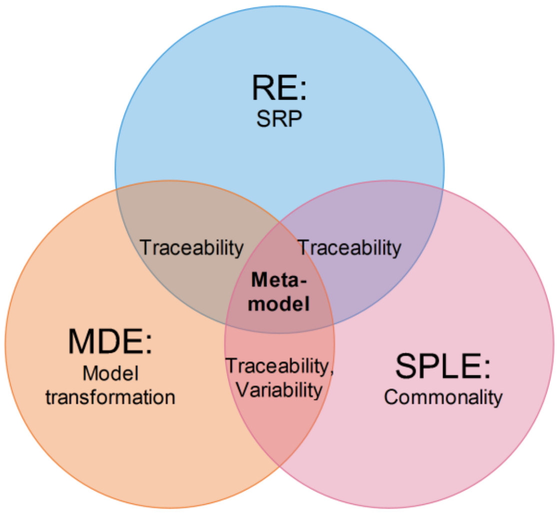

Figure 3: Interrelationships of the combined SE sub-disciplines

In relation with SPL, traceability is recognized by researchers and practitioners as a key aspect, which manages the complexity of commonalities and variabilities in SPL engineering (Anquetil et al., 2008). In the context of model-driven engineering (MDE), traceability also helps in understanding the existence of many dependencies between MDE artifacts (Paige, Olsen, Kolovos, Zschaler, & Power, 2008); it keeps the models consistent and supports propagations between these models (Winkler & Pilgrim, 2010). As such, the application level of traceability in software development is considered as a measure of system quality and process maturity, which is authorized by many standards (Aizenbud-Reshef, Nolan, Rubin, & Shaham-Gafni, 2006). Another important construct in our meta-modeling approach is management of variability, which is a common activity in SPLE and MDE. Management of variability is necessary in software development if a meta-model incorporates some kind of reuse mechanism (Moros et al., 2008). Furthermore, VM is the core task that distinguishes conventional SE and SPLE (Bachmann et al., 2003; Berger et al., 2013).

7 Discussion

As reported in (Ya'u et al., 2016a), though the domain-specific, pattern-based, ontology-based and general approaches address RR problems in some way, using single technique in the existing approaches has a peculiar weakness. It is apparent in the literature that, most of the domain-specific approaches use feature modeling to capture commonalities and variabilities in requirements, but the variability information captured by feature models is incomplete (Moros et al., 2008). Furthermore, feature modeling is thought to be time-consuming, expensive and perhaps limits the opportunity of reuse outside a particular application domain (Naish & Zhao, 2011). For this reason, there is need for further research to simplify the modeling aspect to represent variability and commonality in a cost-effective manner.

In another way, ontologies help especially in formalizing requirements to improve quality and enhance reuse. Nevertheless, formal representation is a developer-oriented technique, which requires additional information to be understood and reused (Zhang et al., 2011). It was also discovered that, most ontology-based approaches depend on static knowledge instead of dynamic knowledge, which offers more reuse opportunity (Zong-yong, Zhi-xue, Ying-ying, Yue, & Ying, 2007). General approaches in their case, offer broader scope and greater opportunity for reuse in variety of domains. However, the more generic approach is, the more time it consumes for detailed analysis and description thus, reducing the benefit of reuse in terms of development timeframe (Wiegers & Beatty, 2013).

In the case of pattern-based approach, of course pattern provides reusable structure. However, it was discovered in a survey (Mahendra & Ghazarian, 2014) that, the concept of pattern benefits only few software developers due to the following reasons: 1) requirements pattern catalogues are not easily accessible to researchers and practitioners; 2) there is late growth trend in construction of pattern catalogues and; 3) there is lack of tool to support the implementation of patterns. It was reported that, there is still few proposals on SRP, which are basically distinguished in criteria such as scope, formalism for constructing patterns, usage and goal of patterns and underlying meta-model for patterns (Franch et al., 2010). As stated earlier, SRP is of utmost important in any software related development processes, due to its recognized nature of enhancing reuse. Nevertheless, SRP requires a well-defined and broader underlying meta-model, which would describe more concepts.

As discussed in Section III, like SRP, meta-model approaches to RR are also few in the literature and proposals based on these meta-models are generic and therefore, have limiting power to reuse. Furthermore, the meta-modeling approaches do not consolidate all key aspects that enhance reuse, which include consistent and reusable structure (in this case, SRP), VM and traceability of reusable software artifacts.

