Published in IET Software
Received on 2nd March 2010
Revised on 3rd June 2010
doi: 10.1049/iet-sen.2010.0020

ISSN 1751-8806

Modelling software process variability: an empirical study

T. Martínez-Ruiz[1] *F. García*[1] *M. Piattini*[1] *J. Münch*[2]

[1]*Alarcos Research Group, Department of Information Technologies and Systems, Escuela Superior de Informática, University of Castilla-La Mancha, Paseo de la Universidad, 4, 13071 Ciudad Real, Spain*
[2]*Fraunhofer Institute for Experimental Software Engineering (IESE), Fraunhofer Platz 1, 67663 Kaiserslautern, Germany*
E-mail: tomas.martinez@uclm.es

Abstract: Variability in software process models justifies tailoring them to meet the specific goals and characteristics of organisations and projects. Existing process modelling notations typically do not have constructs which are appropriate for expressing process variability. To fill this gap, the authors have extended software process engineering metamodel (SPEM) to vSPEM, by adding new variability constructs (such as variants and variation points). This article presents an empirical validation to check whether the variability constructs supported through vSPEM are more appropriate for modelling variant-rich processes than SPEM, in terms of understandability of the notation, as well as of their variability mechanisms. The results indicate that the vSPEM variability mechanisms understandability is a 126.99% higher than for SPEM. On the other hand, process diagram understandability is a 34.87% lower with vSPEM than with SPEM. If we compare the relative results obtained with regard to understandability of diagrams and understandability of variation mechanisms of both vSPEM and SPEM, the enhancement just mentioned is 3.64 times that fall. The results indicate accepting that a slight decrease in understandability of the diagrams might lead to a large increase in understandability when using variability mechanisms of vSPEM.

1 Introduction

The organisations that are dedicated to software development are conscious of the need to deploy well-defined processes, with the objective of not only raising their maturity level, but also of improving the manner in which their products are developed, and thus their quality [1]. However, software processes are complex entities, in that they are related both to the organisation's workforce and to its policies [1, 2] and this implies that the implantation and the improvement of a software process is not a trivial task.

First of all, software process models have been suggested as a way to describe activities in software development and involved artefacts and roles [3]. In addition, process reference models have been developed [4] with the objective of offering organisations basic guidelines towards process implementation. These models include best practices extracted from industry and refined by experts to offer organisations the support necessary to create their own processes. The literature therefore contains well-known and widely implemented reference models, such as capability maturity model (CMMI) [5] or ISO 12207 [6], and assessment models which improve process capacity such as standard CMMI appraisal method for process improvement (SCAMPI) [7], or ISO 15504 [8, 9].

However, reference models are defined in a highly generic manner in order to enable them to be applied to any organisation or project. In the long term, this implies that, in practice, it is extremely difficult to use them in a specific project without adapting them beforehand [10], since 'Just as there are no two identical projects, there are no two identical processes in the World' [11]. Owing to the fact that software development projects are very much context-dependent and comprise a lot of human-based and creative activities, they need to be adapted to specific project goals and environment characteristics. This implies that if the processes are not to be 'abandoned' and are to be correctly executed, they must be incorporated into the manner in which work is carried out within the organisation [12], which is known as 'software process institutionalisation'. According to Chrissis *et al.* [12], in order to attain institutionalised processes, it is necessary for these processes to be adapted from the organisation's own set of standard processes.

As a result of process generation, another process that is almost identical to the original is generated, but with slight modifications that make it unique and specific to the project and to the organisation. The original process, together with those that have been adapted from it, make up what is known as a process family [13, 14] in a similar way to programme families [15]. In a process family, processes can make good use of their similarities and exploit their differences, in a manner much like that of software product lines [14, 16].

To support the variant-rich processes approach, and thus facilitate the adaptation of processes, their models must be

able to represent possible variants appropriately. To do this, it is necessary for the process modelling languages, and therefore the metamodels that support them, to include variability constructors. Software process engineering metamodel (SPEM) [17], therefore, includes specific mechanisms with which to define the method's variability, although not directly in the process, an important aspect to take into consideration if the adaptation of said processes to the nature of each project is to be made possible. To this end, we have defined an extension in which we have included specific variability mechanisms for variant-rich process modelling, known as vSPEM [18, 19]. This extension implies the application of the product line philosophy [20] in order to manage the variability in process families. The variability included in vSPEM therefore complements SPEM v.2.0 since, (i) it supports the process element variation capacity (work product use, task use etc.) which, in accordance with SPEM v.2.0, cannot vary, and (ii) it permits the generically defined methods of the adaptations that are necessary in the process to be uncoupled, thus allowing them to fit in with the project's characteristics and the organisational culture.

An empirical validation has been conducted to determine which process variability mechanism implementation (through variation points or modifications to the definition of the method) could best support the adaptation of software processes to the context of a project and a specific organisation. It is also easier to apply and understand. The intention of this experiment is to obtain first evidence about whether it is more effective to carry out process variations by using the variation point and variant focus, included in the vSPEM extension, or to carry out changes and modifications to the structure of the method by applying the SPEM variation mechanisms.

Bearing in mind the issues identified previously, this paper describes the proposed enhanced variability mechanisms and the results of the study conducted. The remainder of this paper is organised as follows. Section 2 shows those works related to initiatives for process tailoring and variability, in which particular attention is paid to the description of the SPEM variability mechanisms. In Section 3 the enhanced vSPEM for tailoring processes is explained. Section 4 describes the empirical study conducted, and includes the experimental settings, data analysis, their interpretation and a discussion of experimental validity. Finally, conclusions and future work are outlined in Section 5.

2 Related work

The literature related to this topic contains works on variability in software processes and variant-rich processes, as well as those focusing on process lineation in accordance with project characteristics. The concept of variability in processes, along with those elements that are necessary for relating generic and specific process descriptions are presented in [21]. Rombach [14] proposes the application of the product line paradigm to software processes, giving rise to management of variant-rich processes. Similarly, in this work they identify two approaches for the design of variant-rich processes: ascending and descending.

According to Puhlmann *et al.* [22] processes can be personalised by using variants and applying variation mechanisms to product lines. Schneiders and Weske [23] propose the use of an unified modelling language (UML) activity diagram to represent variant-rich processes architecture, and stereotypes are used to identify variation points and variations. According to Simidchieva *et al.* [24], a set of related processes can be managed as a process line in such a way that they are variations of the same metaprocess. Armbrust *et al.* [25, 26] propose applying the method used in product lines, based on (i) determining the process line's scope, (ii) modelling it and (iii) defining the architecture.

On the other hand, adapting a process may, on occasions, require diverse variations which affect diverse points of its structure transversally. For these adaptations to be carried out, Ma *et al.* [27] propose the application of aspect-oriented software development (AOSD) paradigm concepts [28].

Of those initiatives that apply variability to software processes in order to align the processes themselves with the projects found, we should highlight, on the one hand, Martins and Silva [29, 30], an initiative based on three fundamental steps: (i) defining the process, (ii) adapting and monitoring the process execution and (iii) measuring the process. Process and project alignment methodology (ProPAM) similarly includes a metric with which to estimate how the processes are adjusted to the projects' needs. Killisperger *et al.* [31] propose an environment through which to apply variations automatically in processes through variation operations. We should stress that the authors differentiate between primary variations and variations during execution. Silva Barreto *et al.* [32] propose another environment in which to carry out variations in software processes. This is based on the definition of variations of those elements of which the process is composed, and on their later integration.

On the other hand, Martínez-Ruiz *et al.* [33] propose SPrInT, an environment for software process institutionalisation based on a four-phase iterative cycle structured from variant-rich processes. This environment supposes an initiative with which to obtain processes that are adapted according to the characteristics of the project or the organisation, and the use of these adapted processes through an iterative application of the descending and ascending approaches proposed by Rombach. Therefore after each iteration, not only is software process institutionalisation achieved in an independent manner, but the variant-rich processes are also slightly more institutionalised, thus permitting the processes generated to be incorporated into the organisation's characteristics better. SPrInT uses vSPEM as a software process variability support [18, 19].

Given that the proposal presented in this work is based on SPEM, the following subsection summarises its main characteristics and describes how SPEM manages process variability.

2.1 SPEM and process variability

The SPEM [17] is the object management group (OMG) proposal for the interrelated modelling of processes and methods. The current version of SPEM is based on the definition of methods that are used to create reusable libraries. These are later reused to define software processes. By using both methods and processes, specific project plans are created [17]. The structure of SPEM includes seven packages whose functionality can be specifically adapted to different uses. These packages include elements with which to define processes and methods and their main constructors are tasks, work products, roles and guidance.

