Applying AOSE Concepts to Model Crosscutting Variability in Variant-Rich Processes

Tomás Martínez-Ruiz, Félix García, Mario Piattini
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
+34 926 295300 ext. 6648, 6475, 3715
{tomas.martinez, felix.garcia, mario.piattini}@uclm.es

Jürgen Münch
University of Helsinki and Fraunhofer IESE
Gustaf Hällströminkatu 2b, FI-00014 Helsinki, Finland
+358 9 191 51330
juergen.muench[at]cs[dot]Helsinki[dot]fi

Abstract—Software process models need to be variant-rich, in the sense that they should be systematically customizable to specific project goals and project environments. It is currently very difficult to model Variant-Rich Process (VRP) because variability mechanisms are largely missing in modern process modeling languages. Variability mechanisms from other domains, such as programming languages, might be suitable for the representation of variability and could be adapted to the modeling of software processes. Mechanisms from Software Product Line Engineering (SPLE) and concepts from Aspect-Oriented Software Engineering (AOSE) show particular promise when modeling variability. This paper presents an approach that integrates variability concepts from SPL and AOSE in the design of a VRP approach for the systematic support of tailoring in software processes. This approach has also been implemented in SPEM, resulting in the vSPEM notation. It has been used in a pilot application, which indicates that our approach based on AOSE can make process tailoring easier and more productive.

AOSE, Variant-Rich Processes, Process Variability, Process Tailoring, Process Lines, Aspect Oriented Software Development.

I. INTRODUCTION

The tailoring of software processes makes them fit for different organizations and projects [1]. It also leads to the appearance of certain needs that are similar to those found in Software Product Line Engineering (SPLE). The Variant-Rich Process (VRP) approach was developed to meet these needs. This approach proposes the inclusion of variants of process-composing elements in the processes themselves through the use of on-point variations, thus making process tailoring easier. It is based on applying assets from SPL and AOSE in the vSPEM notation. It has been used in a pilot application, which indicates that our approach based on AOSE can make process tailoring easier and more productive.

AOSE, Variant-Rich Processes, Process Variability, Process Tailoring, Process Lines, Aspect Oriented Software Development.

II. STATE OF THE ART

The behavior of a software process depends on its structure [11], and process tailoring activities are useful in making each process fit its appropriate behavior when it is instantiated. The topic of process tailoring is widely dealt with in literature, as Pedreira et al. show [12], and is supported in several ways, as the systematic review of Martínez-Ruiz et al. demonstrates [4]. Tailoring is carried out by means of changes to the elements of which the processes are composed and to their relations, principally in activities, artifacts, and roles. Variations are also executed in two ways, namely in a detailed and intensive manner in order to configure each element with its suitable value, and in a crosscutting manner in order to configure a large number of process composing elements simultaneously.

Moreover, since the emergence of the product line approach [13], most of the initiatives with which to model variability in processes have attempted to adapt it [4], as was proposed by Rombach in [2]. Some works concerning process tailoring have thus been proposed based on process lines and variation mechanisms [14], using stereotypes [15] and defining new specific notations [9, 10]. Other proposals include environments in which to represent variability and to carry out the tailoring and monitoring of the process during its execution.
Some approaches deal with the application of AOSE to product lines. According to Kulesza et al. [19], aspects may be used to introduce flexibility into a system. Each aspect signifies temporal behavior, which may imply variability. Laddaga et al. [20] show aspect management techniques that may be applied to dynamic architectures, mainly product lines. In this respect, Apel et al. [21] present some problems of Feature-Oriented Programming and how they are solved by means of using AOSE. The correspondence between elements from AOSE and variability concepts in product lines is shown in [22] and [23]. González-Bauxilli et al. [24] present a meta-model that can be used to combine aspects from product lines and aspects, and they define how to transform traditional models into aspect models.

According to López Herrejon and Batory [25], advice and pointcuts may modify a class or method behavior. They can also be formalized by using use case slides and algebra techniques. Colyer et al. [26] describe some of the limitations involved in applying aspects to product lines: the concerns must be orthogonal, without dependencies between them, in order to allow their independent use. Mezini and Ostermann [27] present an analysis of how to model variability using mechanisms of aspect-oriented languages, mainly Caesar. Figueiredo et al. [28] present and evaluate the suitability of AOSE in modeling product lines and compare the results with other product line approaches. The results obtained show that AOSE is more efficient.