Based on these findings, it is noticeable that, the existing RR approaches have limitations in providing solution to systematic RR. We therefore recommend that, solutions of the existing approaches should be integrated in a new strategy that could synergize and consolidate RR technique. Furthermore, integrating RE, SPLE and MDE can open more new research trends, that can guide and support researchers and practitioners in utilization of reuse opportunity, thus increasing software productivity.

8 Conclusion and future work

In this paper, we present the concepts and evidences that show the importance of SRP and meta-model in SE. We observed that, available meta-modeling approaches for RR in the literature fall short in consolidating key elements to synergize reuse. These include reusable structures, VM and traceability of reusable software artifacts. This indicates that, there is a clear gap to accomplishing systematic RR. To fill this gap and highlight our contribution, we propose a meta-modeling strategy for RR, which encompasses SRP, VM and traceability. Our approach syndicates RE, SPLE and MDE sub-disciplines of SE and can be applied in any software product family development. We believe that, this approach can empower systematic RR in software development lifecycle. To the best of our knowledge as we reported in a previous research (Ya'u et al., 2016a), there is no approach in the literature that reported such a meta-model that incorporates SRP, VM and traceability of requirements. As such, our proposal is novel and can help developers in RE, SPL, MDE and SE in general to exploit greater benefits of reuse, which gyrates across cost effectiveness, quality products and time to market.

For our future work, we are currently working on the last portion of the meta-model and implementation of a tool support to demonstrate our framework in the e-learning domain. In addition, a quasi-experiment with requirement engineering final year undergraduate student has been designed to evaluate the correctness of the tool.

References

Aizenbud-Reshef, N., Nolan, B. T., Rubin, J., & Shaham-Gafni, Y. (2006). Model traceability. *IBM Systems Journal, 45*(3), 515-526.

Anquetil, N., Grammel, B., Galvao Lourenco da Silva, I., Noppen, J., Shakil Khan, S., Arboleda, H., . . . Garcia, A. (2008). Traceability for model driven, software product line engineering.

Bachmann, F., Goedicke, M., Leite, J., Nord, R., Pohl, K., Ramesh, B., & Vilbig, A. (2003). *A meta-model for representing variability in product family development.* Paper presented at the International Workshop on Software Product-Family Engineering.

Bakar, N. H., & Kasirun, Z. M. (2014). Exploring Software Practitioners' Perceptions and Experience in Requirements Reuse A Survey in Malaysia. *International Journal of Software Engineering and Technology, 1*(2).

Basha, N., & Moiz, S. A. (2012). *Component based software development: A state of art.* Paper presented at the Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on.

Benitti, F. B. V., & Silva, R. C. d. (2013). Evaluation of a Systematic Approach to Requirements Reuse. *Journal of Universal Computer Science, 19*(2).

Berger, T., Rublack, R., Nair, D., Atlee, J. M., Becker, M., Czarnecki, K., & Wąsowski, A. (2013). *A survey of variability modeling in industrial practice.* Paper presented at the Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems.

Cavalcanti, Y. C., do Carmo Machado, I., da Mota, P. A., Neto, S., Lobato, L. L., de Almeida, E. S., & de Lemos Meira, S. R. (2011). *Towards metamodel support for variability and traceability in software product lines.* Paper presented at the Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems.

Cerón, R., Dueñas, J. C., Serrano, E., & Capilla, R. (2005). *A meta-model for requirements engineering in system family context for software process improvement using CMMI.* Paper presented at the International Conference on Product Focused Software Process Improvement.

Champeau, J., & Rochefort, E. (2003). *Model engineering and traceability.* Paper presented at the Workshop SIVOES-MDA, UML'03.

Chernak, Y. (2012). *Requirements Reuse: The State of the Practice.* Paper presented at the 2012 IEEE International Conference on Software Science, Technology and Engineering.

Chung, L., Paech, B., Zhao, L., Liu, L., & Supakkul, S. (2012). *Repa requirements pattern template.* Paper presented at the International Workshop on Requirements Patterns (RePa '12).

Czarnecki, K., Grünbacher, P., Rabiser, R., Schmid, K., & Wąsowski, A. (2012). *Cool features and tough decisions: a comparison of variability modeling approaches.* Paper presented at the Proceedings of the sixth international workshop on variability modeling of software-intensive systems.