SPEM 2.0 includes method variability mechanisms that support the customisation of method definition and avoid their being directly modified. These variability mechanisms are supported by means of the definition of an abstract class and an enumeration in the plug-in package method shown in Fig. 1. The abstract class Variability Element represents 'an element that may vary', and this capability of variation is added to the elements that inherit from it. These elements are the Activity (from ProcessStructure package), MethodContentElement (from MethodContent package) and Section (from ManagedContent package). Given that the Method Content Element is also an abstract class, and that all the elements used to define the method are inherited from it, they also acquire the capability to vary (Fig. 1). The Variability Element also defines a reflexive relationship that specifies which variation element varies with other variation element. The second element is the enumeration variability type. This defines the type of variability that is applied to the relationship previously described, and may be of five types, 'na', 'contributes', 'replaces', 'extends', and 'extends-replaces'. The usage rules can be found in [17].

The variations carried out on the definition of the methods through the variability mechanisms included in SPEM 2.0 are then applied to specific processes derived from the method. This is illustrated with an example. Fig. 2 shows part of the method extracted from a development methodology, which includes two tasks (design and implementation), with their corresponding roles and related products. The process that is created from the method is shown in Fig. 3, and is modelled with the SPEM 2.0 notation.

As can be observed in Figs. 2 and 3, the SPEM variation mechanisms make it possible to carry out variations to the method. In particular, in Fig. 4 a variant to the specific original model has been modelled for projects realised with the C# programming language. Similarly, Fig. 5 shows the process that has been structured from the method with the variations from Fig. 4.

To continue with the example shown in Figs. 2 and 3, the SPEM notation can be used to carry out modifications to the definition of those methods that will also affect the process. In order to adapt the process in a project that has been developed using the C# language, both the implementation task and the products generated must be adapted by applying the characteristics of the programming language (Fig. 5). As

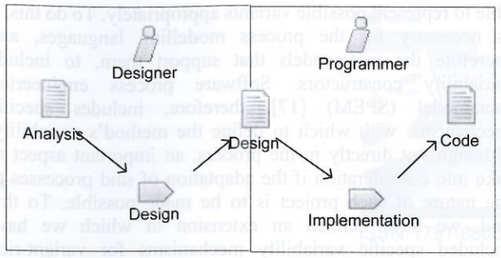

Fig. 2 *Part of method modelled using the SPEM notation*

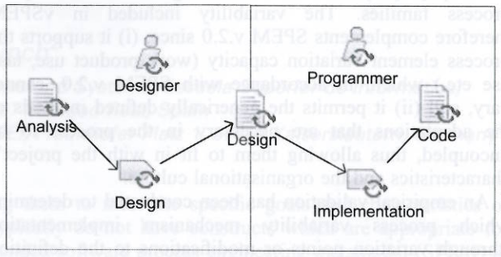

Fig. 3 *Process corresponding to the method of Fig. 2*

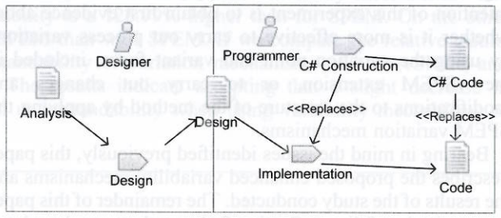

Fig. 4 *Application of the SPEM variability mechanisms on the method definition of Fig. 2*

will be noted, if only the SPEM variation mechanisms are available, then the only way to carry out these adaptations is by doing so on the definition of the method by using

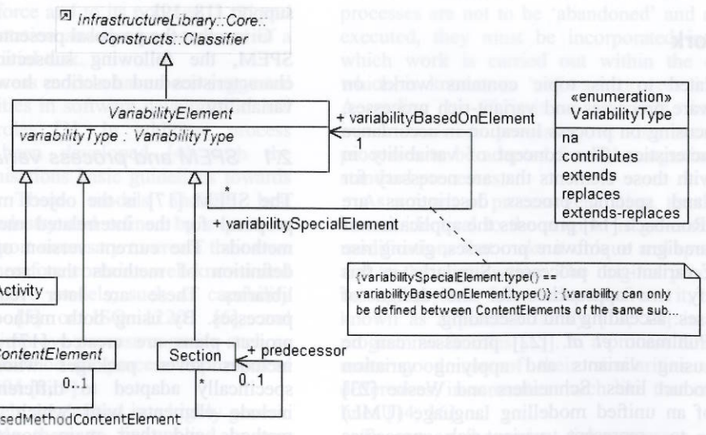

Fig. 1 *SPEM variability mechanisms [17]*

IET Softw., 2011, Vol. 5, Iss. 2, pp. 172–187
doi: 10.1049/iet-sen.2010.0020

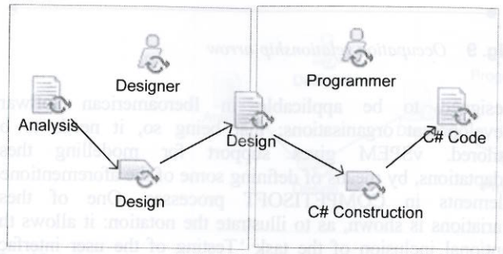

Fig. 5 *Process of Fig. 3 after applying the variability mechanism in its method definition, as shown in Fig. 4*

contributes and replaces relations (Fig. 4), which are applied to the process (Fig. 5).

The SPEM variation mechanisms are not, therefore, the most suitable mechanisms with which to model variability, in that they substantiate the variant-rich processes. That is for the following reasons:

- The variability mechanisms of contributes and replaces are applicable for 'modifying' any element in a method. In that sense, SPEM do not define any limitation, in order to ensure that the aims of the method, and later the process, are not modified. These mechanisms are focused on facilitating reuse, but they are not useful without tailoring.
- The variations that are needed to adapt processes according to the characteristics of a given project do not ensure modifications in the work that is being carried out (the method), but rather in how this is performed (the process).
- Carrying out variations in the definition of the method each time a process needs to be adapted to a new project is no trivial task. Moreover, it involves the redefinition of the methods that need maintenance and will be difficult to reuse.
- Additionally, SPEM does not include a specific notation with which to represent process variability, which is represented by using the UML association and inheritance relationships, characterised by means of stereotypes. However, when real processes are modelled and process diagrams are bigger, the fact that variability relationships are similar to association ones means they may be confused with each other.

Having taken these limitations of SPEM into consideration, an extension to SPEM was proposed to complement it with mechanisms that will allow the variability of software processes to be handled in a suitable manner. This extension proposal is summarised in the following section.

3 vSPEM language

If we are to provide SPEM with the capability of modelling and managing variant-rich processes, its metamodel must be enriched with new elements focused on software process variability. In order to achieve this, a new package, called ProcessLineComponents, has been added to the SPEM metamodel (see Fig. 6).

The new process variability mechanisms included in this new package are based on variation points and variants by following the product lines paradigm. Variation points are the places at which variations occur, and the elements may be different from one process to another. Variants are specific implementations of this variability, and each one of these variants makes the process unique.

From a more abstract to a more concrete level, the first element created is the ProcLElement. This element is an abstraction of all the elements related to variations in software process elements and inherits from a UML classifier. Two specifications of ProcLElement are VarPoint and Variant, which are used to represent variation points and variants, respectively. However, many elements are used to build processes in SPEM, such as activity, taskUse, RoleUse, WorkProductUse and ToolUse. Variant and VarPoint are therefore abstract classes. Concrete variants and variation points inherit from the SPEM element which provides the capability to vary, and from the Variant or VarPoint classes, in order to be considered as a variant or variation point. Fig. 7 shows an example of how the variants and variation points of an activity (VActivity and VPActivity) are created. A detailed explanation of these variability mechanisms can be found in [18, 19].

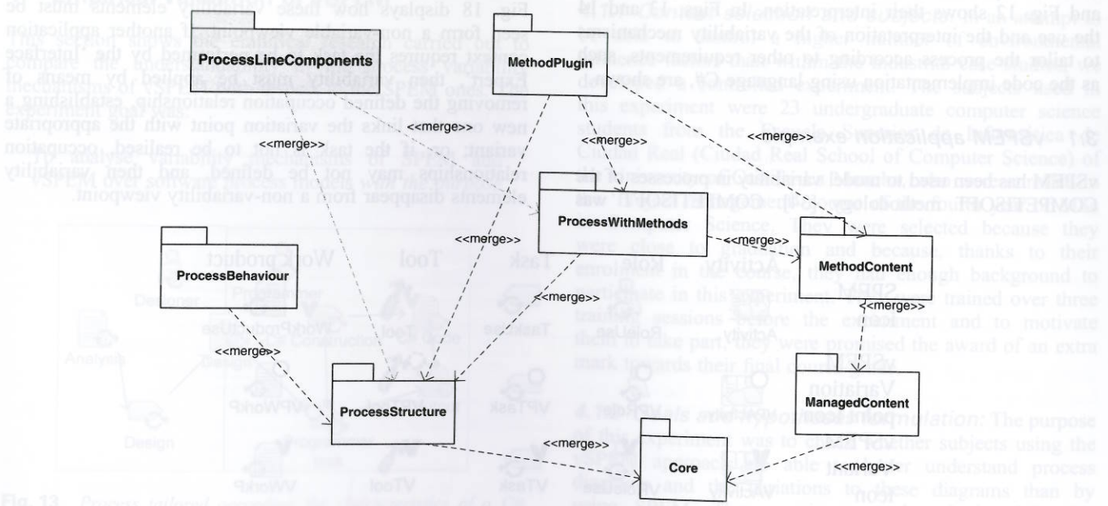

Fig. 6 *SPEM variant-rich process package*

IET Softw., 2011, Vol. 5, Iss. 2, pp. 172–187
doi: 10.1049/iet-sen.2010.0020

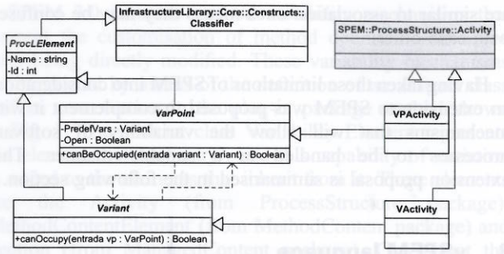

Fig. 7 *Abstract variability elements*

To make variant-rich process diagrams more understandable and to make it easier to distinguish them from process diagrams without process variability, we have defined some new icons based on the original SPEM icons, as is shown in Fig. 8.