Aspects have also been combined with business processes, as proposed by Odgers [29], because of their flexibility and dynamism. Moreover, Charfi et al. [30] present AO4BPEL, an aspect-oriented extension to BPEL which deals with support for crosscutting concerns and dynamic adaptation. In addition, the work of Sutton [8] allows us to consider Aspects in processes [31]. Quites et al. [7] propose using them to design high-level management policies in software process models. Mishali and Katz [6] propose applying aspects to monitor or enforce XP practices over Eclipse.

In a previous work, vSPEM, a SPLE-based Variant-Rich Process approach, which could be used to carry out any variation supports process tailoring based on aspect mechanisms rather than being focused solely on process reuse. It is also a generic approach, which could be used to carry out any variation (including any specific characteristics) in a software process, and in any of the process constructors.

III. ENRICHING THE VARIANT-RICH PROCESS APPROACH WITH AOSE CONCEPTS

The AOSE approach was analyzed and mapped onto the SPELE-based Variant-Rich Process approach through an analysis of the AspectJ language [32, 33] and its constructors (see references [32, 33] for further information about these constructors). It was then enhanced with the capability to handle crosscutting variations. The vSPEM notation was also enriched with new AOSE-inspired constructors. The added or modified elements are highlighted in Fig. 1.

A. Crosscutting Variability in Variant-Rich Processes

In order to introduce aspects into process line variability, analogies may be determined between the concepts in AOSE and Variant-Rich software processes, as Tables I and II show.

Both tables show that there are similar concepts in software programming and software processes. They also show that variant rich processes lack some of those which could be used to manage crosscutting variations by using the AOSE paradigm assets. These have been then modeled as Fig. 1 presents.

B. Variation Points (Redefined)

Variation points are the places where the elements included in the crosscutting variation should be inserted. Table III describes the different types of variation points that exist in Variant-Rich Processes, based on analyzing the process elements’ behavior and their variability support.

| Programming Concept | Variant-Rich Process Concept |
|----------------------|-----------------------------|
| Class                | Process or sub process      |
| Methods              | Work Units (activities, tasks …) |
| Attributes           | Work Products               |
| Constants            | Resources (humans, tools…)  |

| AOSE Concept | New Variant-Rich Process Concept |
|-------------|----------------------------------|
| Crosscutting concern | Crosscutting Variation |
| Aspect      | Process Aspect (see Sect. III.E) |
| Join point  | Variation Point (see Sect. III.B) |
| Pointcut    | Process Pointcut (see Sect. III.C) |
| Advice declaration | Process Advice (see Sect. III.D) |
| Behavior in the advice | Variants |

| AOSE Join Point | Variant-Rich Process Variation Point |
|-----------------|-------------------------------------|
| A call to a method or constructor | Call of a work unit (activity, task) the called element (in the caller element) |
| Execution of method, constructor or advice | Call of a work unit (activity, task…) in the called element |
| Access or update of a field | Use or creation of a work product |
| Access of a field | Use of a resource (human, tool) |
| Initialization of classes and objects | First use or delivery of work products |
The properties of Process Pointcuts are their names and expressions. The latter is a string which filters the variation points used to carry out the crosscutting variation. Pointcuts also include a method with which to evaluate the expression and obtain the variability elements that satisfy this expression.

D. Process Advices and Variants

Process advices, therefore, focus on modifying the occupation relationships between the variation points and the variants to satisfy the crosscutting variation. This is achieved by creating certain designators (see Note 1). By using these, process advices are declared as:

\[
\text{Advice } \langle \text{process pointcut} \rangle \langle \langle \text{VariabilityElements list} \rangle \rangle \{ \langle \text{actions of the crosscutting variation} \rangle \}
\]

“Advice” is therefore the word used to indicate process advice; process pointcut is the name of the process pointcut that activates the advice (or the expression combining process pointcuts by using logic connectors). The evaluation method creates occupation relationships by using the variability elements.

E. Process Aspect

Process aspects support every crosscutting variation in variant-rich software processes. They are therefore composed of process pointcuts and some process advices, and include some variants and perhaps variation points that are used to carry out the crosscutting variation. They also encapsulate and abstract the user from the aforementioned elements.

Since process aspects are used to introduce variability by carrying out variations, they are a specialization of both the VElement constructor (more details of VElement can be found in [9]). Process aspects also include the active attribute which shows whether the aspect is used to tailor the Variant-Rich Process, and one operation which makes the actual changes.