Easterbrook, S., Singer, J., Storey, M.-A., & Damian, D. (2008). Selecting empirical methods for software engineering research *Guide to advanced empirical software engineering* (pp. 285-311): Springer.

Franch, X., Palomares, C., Quer, C., Renault, S., & De Lazzer, F. (2010). A metamodel for software requirement patterns. *Requirements Engineering: Foundation for Software Quality* (pp. 85-90): Springer.

Gabriel, R. P. (1996). *Patterns of software Tales from the Software Community*. 198 Madison Avenue, New York, New York, 10016-4314: Published by Oxford University Press, Inc.,.

Goknil, A., Kurtev, I., & Millo, J.-V. (2013). *A metamodeling approach for reasoning on multiple requirements models.* Paper presented at the Enterprise Distributed Object Computing Conference (EDOC), 2013 17th IEEE International.

Goknil, A., Kurtev, I., & van den Berg, K. (2008). *A metamodeling approach for reasoning about requirements.* Paper presented at the European Conference on Model Driven Architecture-Foundations and Applications.

Goldin, L., & Berry, D. M. (2013). Reuse of requirements reduced time to market at one industrial shop: a case study. *Springer-Verlag*.

Goldin, L., Matalon-Beck, M., & Lapid-Maoz, J. (2010). *Reuse of Requirements Reduces Time to Market.* Paper presented at the 2010 IEEE International Conference on Software Science, Technology & Engineering.

Gomaa, H., & Shin, M. E. (2002). *Multiple-view meta-modeling of software product lines.* Paper presented at the Engineering of Complex Computer Systems, 2002. Proceedings. Eighth IEEE International Conference on.

Hauksdóttir, D., Mortensen, N. H., & Nielsen, P. E. (2013). Identification of a reusable requirements structure for embedded products in a dynamic market environment. *Computers in Industry, 64*(4), 351-362. doi:10.1016/j.compind.2012.10.008

Hoffmann, M., Kühn, N., Weber, M., & Bittner, M. (2004). *Requirements for requirements management tools.* Paper presented at the 12th IEEE International Requirements Engineering Conference, 2004. .

Konrad, S., & Cheng, B. H. (2002). *Requirements patterns for embedded systems.* Paper presented at the Proceedings of IEEE Joint International Conference on Requirements Engineering, 2002. .

Liang, P., Avgeriou, P., & Wang, C. (2011). Managing Requirements Knowledge using Architectural Knowledge Management Approaches.

Liew, S. C., Liew, S.W., Zain, J.M. (2010). Reversible medical image watermarking for tamper detection and recovery with Run Length Encoding compression. *World Academy of Science, Engineering and Technology*, 72, 799-803.

López, O., Laguna, M. A., & Peñalvo, F. J. G. (2002a). *Metamodeling for Requirements Reuse.* Paper presented at the WER.

López, O., Laguna, M. A., & Peñalvo, F. J. G. (2002b). *Metamodeling for Requirements Reuse.* Paper presented at the Proceedings of 5th. International Workshop on Requirements Engineering (WER'02) November, 11-12, 2002., Valencia, Spain.

Mahendra, P., & Ghazarian, A. (2014). Patterns in the Requirements Engineering: A Survey and Analysis Study. *WSEAS Transaction on Information Science and Application, 11*.

Monzon, A. (2008). *A practical approach to requirements reuse in product families of on-board systems.* Paper presented at the 16th IEEE International Requirements Engineering, RE'08.

Moon, M., Chae, H. S., Nam, T., & Yeom, K. (2007). *A metamodeling approach to tracing variability between requirements and architecture in software product lines.* Paper presented at the Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on.

Moros, B., Toval, A., & Vicente Chicote, C. (2008). *Metamodeling variability to enable requirements reuse.* Paper presented at the Exploring Modellling Language for System Analysis and Design, EMMSAD 2008.

Naish, J., & Zhao, L. (2011). *Towards a Generalised Framework for Classifying and Retrieving Requirements Patterns*. Paper presented at the IEEE First International Workshop onRequirements Patterns (RePa), 2011., Trento.

Navarro, E., Letelier, P., Mocholi, J. A., & Ramos, I. (2006). A metamodeling approach for requirements specification. *Journal of Computer Information Systems, 46*(5), 67-77.