The aspects mentioned above tell us that specific processes are created when variants occupy variation points. A variation point that is occupied by a variant is considered to be a concrete SPEM element. Neither empty variation points nor variants that do not occupy any of them are considered as part of the process. To display how the variants occupy variation points, the Occupation relationship is created. Furthermore, to differentiate this relationship from others, it is drawn using an arrow with a filled circle at the rear tip (Fig. 9).

This notation is illustrated in the example shown in Figs. 2 and 3 (Section 2.1). The left-hand side of Fig. 10 shows the core process of the variant-rich processes (modelled with the corresponding notation), which corresponds with the method shown in Fig. 2, whereas the right-hand side shows the variants which are available to personalise processes.

The variant-rich process shown in Fig. 10 can be used to tailor processes according to project and organisation needs. Then personalised processes are created by simply occupying the variation points with the necessary variants. Figs. 11–14 show the use of the variability mechanism to tailor the core process. First of all, Fig. 11 shows the use of variability mechanism to tailor the core process according to some requirements, the implementation in any language, and Fig. 12 shows their interpretation. In Figs. 13 and 14 the use and the interpretation of the variability mechanisms to tailor the process according to other requirements, such as the code implementation using language C#, are shown.

3.1 vSPEM application example

vSPEM has been used to model variability in processes of the COMPETISOFT methodology [34]. COMPETISOFT was

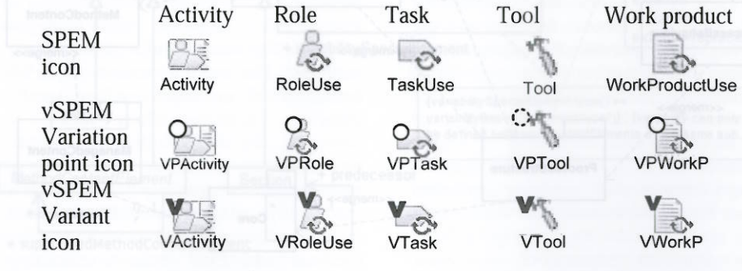

Fig. 8 *Concrete variants and variation point graphic icons*

IET Softw., 2011, Vol. 5, Iss. 2, pp. 172–187
doi: 10.1049/iet-sen.2010.0020

Fig. 9 *Occupation relationship arrow*

designed to be applicable in Iberoamerican software development organisations; that being so, it needs to be tailored. vSPEM gives support for modelling these adaptations, by means of defining some of the aforementioned elements in COMPETISOFT processes. One of these variations is shown, as to illustrate the notation: it allows the optional inclusion of the task 'Testing of the user interface prototype' within the construction activity of the software development process. This task focuses on identifying critical defects, in conjunction with the user or other interface expert. Variability exists as regards the role, therefore and all this variability may be modelled by using the vSPEM notation.

First, it is necessary to define two variation points related to construction activity; the first one is the task variation point (VPTask) that makes the inclusion of the new task possible, and it must be placed at the point where variability occurs, vptask1 – that is where the aforementioned task is to be placed as a variant. A VPRole, called vprole1 also has to be defined. This is a role variation point to allow variability about roles in the construction activity, and in the new task. Fig. 15 shows a diagram of the construction activity that includes both variation points. Moreover, it is also necessary to define those variants that are to be placed in the variation points mentioned earlier. As explained above, to support this variability, at least three variants are necessary. These are the variants for the 'Testing of the user interface prototype' task (VTask), and another two role variants (VRole) for representing the 'User', and the 'Interface Expert'. These variation points and variants have their own properties, but lack of space obliges us to leave them out. Fig. 16 illustrates these.

Finally, to tailor the software process, it is only necessary to decide the variability around the variation points, by means of defining the occupation relationships that link each variation point with the appropriate variant. Fig. 17 shows the occupation relationships that make the 'Testing of the User Interface Prototype' possible, and another occupation relationship that links the role variation point with the corresponding role variant – User or Interface Expert. Fig. 18 displays how these variability elements must be seen form a non-variable viewpoint. If another application context requires the task to be performed by the 'Interface Expert', then variability must be applied by means of removing the defined occupation relationship, establishing a new one that links the variation point with the appropriate variant; or, if the task is not to be realised, occupation relationships may not be defined, and then variability elements disappear from a non-variability viewpoint.

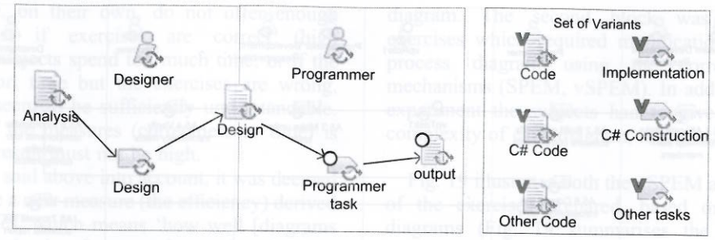

Fig. 10 *Example of a variant-rich process which includes a core process (left) and a set of variants (right)*

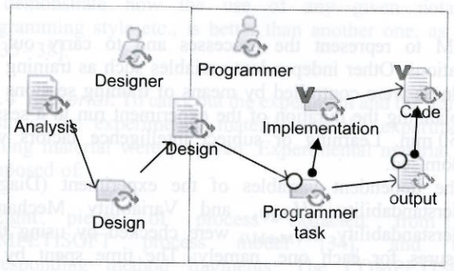

Fig. 11 *Process tailored according to the characteristics of one development project*

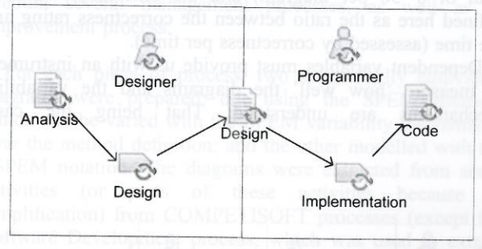

Fig. 12 *Interpretation of the variability mechanisms of process of Fig. 11*

4 Empirical validation of vSPEM

This section shows the empirical research carried out to compare the understandability of the process variability mechanisms of vSPEM with respect to the SPEM ones. The experiment goal was:

To analyse variability mechanisms of SPEM and vSPEM over software process models *with the purpose*

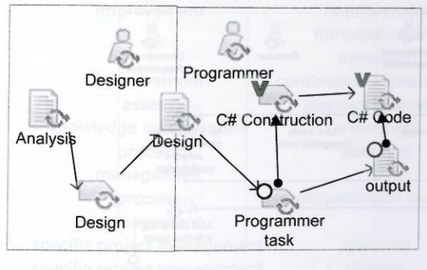

Fig. 13 *Process tailored according the characteristics of a C# specific development project*

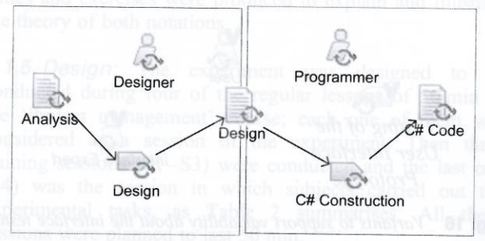

Fig. 14 *Interpretation of the tailored process of Fig. 13*

of comparing them, *with regard to their* notation and variability mechanisms understandability, *from the point of view of* process engineers *within the context of* undergraduate students of software engineering.

The result we have obtained shows vSPEM notation is quite a lot more difficult to understand than SPEM because of the fact that it adds more constructors to the standard, but after analysing the results of the variations realised, it could be shown that vSPEM variability mechanisms are much more understandable when varying processes. Statistical analysis shows that these results are statistically significant.

4.1 Experiment planning

4.1.1 Context selection and subjects: In an attempt to be able to control a higher number of environmental influence factors than within the industrial case studies, we developed a controlled experiment. The subjects used in this experiment were 23 undergraduate computer science students from the Escuela Superior de Informática de Ciudad Real (Ciudad Real School of Computer Science) of the University of Castilla-La Mancha, who were enrolled in the 'Project management' course of the fourth year of MSc in Computer Science. They were selected because they were close to graduation and because, thanks to their enrolment in the course, they had enough background to participate in this experiment. They were trained over three training sessions before the experiment and to motivate them to take part, they were promised the award of an extra mark towards their final course grade.