IV. PILOT STUDY USING CROSSCUTTING VARIATIONS

The aforementioned approach has been used to model variability in the Java Process Line (more details about this process line can be found in [34]). This process line includes several variabilities (as Table IV details), but when it is tailoring, some of these are similarly treated, depending on the same criterion. The on-point variations of these points are encapsulated in one crosscutting variation. This study shows how to create a crosscutting variation for the first criterion.

This criterion implies that if Satellite 2 is developed, one task and one work product must be added in order to tailor the process. This implies one aspect, including some variants and variation points with which to realize the on-point variations.

The aspect additionally includes a process pointcut, which is also transparent from the end user’s point of view. Its aim is

| Variability                      | Criterion                        |
|---------------------------------|----------------------------------|
| FMECA work product             | Include in satellite 2           |
| Analyze HW and SW interaction task | Include in satellite 2         |
| Rationale for each requirement artifact | Not included in science missions |
| Quality source code work product | Not included in science missions |
| Requirements in design work product | Not included in science missions |
to filter task and work product variation points in Activity 1.2.2., *Software Design*, using the following constructors. Patterns have been used to avoid listing the full names:

```plaintext
pointcut ppc1 (VPTask vpt1, VPWorkP vpw1, VPWorkP vpw2):
  vpt1=(execution("1.2.2*"));
  vpw1=(use(*) & & within("1.2.2*"));

This pointcut obtains a task variation point within the activity and may be used by the advice. Some other variation points are also retrieved. They contain the input and output products of the task. These variation points are taken by the advice. This is written as:

```plaintext
advice ppc1 (VPTask vpt1, VPWorkP vpw1, VPWorkP vpw2){
  vpt1.occupe(Analyze HW SW Interact.);
  vpw1.occupe(FMECA); }
```

The instructions in the process advice allow the variants (FMECA work product and the Analyze HW SW Interaction task) to be placed in the variation points obtained previously–vpw1 and vpt1, respectively. Moreover, as is set out in detail in [10], the variant2variant dependency between them is converted into a relationship in the completely opposite direction. After interpreting variability, the FMECA work product and the Analyze HW SW Interaction task are then created as new elements in Activity 1.2.2., *Software Design*.

With the AOSE-based mechanisms, the tailoring of a process with crosscutting variations is now simplified by activating aspects rather than selecting which variants to place in which variation points, and combining them. This reduces the effort and abstracts the user when tailoring the process.

A. Lessons Learned

After analyzing the application of the vSPEM in a real Variant-Rich Process, some advantages and disadvantages of using crosscutting variations were found. The first advantage was that crosscutting variations include several on-point variations that make it unnecessary to realize these (on point) variations manually, and this thus reduces the tailoring effort. Taking the pilot application we have described above, the Variant-Rich Process initially contained five on-point variations to be evaluated in order to obtain a tailored process, which signifies a tailoring effort of \(5n\), where \(n\) is the number of tailored processes from the Variant-Rich Process. After applying the approach proposed in this paper, these were reduced to two crosscutting variations. This signifies an effort of \(2n\), bringing about a reduction for half the initial effort for process tailoring. Once the AOSE-based mechanisms have been defined, they could be easily used and reused for tailoring processes. In addition, all the new elements defined over the process line are transparent from the user viewpoint, with the exception of the aspect, which is actually “activated”. This signifies that the use of crosscutting mechanisms does not imply an increase in process tailoring complexity.

Moreover, when varying process composing elements, it is necessary to ensure that the variations of these elements remain consistent. The crosscutting of variations ensures that consistency is not the process engineer’s responsibility, since it is configured in the process advices and process pointcuts, and allows the user to be abstracted out from such details, which is the second advantage. In addition, as process aspects are consistent themselves, they could be built without dependences between them, as Colyer [26] proposes.

However, AOSE-based crosscutting variations have one main disadvantage: they do not offer the user the ability to decide, in detail, which variants should occupy each variation point. On the other hand, this disadvantage of crosscutting variations is the most important advantage of SPLE-based on-point variations. As a result, the combination of both types of variations leads to the creation of the most complete process tailoring paradigm.

To sum up, this pilot study shows very promising results: Developing an SPLE&AOSE-based Variant-Rich Process approach is a promising and robust initiative for supporting effective process tailoring and making it feasible for organizations to use it to tailor their own software processes in the near future. In addition, the more complex the processes are, the more on-point variations they include, and the more are encapsulated into aspects, which makes them more useful mechanisms in tailoring real and complex software processes.

V. CONCLUSIONS AND FUTURE WORK

Software processes must be tailored by means of on-point and crosscutting variations. The Variant-Rich Process approach must therefore focus on supporting both types of tailoring in order to offer organizations usable mechanisms with which to tailor their processes. Since there is no way to manage crosscutting variations through tailoring mechanisms based on product lines, this paper focuses on the transferred AOSE paradigm to fill this gap and to support process model tailoring as required by real processes.

As a result, the Variant-Rich Process approach has been enriched with new AOSE-inspired concepts such as the process aspect. These were merged with the previous SPLE-based concepts, with the intention of all of them being able to manage variability in a consistent and complementary manner.

Managing crosscutting variations in Variant-Rich Processes implies two main advantages. First, it ensures variability consistency, and second, it facilitates tailoring. As variations in real processes involve several elements, this reveals that crosscutting variability mechanisms are actually needed in order to guarantee consistency when varying all these elements. Moreover, tailoring processes from a Variant-Rich Process without consideration for crosscutting variations signifies that all on-point variations must be decided on one by one. In contrast, crosscutting variations allow abstraction, signifying that these variations can be carried out simultaneously.

Our approach has also been implemented on SPEM, resulting in the vSPEM notation. This notation has been used to model variability in a real process and later to resolve this variability by tailoring processes. The main lesson learned from the results obtained in the pilot study are that the AOSE-based Variant-Rich Process approach promises to be a powerful and robust initiative for modeling variability in software processes in alignment with tailoring requirements in current software development organizations. It also shows that
it is possible not only to use AOSE concepts to model variability, but also to employ AOSE lifecycle techniques for coping, along with determining variability in software process models—without variability.