Nerurkar, N. W., Kumar, A., & Shrivastava, P. (2010). Assessment of reusability in aspect-oriented systems using fuzzy logic. *ACM SIGSOFT Software Engineering Notes, 35*(5), 1. doi:10.1145/1838687.1838706

Paige, R. F., Olsen, G. K., Kolovos, D. S., Zschaler, S., & Power, C. (2008). Building model-driven engineering traceability classifications.

Palomares Bonache, C., Quer Bosor, M. C., Franch Gutiérrez, J., Guerlain, C., & Renault, S. (2012). A catalogue of non-technical requirement patterns.

Palomares, C., Quer, C., Franch, X., Renault, S., & Guerlain, C. (2013). *A catalogue of functional software requirement patterns for the domain of content management systems.* Paper presented at the Proceedings of the 28th annual acm symposium on applied computing.

Royer, J.-C., & Arboleda, H. (2013). *Model-Driven and Software Product Line Engineering*: John Wiley & Sons.

Sandhu, P. S., Aashima, Kakkar, P., & Sharma, S. (2010). *A Survey on Software Reusability*. Paper presented at the IEEE International Conference on Mechanical and Electrical Technology (ICMET 2010).

Seman, A., Abu Bakar, Z., Mohd. Sapawi, A. (2010). Centre-Based Hard Clustering Algorithms For Y-Str Data, *Malaysian Journal of Computing*, 1, 62-73.

Sinnema, M., & Deelstra, S. (2007). Classifying variability modeling techniques. *Information and Software Technology, 49*(7), 717-739.

Sinnema, M., Deelstra, S., Nijhuis, J., & Bosch, J. (2004). Covamof: A framework for modeling variability in software product families *Software product lines* (pp. 197-213): Springer.

Sjoberg, D. I., Dyba, T., & Jorgensen, M. (2007). *The future of empirical methods in software engineering research.* Paper presented at the 2007 Future of Software Engineering.

Slavin, R., Shen, H., & Niu, J. (2012). *Characterizations and boundaries of security requirements patterns.* Paper presented at the Requirements Patterns (RePa), 2012 IEEE Second International Workshop on.

Srivastava, S. (2013). A Repository of Software Requirement Patterns for Online Examination System. *International Journal of Computer Science Issues (IJCSI '13), 3*.

Van Gigch, J. P. (2013). *System design modeling and metamodeling*. US: Springer Science & Business Media.

Velasco, J. L., Valencia-García, R., Fernández-Breis, J. T., & Toval, A. (2009). Modelling reusable security requirements based on an ontology framework. *Journal of Research and Practice in Information Technology, 41*(2), 119.

von der Maßen, T., & Lichter, H. (2002). *Modeling variability by UML use case diagrams.* Paper presented at the Proceedings of the International Workshop on Requirements Engineering for product lines.

Wiegers, K. (2005). *More about software requirements: thorny issues and practical advice*. Redmond, Washington: Microsoft Press.

Wiegers, K., & Beatty, J. (2013). *Software Requirements* (3rd ed.). Redmond, Washington: Microsoft Press.

Winkler, S., & Pilgrim, J. (2010). A survey of traceability in requirements engineering and model-driven development. *Software and Systems Modeling (SoSyM), 9*(4), 529-565.

Withall, S. (2007). *Software requirement patterns*: Pearson Education.

Ya'u, B. I., Nordin, A., & Salleh, N. (2016a). *Investigation of Requirements Reuse (RR) Challenges and Existing RR Approaches*. Paper presented at the Advanced Research in Engineering and Information Technology International Conference, Bandong, Indonesia.

Ya'u, B. I., Nordin, A., & Salleh, N. (2016b). *Software requirements patterns and meta model: A Strategy for enhancing Requirements Reuse (RR).* Paper presented at the Information and Communication Technology for The Muslim World (ICT4M), 2016 6th International Conference on.

Ya'u, B. I. (2015). Component-Based: The Right Candidate for Restructuring the Nature of Software Development in Organizations. *International Journal of Engineering and Computer Science, 4*(8), 8.

Zhang, Z., Nummenmaa, J., Guo, J., Mai, J., & Wang, Y. (2011). Patterns for Activities on Formalization Based Requirements Reuse. *Knowledge Engineering and Management, AISC Springer-Verlag, 123*, 695-707.

Zong-yong, L., Zhi-xue, W., Ying-ying, Y., Yue, W., & Ying, L. (2007). *Towards a multiple ontology framework for requirements elicitation and reuse.* Paper presented at the IEEE 31st Annual International Conference on Computer Software and Applications , COMPSAC 2007. .