4.1.2 Goals and hypotheses formulation: The purpose of this experiment was to check whether subjects using the vSPEM approach are able to better understand process diagrams and the variations to these diagrams than by using SPEM. The experiment evaluated the following hypotheses:

IET Softw., 2011, Vol. 5, Iss. 2, pp. 172–187
doi: 10.1049/iet-sen.2010.0020

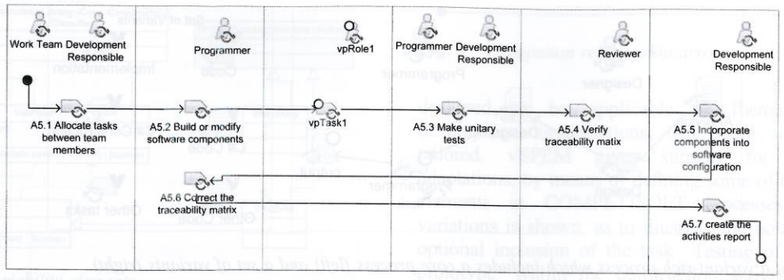

Fig. 15 *Core process of the COMPETISOFT process model with two variation points*

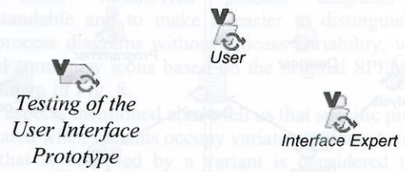

Fig. 16 *Variants to support variability about the interface testing*

- H_{DigU}: The vSPEM process model diagrams are significantly more understandable than SPEM process model diagrams.
- H_{VMechU}: The process variability mechanisms of vSPEM are significantly more understandable than SPEM ones.

4.1.3 Selection of variables: The main independent variable (i.e., factor) was the modelling notation used. Both treatments of this factor were by using either vSPEM or SPEM to represent the processes and to carry out the variations. Other independent variables such as training and tiredness were controlled by means of training sessions and by planning the duration of the experiment run in a session of 50 min. Learning or subject intelligence factors were randomised.

The dependent variables of the experiment (Diagram Understandability, H_{DigU}, and Variability Mechanism Understandability, H_{VMechU}) were checked by using three measures for each one, namely: The time spent by the participants on answering three questions or conducting three modifications (assessed in seconds), the correctness of the answers (rated from 0 to 10 for each exercise, making a total of 0–30 per diagram) and the efficiency, which is defined here as the ratio between the correctness rating and the time (assessed by correctness per time).

Dependent variables must provide us with an instrument to measure 'how well' the diagrams and the variability mechanisms are understood. That being the case,

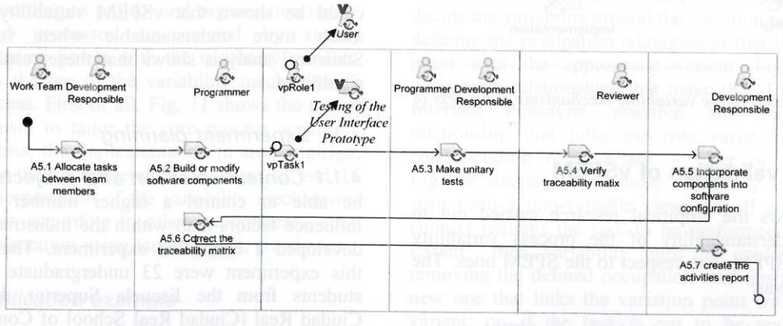

Fig. 17 *Core process with some variants occupying the variation points*

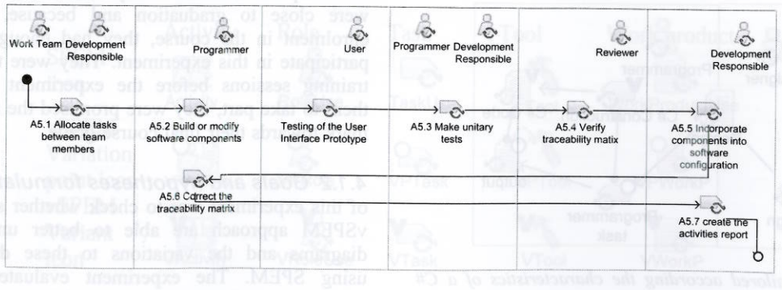

Fig. 18 *Interpretation of the realised variation from a non-variability viewpoint*

IET Softw., 2011, Vol. 5, Iss. 2, pp. 172–187
doi: 10.1049/iet-sen.2010.0020

correctness and time, on their own, do not offer enough information, that is, if exercises are correct (high correctness) but the subjects spend too much time, or if the subjects spend a short time but the exercises are wrong, mechanisms do not seem to be sufficiently understandable. In each case, one of the measures (correctness or time) is high, but the overall result must not be high.

Taking all we have said above into account, it was deemed necessary to introduce a new measure (the efficiency) derived from the previous ones, which means 'how well [diagrams and/or variability mechanisms] are understood, related to time'. Moreover, efficiency – as the rate between correctness and time – is widely used in empirical studies to demonstrate how the use of any given notation, programming style etc., is better than another one, as seen in [35–38].

4.1.4 Material: To carry out the experiment and the training sessions, the experimental material and the experimental training material were prepared. Experimental material was composed of:

- Eight pieces of process extracted from the COMPETISOFT process model [34], and their corresponding method fragments. The COMPETISOFT process reference model [38] is composed of ten processes, covering from the business management process to the development and maintenance processes, and taking into account several management processes, as well as an improvement process.

For each piece of process, two semantically equivalent diagrams were prepared: one using the SPEM notation, which can be varied with the SPEM variability mechanisms over the method definition; and the other modelled with the vSPEM notation. The diagrams were extracted from some activities (or parts of these activities because of simplification) from COMPETISOFT processes (except the Software Development process, which was used to extract the examples used during the training sessions). Parts of process were randomly selected from the process models. The size and complexity were therefore different in all the diagrams. Table 1 details the process and the activity from which the diagrams in the exercises were extracted.

- Two blocks of questions were designed for each piece of the process. The first one included questions about the diagram. The second block was composed of three exercises which required modifications to be made to the process diagram using the corresponding variability mechanisms (SPEM, vSPEM). In addition, at the end of the experiment the subjects had to give their opinion on the complexity of each diagram, rating it accordingly.

Table 1 Process and activities used for extracting the diagrams in the exercises

Exercise	Process	Activity
1	improvement	improvement iteration
2	improvement	improvement formulation and execution
3	improvement	continuous improvement
4	assess	process assessing
5	knowledge management	executing
6	process management	planning
7	process management	implementation planning
8	specific project management	planning
9	specific project management	closing
10	business management	business planning

Fig. 19 illustrates both the vSPEM and the SPEM versions of the exercises prepared, based on the aforementioned diagrams (Fig. 19 summarises the material of exercise number 8). Three other similar sets of material, extracted from the Software Development process, were also prepared for use during the training sessions (S1–S3), and some slides and exercises were produced to explain and illustrate the theory of both notations.

4.1.5 Design: The experiment was designed to be conducted during four of the regular lessons of 50 min of the 'Project management' course; each one of them was considered as a session of the experiment. Then three training sessions (S1–S3) were conducted and the last one (S4) was the session in which subjects carried out the experimental tasks, as Table 2 summarises. All these sessions were planned to last 50 min.

During the first training session (S1), the SPEM metamodel, along with its variability mechanisms were explained and these explanations were illustrated with several examples. Finally, the students solved some exercises and at the end of the session the instructor went over these exercises.

The second training session (S2) was conducted by using the same structure, but this time explaining the vSPEM extension and solving some exercises of this notation. The first two training sessions were planned to include 30 min spent in theory and 20 min solving exercises. Once they were acquainted with all the explanation of the theory, two sets of exercises similar to the ones in the experiment were given in this session. The first was one that contained the solution, so as to have an example of the experiment material, and the other one was to be resolved in training session three.

The third training session (S3) was used to solve a complete example of the experiment and to give some indications which subjects had to follow to carry out the experimental tasks during session four. The first part was planned to last around 40 min and then 10 min were used to give the experimental instructions.

4.1.6 Task and treatment: A within-subject design was applied for the experiment. Each participant was given eight diagrams, four modelled with SPEM and four with vSPEM. The distribution was performed through randomisation to prevent and minimise learning effects: for each participant, the order of the diagrams and the given version of each diagram (using SPEM or vSPEM notation) were determined by random functions. In all, each participant received the eight exercises in a different order that included four SPEM and four vSPEM randomised diagrams.

For each one of the eight exercises they were given, students had to examine the diagram in the exercise and then answer the three diagram understandability questions about this diagram (writing the time at the beginning and the end). Then they were asked to carry out three modifications using the corresponding variability mechanisms (SPEM or vSPEM). The experimental material and the S4 session was planned to last 50 min at most, but

IET Softw., 2011, Vol. 5, Iss. 2, pp. 172–187
doi: 10.1049/iet-sen.2010.0020

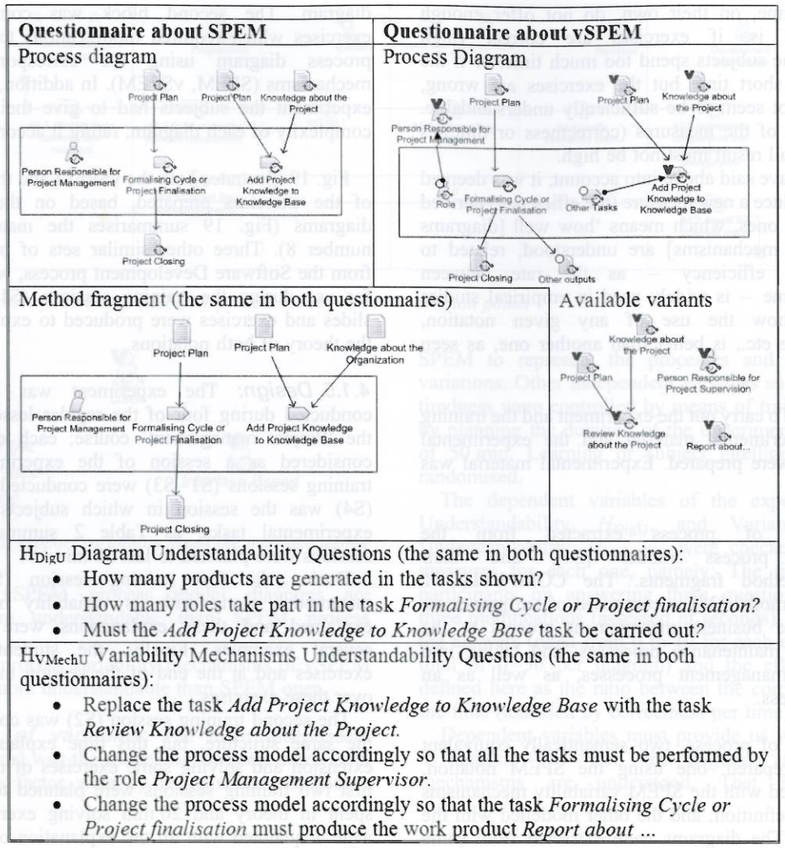

Fig. 19 *Excerpt of experimental material: versions with vSPEM and SPEM notations*

another 10 extra minutes could be used until the following lesson started.

4.2 Execution

In order to guarantee that the material had no mistakes, it was checked several times by authors. It should also be noted that a pilot experiment was run beforehand by a student with a similar background to the subjects, who would be participating in the experiment and with ten models. The results obtained were those that were expected, and no significant changes to the design were necessary. However, in order to ensure that the subjects would spend about an hour (the time of each session) on the experiment, we reduced the number of exercises for each subject from ten to eight.

The three training sessions were executed as was planned; subjects took an active part in sessions by solving exercises. No contingencies occurred. The experiment took place without any problems during the last (fourth) session. A large clock was screened onto the blackboard to make sure all subjects wrote accurate times. In the end, most people finished on time (50 min).

Once the experiment had been carried out, 23 subjects returned their questionnaires. Of these, two were removed, the first because the subject had forgotten to write the time at the beginning and at the end of each questionnaire, and the second because (s)he did not complete most of the given exercises.

4.3 Analysis and interpretation

4.3.1 Descriptive statistics: After the experiment, material was marked by the main author and the whole evaluation process was supervised and checked by all the authors. In order to obtain the measures mentioned above from exercises, time was obtained from the number of seconds subjects used to answer the Diagram or Variability Mechanisms Understandability questions – subjects wrote the initial and the final time; the measure was obtained by means of the difference between these. The correctness of exercises was evaluated according to the 0–10 scale, in order to make it possible to appreciate different values of correctness and the completeness of the questions answered, hence the correctness of each block of three questions was rated from 0 to 30 points. After evaluating Diagram Understandability exercises, these were seen as correct or completely wrong (Exercises answering the questions about what role some given activity realises or what work product a task uses as input only may be entirely correct – if the element is correct – or completely incorrect – if not.), in a binary way. However, in order to use the same scale for assessing the Diagram and the Variability Mechanisms Understandability exercises, and because the correctness

doi: 10.1049/iet-sen.2010.0020

Table 2 Overview of the experiment sessions planning and the material used

	Content	Material used	Time, min
training session S1	SPEM and SPEM variability mechanisms explanation	SPEM slides	30
	SPEM and SPEM variability mechanisms exercises	SPEM exercises and slides	20
training session S2	vSPEM and vSPEM variability mechanisms explanation	vSPEM slides	30
	vSPEM and vSPEM variability mechanisms exercises	vSPEM exercises and slides they were given extra material for session S3	20
training session S3	SPEM and vSPEM exercises (similar to experimental tasks) and doubts solving instructions for the experiment at S4	solved exercises replica of the experiment material	40
		1 slide	10
experimental session S4	experiment tasks	randomised material	50

marks were used to build the extra mark given to subjects as a reward, Diagram Understandability exercises were also assessed on a 0–10 scale, on a binary basis (ten if correct, zero otherwise). Finally, efficiency was obtained by means of applying the formula (efficiency = correctness/time). Therefore from each one of the 21 participants, we obtained eight sets of data about Diagram Understandability and Variability Mechanism Understandability, which makes 168 cases. Statistical analyses have been based on the averages of these variables, to check the hypothesis presented, using the SPSS tool. Table 3 shows an excerpt of the data table we handled.

Figs. 20 and 21 show the box plot of the results obtained from the 21 participants (shown partially in Table 3) of the Diagram and Variability Mechanisms Understandability, respectively. These set out the dispersion, and quartiles of the three variables – correctness, time and efficiency, called DIG_CORR, DIG_TIME and DIG_EFF as regards Diagram

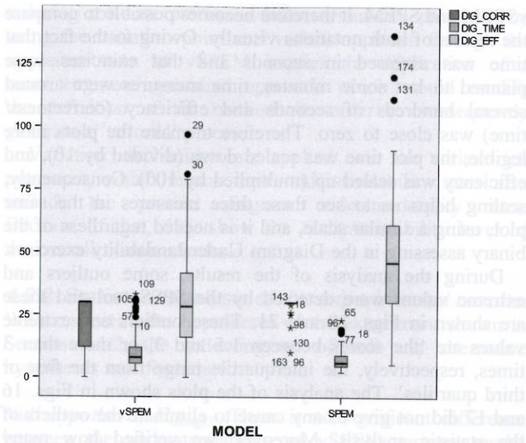

Fig. 20 *Box plot of the data obtained with regard to diagram understandability, with outliers*

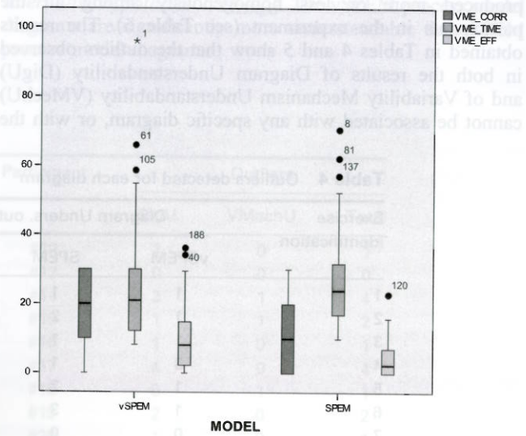

Fig. 21 *Box plot of the data obtained as regards variability mechanism understandability, with outliers*

Understandability (Fig. 20) and VME_CORR, VME_TIME and VME_EFF with respect to Variability Mechanisms Understandability (Fig. 21)- between the two notations –

Table 3 Excerpt of the data set obtained after analysing the experimental material

Subj	Notation	Exercise	Dig_Corr	Dig_time	Dig_Eff	Vme_Corr	Vme_time	Vme_Eff	Comp
1	1	VSPEM_6	10	275	0.03636364	10	586	0.01706485	3
1	1	VSPEM_2	20	125	0.16	28	113	0.24778761	2
1	2	SPEM_8	20	89	0.2247191	26	219	0.11872146	3
1	1	VSPEM_3	30	70	0.42857143	18	222	0.08108108	3
1	1	VSPEM_4	10	331	0.03021148	18	112	0.16071429	4
1	2	SPEM_1	30	91	0.32967033	27	482	0.0560166	4
1	2	SPEM_7	30	94	0.31914894	10	145	0.06896552	3
1	2	SPEM_5	30	71	0.42253521	14	169	0.08284024	4
2	1	VSPEM_3	20	62	0.32258065	22	82	0.26829268	3
2	2	SPEM_1	30	100	0.3	29	301	0.09634551	3
....									

doi: 10.1049/iet-sen.2010.0020

vSPEM and SPEM. It therefore becomes possible to compare the results of both notations visually. Owing to the fact that time was assessed in seconds and that exercises were planned to last some minutes, time measures were around several hundreds of seconds and efficiency (correctness/time) was close to zero. Therefore to make the plots more legible, the plot time was scaled down (divided by 10), and efficiency was scaled up (multiplied by 100). Consequently, scaling helps us to see these three measures in the same plot, using a similar scale, and it is needed regardless of the binary assessing in the Diagram Understandability exercises.

During the analysis of the results, some outliers and extreme values were detected by the SPSS tool and these are shown in Figs. 20 and 21. These outliers and extreme values are 'the scores between 1.5 and 3, or more than 3 times, respectively, the interquartile range from the first or third quartiles'. The analysis of the plots shown in Figs. 16 and 17 did not give us any cause to eliminate the outliers of the statistic analysis. Moreover, we verified how many outliers appeared in each of the exercises according to the treatment (see Table 4), and whether the order in which the exercises were carried out had influenced the outliers (see Table 5). We also verified whether the outliers had been produced more or less homogenously among all the participants in the experiment (see Table 6). The results obtained in Tables 4 and 5 show that the outliers observed in both the results of Diagram Understandability (DigU) and of Variability Mechanism Understandability (VMechU) cannot be associated with any specific diagram, or with the random order in which the subjects answered the diagrams. Table 6 similarly shows that the outliers are generated in a distributed manner for the various participants. We do not therefore consider that there is justification for excluding the outliers from the statistical analysis.

Once we had decided that the outliers must be considered, we first performed a descriptive analysis of the data, detailed in Table 7 for the Diagram Understandability results and in Table 8 for the Variability Mechanism Understandability results.

The results of the experiment show that the time that the participants spent on understanding and answering the questions relating to the diagrams modelled with vSPEM was 27.65 s longer than that spent on diagram modelling using the SPEM notation, which means 38.01% worse. Furthermore, the correctness of answers for vSPEM was 18.49% worse (5.06 negative difference) and efficiency measure was worse by 34.87% (0.17 worse).

In contrast, the variability mechanism understandability when the participants carried out the modifications provided better results when using vSPEM than when using SPEM: the participant effort in carrying out the tasks was slightly superior in the diagrams modelled using SPEM than when using vSPEM, with a positive difference of 23.14 s, which means 38.01% better. Moreover, the correct answers were improved by 67.79% (6.96 positive difference) and the efficiency measure was improved by 0.05304 (from 0.04177 in the case of SPEM to 0.09480 in the case of vSPEM), which is an improvement of 126.99%.

Table 4 Outliers detected for each diagram

Exercise identification	Diagram Unders. outliers			Variability Mech. Undest. outliers		
	vSPEM	SPEM	Total	vSPEM	SPEM	Total
1	1	1	2	0	1	1
2	1	2	3	0	0	0
3	1	1	2	0	0	0
4	3	1	4	1	1	2
5	1	2	3	1	0	1
6	1	3	4	2	0	2
7	0	0	0	0	0	0
8	0	4	4	1	1	2
9 (pilot)	0	0	0	0	0	0
10 (pilot)	0	0	0	0	0	0
total	8	14	22	5	3	8

Table 5 Outliers detected for order in which diagrams were modelled

Exercise order	Diagram Unders. outliers			Variability Mech. Undest. outliers		
	vSPEM	SPEM	Total	vSPEM	SPEM	Total
1st	3	1	4	2	2	4
2nd	2	2	4	0	0	0
3rd	0	4	4	0	0	0
4th	0	0	0	1	0	1
5th	2	1	3	1	0	1
6th	1	4	5	0	0	0
7th	0	1	1	0	0	0
8th	0	1	1	1	1	2
9th (pilot)	0	0	0	0	0	0
10th (pilot)	0	0	0	0	0	0
total	8	14	22	5	3	8

IET Softw., 2011, Vol. 5, Iss. 2, pp. 172–187
doi: 10.1049/iet-sen.2010.0020

The results obtained show that the mechanisms of vSPEM based on variation points and variants make the understandability of diagrams worse, although understandability will inevitably be affected when more constructors are added to the language, as in the case of vSPEM. On the other hand, the results obtained show that it is much more efficient to carry out modifications to processes by using the vSPEM variation mechanisms based on the modification of methods. If we compare the relative results obtained in diagram understandability and those of vSPEM variation mechanisms with regard to SPEM we observe that the improvement in understandability of the use of vSPEM variability mechanisms (126.99%) means that the worsening in the diagram understandability (34.87%) is compensated by an improvement that is 3.64 times greater than that worsening. Any improvement, as slight as it may be, to the tailoring task, which is carried out in a regular manner for the processes of all the projects of an organisation, brings about a benefit in the short–medium term. In this case, bearing in mind the results of software variant-rich processes using mechanisms based on variation points and variants with regard to the method modification mechanisms, it would seem clear that once the initial effort to understand the language (in this case vSPEM) has taken place, the benefits of applying our notation will be obtained in the short term. Furthermore, tailoring software processes from variant-rich processes by using the variation point and variant approach through the implementation of vSPEM implies the application of a more engineering-centred approach which would modify the definition of the methods in order to subsequently generate a 'tailored' process.

Variation mechanisms are applied over the variant-rich processes, which means their understandability rates also include the process diagram understandability rates. Because of that, they show the understandability level of the overall notation better.

4.3.2 Statistical analysis of the experimental results:

With respect to the hypotheses of the experiment shown in Section 4.1.2, the following set of null and alternative hypotheses was formulated.

Null hypothesis, $H_{0\text{DigU}}$: There is no significant difference between the understandability of vSPEM process model diagrams and the understandability of SPEM ones.

Alternative hypothesis 1, $H_{1\text{DigU}}$: The vSPEM process model diagrams are significantly less understandable than SPEM process model diagrams.

Alternative hypothesis 2, $H_{2\text{DigU}}$: The vSPEM process model diagrams are significantly more understandable than SPEM process model diagrams.

Table 6 Outliers generated for various participants

Participant	Outliers			Participant	Outliers		
	DigU	VMechU	Total		DigU	VMechU	Total
#1	0	1	1	#12	1	0	1
#2	1	0	1	#13	0	0	0
#3	2	0	2	#14	3	1	4
#4	3	0	3	#15	1	1	2
#5	0	1	1	#16	1	0	1
#6	0	0	0	#17	4	0	4
#7	0	0	0	#18	0	1	1
#8	1	1	2	#19	2	0	2
#9	1	0	1	#20	1	0	1
#10	1	0	1	#21	0	1	1
#11	0	1	1	total	22	8	30

Table 7 Descriptive statistics of diagram understandability results

Diagram Understandability measures	SPEM		vSPEM	
	Mean	Std Dev	Mean	Std Dev
correctness	27.38095238	5.977678886	22.31764706	9.446858592
time, s	72.75	40.24529457	100.4	68.64560472
efficiency, cor/s	0.47775	0.25036	0.31116	0.21185

Table 8 Descriptive statistics of Variability Mechanism Understandability

Variability Mechanism Understandability measures	SPEM		vSPEM	
	Mean	Std Dev	Mean	Std Dev
correctness	10.27160494	10.96359013	17.2345679	11.51115937
time, s	264.3703704	130.9691227	241.2345679	149.6676545
efficiency, cor/s	0.04177	0.04860	0.09480	0.09055

IET Softw., 2011, Vol. 5, Iss. 2, pp. 172–187
doi: 10.1049/iet-sen.2010.0020

Null hypothesis, $H_{0VMechU}$: There is no significant difference in understandability of the process variation mechanisms between SPEM and vSPEM.
Alternative hypothesis 1, $H_{1VMechU}$: The process variability mechanisms of vSPEM are significantly less understandable than the SPEM ones.
Alternative hypothesis 2, $H_{2VMechU}$: The process variability mechanisms of vSPEM are significantly more understandable than SPEM ones.

The following step was to obtain an insight into whether these differences were statistically significant. In order to verify this, the Mann–Whitney U and the Student t-tests have been applied to non-normal and to normal variables, respectively (a significance level of α has been stated at 0.05). To determine the normality of the variables, we have used the Kolmogorov–Smirnov test with the same significance level. These tests are used to measure the interaction between the independent variables under study when the dependent variables are measured. First the diagram understandability, and then the variability mechanism understandability are analysed. Tables 9–13 show the analyses performed.

The results of the Kolmogorov–Smirnov test with regard to the diagram understandability results are shown in Table 9. These results show that none of the variables is normal, signifying that a non-parametric test may be applied; the Mann–Whitney U test is therefore used. The results of this test are shown in Table 10.

The results of diagram understandability shown in Table 7 indicate that notation influences diagram understandability and that diagrams modelled with vSPEM are more difficult to understand than those modelled with SPEM. Based on the statistical analysis carried out in Table 10, we have obtained statistical evidence of these results ($p < \alpha = 0.05$) in all the three independent variables, so then, there is a significant difference and H_{0DigU} can be rejected. Moreover, considering the descriptive statistics shown in Table 7, SPEM allows us to specify process diagrams that are more correct, spending less time, which makes the efficiency

Table 9 Normality in Diagram Understandability results

Variable	Kolmogorov–Smirnov		Normal distribution	Test to apply
correctness	Z	1.899	no	U of Mann–Whitney
	Sig.	0.001		
time, s	Z	1.656	no	U of Mann–Whitney
	Sig.	0.008		
efficiency, cor/s	Z	2.186	no	U of Mann–Whitney
	Sig.	0.000		

Table 10 Mann–Whitney results for Diagram Understandability results

Variable	Mann–Whitney		Ranks		
			Model	Mean rank	Sum of ranks
correctness	U	2526.500	SPEM	97.42	8183.50
	Sig.	0.000	vSPEM	72.72	6181.50
time, s	U	2637.500	SPEM	73.90	6207.50
	Sig.	0.003	vSPEM	95.97	8157.50
efficiency,	U	2142.000	SPEM	102.00	8568.00
cor/s	Sig.	0.000	vSPEM	68.20	5797.00

Table 11 Normality test on Variability Mechanism Understandability results

Variable	Kolmogorov–Smirnov			
			Normal distribution	Test to apply
correctness	Z	1.807	no	U of Mann–Whitney
	Sig.	0.003		
time, s	Z	1.336	yes	t of Student
	Sig.	0.056		
efficiency, cor/s	Z	2.121	no	U of Mann–Whitney
	Sig.	0.000		

Table 12 Student t for variability mechanism time results

		Levenne test		t test for equality of means					95% Interval conf	
		F	Sig.	t	df	Sig.	Mean diff.	Std. Err. Diff.	Lower	Upper
time, s	equal variances	0.343	0.559	−1.047	160.000	0.297	−23.136	22.098	−66.777	20.505
	not equal variances	0.000	0.000	−1.047	157.233	0.297	−23.136	22.098	−66.783	20.511

IET Softw., 2011, Vol. 5, Iss. 2, pp. 172–187
doi: 10.1049/iet-sen.2010.0020

Table 13 Mann–Whitney U for variability mechanism correctness and efficiency results

Variable	Mann–Whitney		Ranks		
			Model	Mean rank	Sum of ranks
correctness	U	2189.500	SPEM	68.031	5510.500
	Sig.	0.000	vSPEM	68.200	5797.000
efficiency, cor/s	U	2109.500	SPEM	67.043	5430.500
	Sig.	0.000	vSPEM	95.957	7772.500

measure in understandability of SPEM better than vSPEM, so H_{1DigU} must be accepted.

- H_{1DigU}: The vSPEM process model diagrams are significantly less understandable than SPEM process model diagrams.

As with the diagram understandability variables, the normality of variability mechanisms variables was calculated by using the Kolmogorov–Smirnov test. Table 11 shows these results. Tables 12 and 13 then show the result of applying the Student t and the Mann–Whitney U tests, respectively.

From the statistics in Table 8, descriptive results show it is easier to modify process diagrams by using the vSPEM notation than by using SPEM notation. Furthermore, differences are also statistically significant in the case of the independent variables correctness and efficiency ($\text{sig} = 0.000 < \alpha = 0.05$), as Table 13 shows, signifying that $H_{0VMechU}$ must be rejected. Based on these results and the descriptive statistics shown in Table 8, we can say correctness in diagram modifications and their efficiency are statistically better when variability mechanisms of vSPEM are used instead of SPEM ones, which means $H_{2VMechU}$ must be accepted.

- $H_{2VMechU}$: The process variability mechanisms of vSPEM are significantly more understandable than SPEM ones.

4.4 Threats to validity

The various issues that might have threatened the validity of the results were analysed during the planning process of the experiment:

1. *Construct validity:* With regard to the validity of the construction of the experiment, the measures selected for each dependent variable aimed to assess the ease with which the participants were able to understand and modify the diagrams. These measures (time, correct answers and efficiency) are commonly used in studies that involve cognitive tasks [36], which is the nature of this research. They are also applied in studies in which dependent variables are evaluated (such as [37]).

However, the participants filled in the data in the given formularies manually, and the measures obtained from this may have lacked accuracy, that is, they may have failed to record the exact time at the beginning and at the end of each task. To alleviate this threat, the last part of the training sessions was used to solve a complete example using similar exercises to the experimental ones, and the experimental process was carefully explained. Furthermore, the experimental material included an initial page with detailed instructions. In addition, to lessen the possibility of the diagrams used in the experiment affecting the results, they were selected from the COMPETISOFT process reference model and their size and complexity were different within a low-medium complexity range.

2. *Internal validity:* Certain aspects might have threatened internal validity. To avoid this, these aspects were addressed in the following way: to reduce persistence effects, the subjects selected had never taken part in a similar experiment previously. The learning effects were reduced by the fact that all the diagrams solved by the participants were extracted from different COMPETISOFT processes, and the participants received the exercises in a random order. The participants knew about process modelling, and both notations (SPEM and vSPEM) were opportunely explained beforehand, but the process models used in the proposed exercises were not explained, signifying that internal validity was not affected by knowledge about the domain. In order to avoid the appearance of fatigue effects, the average duration of the experiment was an hour. We conducted a pilot experiment to verify this, after which we reduced the number of diagrams from ten to eight. The participants were motivated to participate in the experiment with the promise of obtaining extra points in their course marks. Plagiarism and influence among and between participants were controlled by supervising the experimental run.

3. *External validity:* To avoid threats in external validity, the material, training and tasks planned to carry out the experiment were designed bearing in mind the time restrictions and the knowledge level of an average fourth course student. It is commented that professionals only have limited time for process engineering tasks. As the subjects were not professionals and as there only were 27 students in the course, of whom only 22 took part in the experiment, and 20 (plus the pilot) of these latter gave us a valid set of exercises (which means $21*8 = 168$ cases), the power of generalisation of results is limited. Replications of this experiment are planned, with the aim of empowering the experimental validity by means of increasing the number of subjects and, when possible, using process engineers as subjects. The aim of these would be to confirm and improve the results obtained. In these replications the variations must be more realistic and, if possible, they should be extracted from industrial variant-rich processes.

5 Conclusions

In this paper the importance of adapting software processes to the characteristics both of projects and of organisations has been explained. The objective of such adaptation is to attain high capacity levels and permit quality products to be obtained. The literature contains diverse initiatives focusing on the creation of processes which are 'close' to the projects, among which the variant-rich process approach

IET Softw., 2011, Vol. 5, Iss. 2, pp. 172–187
doi: 10.1049/iet-sen.2010.0020

also permits the majority of the advantages proportioned by product lines to be transferred to the field of software processes.

Furthermore, in order to facilitate the specification and understanding of software processes, a model that is in agreement with well-defined elements is also necessary, such as the SPEM metamodel. Moreover, it has been modified according to the product line philosophy to create vSPEM, an extension that provides SPEM with the variation mechanisms for modelling variant-rich processes based on variation points and variants.

This paper describes an empirical study conducted to study the understandability of variant-rich processes, using on the one hand variation points and variants, and on the other modifications to the definition of the method, when carrying out adaptations in processes according to the needs of a project or organisation. This is done by comparing the vSPEM language with that of SPEM, while bearing in mind two key factors of a notation's usability: the understandability of its respective diagrams and the efficiency of its variation mechanisms.

With regard to the understandability of the diagrams, the results obtained indicate that, since vSPEM includes more constructors, its results are worse. However, given that the variation points and the variants are elements have been especially designed to permit variation in software processes and their components, process tailoring according to the characteristics of the organisation and projects is much simpler with vSPEM than with SPEM. These results were to be expected: it was feared that the addition of new constructors would not make the Diagram Understandability better, but the Variability Mechanisms had to be much more understandable because vSPEM is a notation that is specifically designed for process variability. However it could not be foreseen how much better these results would be. Having seen the results obtained, we believe a good starting point could be the fact that the vSPEM Variability Mechanism Understandability is 1.26 times better than in the case of SPEM, and it is 3.64 times the worsening in Diagram Understandability.

It is not in vain that, when using vSPEM, the software processes modifications can be directly carried out on the process models, instead of on the definition of the process method used. Moreover, its mechanisms permit us to specify clearly what may vary and among which new values this may occur. It is similarly possible to observe that the improvement obtained in the understandability of the modifications carried out more than compensates for the worsening in the understandability of the diagrams. Furthermore, it is expected that the performances of vSPEM should be greater once it has been applied the first time, because of the 'extra' effort to learn the notation. Therefore vSPEM can constitute a usable notation for representing variability in software processes, as the 'everyday' benefits are 3.64 times the 'first time' worsening, redeeming the initial learning effort. After analysing the statistics, we have found evidence that the aforementioned differences are statistically significant with regard to efficiency.

The results presented in this paper are, therefore, a good starting point through which to design variant-rich processes using vSPEM based on variation points and variants. This paradigm can be used to adapt software processes to the characteristics of projects and organisations systematically. This, in turn, will permit the institutionalisation of processes, and will definitively define the environment for the institutionalisation of processes based on software variant-rich processes. However, bearing in mind that, according to the results of the experiment, the understandability of vSPEM is lower than that of SPEM, we are conscious of the fact that it is first necessary to tackle the question of improving the proposed notation.

As future work we should first try to get a deeper understanding of the process variability requirements in organisations and their fulfilment by process variability mechanisms. The results of the study described in this article could be seen as motivators for replications (e.g. with variations of the context). These replicas must focus on discovering the aspects of the notation that are most difficult to understand. A following step would be to try to improve these aspects as well as the semantics of vSPEM for modelling variability in software processes. In order to offer a stronger validation, replications must be designed to include a larger sample size and a more realistic environment. Results from such studies or replications could be used for the evolution of process modelling notations and for their application in research and practice.

6 Acknowledgments

This work is partially supported by the Program FPU of the Spanish Ministerio de Educación, and by the INGENIO (JCCM, PAC08–0154-9262), MEDUSAS (CDTI (MICINN), IDI-20090557), PEGASO/MAGO (MICINN and FEDER, TIN2009-13718-C02-01), and ALTAMIRA (JCCM, Fondo Social Europeo, PII2I09-0106-2463) projects. We would also like to thank Marcus Ciolkowski and Sonnhild Namingha from Fraunhofer IESE for reviewing a first version of this article and for providing suggestions to improve it.

7 References

1 Fuggetta, A.: 'Software process: a roadmap', in Finkelstein, A. (Ed.): 'The future of software engineering' (ACM, Limerick, Ireland, 2000), pp. 25–34
2 Cugola, G., Ghezzi, C.: 'Software processes: a retrospective and a path to the future', *Softw. Process: Improv. Pract.*, 1998, **4**, (3), pp. 101–123
3 Humphrey, W.S., Kellner, M.I. (Eds.): 'Software process modeling: principles of entity process models'. Proc. 11th ICSE, 1989
4 Sheard, S.A., Lake, J.: 'Systems engineering standards and models compared'. Proc. Eighth Int. Symp. on Systems Engineering (INCOSE)', 1998, pp. 589–605
5 SEI: 'CMMI for systems engineering/software engineering, version 1.1'. 2002Contract No.: Document Number, Software Engineering Institute (SEI), Pittsburgh
6 ISO: 'ISO/IEC 12207:2002/FDAM 2. Information technology – software life cycle processes'. 2004Contract No.: Document Number, International Organization for Standardization, Geneva
7 SEI: 'Standard CMMI® appraisal method for process improvement (SCAMPI), version 1.1: method definition document (CMU/SEI-2001-HB-001)'. 2001Contract No.: Document Number, Software Engineering Institute (SEI), Pittsburgh
8 ISO: 'ISO/IEC 15504-2:2003/Cor.1:2004(E). Information technology – process assessment – Part 2: performing an assessment'. 2004Contract No.: Document Number, International Organization for Standardization, Geneva
9 ISO: 'ISO/IEC 15504-4:2003/Cor.1:2004(E). Information technology – process assessment – Part 4: guidance on use for process improvement and process capability determination'. 2004Contract No.: Document Number, International Organization for Standardization, Geneva
10 Yoon, I.-C., Min, S.-Y., Bae, D.-H.: 'Tailoring and verifying software process'. Eighth APSEC, Macao, China, IEEE CS, 2001, pp. 202–209
11 Humphrey, W.: 'Managing the software process' (Addison-Wesley, 1989)
12 Chrissis, M.B., Konrad, M., Shrum, S.: 'CMMI: guidelines for process integration and product improvement' (Pearson, Boston, 2006)
13 Bayer, J., Buhl, W., Giese, C., *et al.*: 'Process family engineering: modeling variant-rich processes'. Report No.: 18/2005, Contract No.: Document Number, PESOA Project, Postdam, Alemania, July 2005
14 Rombach, D.: 'Integrated software process and product lines'. ISPW, 2005, pp. 83–90
15 Parnas, D.L.: 'On the design and development of program families', *IEEE Trans. Softw. Eng.*, 1976, **SE-2**, (1), pp. 1–9
16 Sutton, S.M., Osterweil, L.J. (Eds.): 'Product families and process families'. Tenth Int. Software Process Workshop ISPW'06, 1996
17 OMG: 'Software process engineering metamodel specification'. Report No.: ptc/07-03-03Contract No.: Document Number, Object Management Group, October, 2007
18 Martínez-Ruiz, T., García, F., Piattini, M.: 'Towards a SPEM v2.0 extension to define process lines variability mechanisms', in Lee, R. (Ed.): 'Software engineering research, management and applications' (Springer Verlag, Praga, 2008), pp. 115–130
19 Martínez-Ruiz, T., García, F., Piattini, M. (Eds.): 'Enhanced variability mechanisms to manage software process lines'. EUROSPI 2009 (Alcalá de Henares (Madrid), Publizon, 2009)
20 Clements, P., Northrop, L.: 'Software product lines. Practices and patterns' (Addison-Wesley, Boston, 2002)
21 Ocampo, A., Münch, J.: 'Software process variability: concepts and approaches'. Report No.: Technical Report IESE-Report-124.04/EContract No.: Document Number, Fraunhofer IESE, Kaiserslautern, December 2004
22 Puhlmann, F., Schnieders, A., Weiland, J., Weske, M.: 'Process family engineering: variability mechanisms for process models'. Report No.: 17/2005Contract No.: Document Number, Technical, PESOA, Postdam, Alemania, June 2005
23 Schneiders, A., Weske, M.: 'Activity diagram based process family architectures for enterprise application families', in Doumeingts, G., Müller, J.P., Morel, G., Vallespir, B. (Eds.): 'In enterprise interoperability: new challenges and approaches' (Springer-Verlag, London, 2007), pp. 67–76
24 Simidchieva, B.I., Clarke, L.A., Osterweil, L.: 'Representing process variation with a process family'. ICSP 2007, 2007, pp. 121–133
25 Armbrust, O., Katahira, M., Miyamoto, Y., Münch, J., Nakao, H., Ocampo, A. (Eds.): 'Scoping software process models – initial concepts and experience from defining space standards', ICSP, Leipzig, Germany, 2008
26 Armbrust, O., Katahira, M., Miyamoto, Y., Münch, J., Nakao, H., Ocampo, A.: 'Scoping software process lines', *Softw. Process Improv. Pract.*, 2009, **14**, (3), pp. 181–197
27 Ma, J.-K., Shi, L., Wang, Y.-S., Mei, H. (Eds.): 'Process aspect: handling crosscutting concerns during software process improvement'. ICSP, 2009
28 Filman, R.E., Elrad, T., Clarke, S., Aksit, M.: 'Aspect-oriented software development' (Addison-Wesley, Boston, MA, 2004)
29 Martins, P.V., Silva, A.R.: 'ProPAM: discussion for a new SPI approach', *Softw. Qual. Manage.*, 2009, **11**, (2), pp. 4–17
30 Martins, P.V., Silva, A.R. (Eds.): 'ProPAM. SPI based on process and project alignment'. IRMA 2007, 2007
31 Killisperger, P., Stumptner, M., Peters, G., Grossmann, G., Stückl, T. (Eds.): 'Meta model based architecture for software process instantiation'. ICSP, 2009
32 Silva Barreto, A., Murta, L., Rocha, A.R. (Eds.): 'Software process definition: a reuse-based approach'. XXXIV Conferencia Latinoamericana de Informática, Santa Fe, Argentina, 2008
33 Martínez-Ruiz, T., García, F., Piattini, M. (Eds.): 'Process institutionalization using software process lines'. ICEIS 2009, Milan, 2009
34 Oktaba, H., Piattini, M., Pino, F., *et al.*: 'COMPETISOFT: an improvement strategy for small latin-american software organizations', in Oktaba, H., Piattini, M. (Eds.): 'Software process improvement for small and medium enterprises: techniques and case studies' (Idea Group Inc., 2008)
35 Garzás, J., García, F., Piattini, M.: 'Do rules and patterns affect design maintainability?', *J. Comput. Sci. Technol.*, 2009, **24**, (2), pp. 262–272
36 Eysenck, M., Keane, M.: 'Cognitive psychology: a student's handbook' (Lawrence Erlbaum Associates, Mahwah, NJ, 2000)
37 Genero, M., Moody, D., Piattini, M.: 'Assessing the capability of internal metrics as early indicators of maintenance effort through experimentation', *J. Softw. Maint. Evol. Res. Pract.*, 2005, **17**, (3), pp. 225–246
38 Briand, L.C., Bunse, C., Daly, J.W.: 'A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs', *IEEE Trans. Softw. Eng.*, 2001, **27**, (6), pp. 513–530

www.ietdl.org

IET Softw., 2011, Vol. 5, Iss. 2, pp. 172–187
doi: 10.1049/iet-sen.2010.0020