As future work, we shall focus on analyzing other AOSE implementations to improve our approach. We also intend to carry out experiments to verify the usability of the vSPEM notation extended with AOSE, along with its applicability to real processes. This approach will also be included as a plug-in in the Eclipse framework in an effort to facilitate process tailoring. Finally, organizations need to tailor their processes before representing them, and if they are to improve, processes must be more and more capable. Process institutionalization is the only way to do this. The process tailoring techniques we have developed may therefore be considered as a basis for building an institutionalization framework based on process tailoring and standardization. As institutionalization leads towards better process evolution, fragility problems appearing in the Variant-Rich Process approach owing to the use of aspects will be mitigated. This will be achieved by controlling how the processes and their process aspects need to evolve.

ACKNOWLEDGMENTS

This work is partially supported by the Program FPU of the Spanish Ministry de Educacion, and by the PEGASO/MAGO (MICINN and FEDER, TIN2009-13718-C02-01), MEDUSAS (CDTI (MICINN), IDI-20090557), ALTAMIRA (JCCM, Fondo Social Europeo, PII2I09-0106-2463) and INGENIOSO (MICINN and FEDER, TIN2009-13718-C02-02), projects. We would also like to thank our colleagues at the Japanese Aerospace Exploration Agency (JAXA) for their collaboration, and Sonnhild Namingha from Fraunhofer IESE for reviewing a first version of this paper.

REFERENCES

[1] I.-C. Yoon, S.-Y. Min, and D.-H. Bae, “Tailoring and Verifying Software Process,” in 8th ASEPAC Macao, China: IEEE CS, 2001, pp. 202-209.

[2] D. Rombach, “Integrated Software Process and Product Lines,” in ISPW, vol. LNCS 3840, M. Li, B. Boehm, and L. Osterweil, Eds. Beijing, China: Springer, 2005, pp. 83-90.

[3] L. Osterweil, “Software Processes Are Software Too,” in 9th ICSE Monterey, CA: IEEE Computer Press, 1987, pp. 2-13.

[4] T. Martínez-Ruiz, J. Münch, F. García, and M. Piattini, “Requirements and Constructors for Modeling Variability in Software Processes, a Systematic Review,” sent to SQJ, 2009.

[5] R. E. Filman, T. Elrad, S. Clarke, and M. Aksit, Aspect-Oriented Software Development. Boston, MA: Addison-Wesley, 2004.

[6] O. Mishali and S. Katz. “Using Aspects to Support the Software Process: XP over Eclipse,” in AOSD 2006 Bonn, Germany: ACM, 2006.

[7] R. Qutes Reis, C. A. Lima Reis, H. Schlobbe, and D. J. Nunes, “Towards and Aspect-Oriented Approach to Improve the Reusability of Software Process Models,” in AOSD workshops Enschede, The Netherlands, 2002.

[8] S. M. Sutton, “Aspect-Oriented Software Development and Software Process,” in ISPW, vol. LNCS 3840, M. Li, B. Boehm, and L. J. Osterweil, Eds. Springer, 2005, pp. 177-191.

[9] T. Martínez-Ruiz, F. García, and M. Piattini, “Towards a SPEM v2.0 Extension to Define Process Lines Variability Mechanisms,” in SERA, vol. SCI 150, R. Lee, Ed. Praga: Springer Verlag, 2008, pp. 115-130.

[10] T. Martínez-Ruiz, F. García, and M. Piattini, “Enhanced Variability Mechanisms to Manage Software Process Lines,” in EUROSPI 2009 Alcalá de Henares (Madrid): Publizon, 2009, pp. 12-13-12.23.

[11] L. Osterweil, “Unifying Microprocess and Macroprocess Research,” in ISPW, vol. LNCS 3840, M. Li, B. Boehm, and L. J. Osterweil, Eds.: Springer, 2005, pp. 68-74.

[12] Ó. Pedreira, M. Piattini, M. R. Luaces, and N. Brisaboa, “A systematic review of software process tailoring,” ACM SIGSOFT Software Engineering Notes, vol. 32, pp. 1-6, 2007.

[13] P. Clements and L. Northrop, Software Product Lines. Practices and Patterns. Boston: Addison-Wesley, 2002.

[14] F. Puhlmann, A. Schnieders, J. Weiland, and M. Weske, “Process Family Engineering: Variability Mechanisms for Process Models,” PESOA Project, Postdam, Almanya, Technical 17/2005, 2005.

[15] A. Schnieders and M. Weske, “Activity Diagram Based Process Family Architectures for Enterprise Application Families,” in Enterprise Interoperability: New Challenges and Approaches. vol. II, G. Doumenigts, J. P. Müller, G. Morel, and B. Vallespir, Eds. London: Springer, 2007, pp. 67-76.

[16] P. V. Martins and A. R. Silva, “ProPAM: Discussion for a New SPI Approach,” SQJ, vol. 11, pp. 4-17, 2009.

[17] P. Kllisiperger, M. Stumptner, G. Peters, G. Grossmann, and T. Stückl, “Meta Model Based Architecture for Software Process Instantiation,” in ICSF, vol. LNCS 5543, Y.-s. Wang, Ed. Vancouver, Canada: Springer Verlag, 2009, pp. 63-74.

[18] T. Martínez-Ruiz, F. García, and M. Piattini, “Process Institutionalization using Software Process Lines,” in ICSER 2009. vol. ISAS, J. Cordeiro and J. Filipe, Eds. Milan, 2009, pp. 359-362.

[19] U. Kulesza, V. Alves, A. García, C. d. Lucena, and P. Borba, “Improving Extensibility of Object-Oriented Frameworks with Aspect-Oriented Programming,” in ICSR. vol. LNCS 4039, M. Morisio, Ed. Torino, Italia: Springer-Verlag, 2006, pp. 231-245.

[20] R. Laddaga, P. Robertson, and H. Shrobe, “Aspects of the Real world,” OOPSLA, 2001.

[21] S. Apel, T. Leich, M. Rosenmüller, and G. Saake, “Combining Feature-Oriented and Aspect-Oriented Programming to Support Software Evolution,” in in AMSE’05, at ECOOP’05, W. Cazzola, S. Chiba, G. Saake, and T. Touwé, Eds.: Fakultät für Informatik, Universität Magdeburg, 2005, pp. 3-14.

[22] S.-H. Heo and E. M. Choy, “Representation of Variability in Software Product Line Using Aspect-Oriented Programming,” in 4th SERA, Y.-T. Song, Ed. Washington DC, USA: IEEE CS, 2006, pp. 66-73.

[23] M. Anastasopoulos and D. Muthig, “An Evaluation of Aspect-Oriented Programming as a Product Line Implementation Technology,” in 8th ICSR. vol. LNCS 3107 Madrid: Springer-Verlag, 2004, pp. 141-156.

[24] B. González-Baixauli, M. A. Laguna, and J. C. Sampaio, “A Meta-Model for Requirements Variability Analysis: Application to Tool Generation and Model Composition,” in 2nd ICSOFT, 2007.

[25] R. López-Herrejon and D. Batory, “Modeling Features in Aspect-Based Product Lines with Use Case Slices: An Exploratory Case Study,” in MoDELS 2006. vol. LNCS 4364, T. Kühne, Ed. Genoa, Italy: Springer-Verlag, 2007, pp. 6-16.

[26] A. Colyer, A. Rashid, and G. Blair, “On the Separation of Concerns in Program Families,” Lancaster Univ. COMP-001-2004, 2004.

[27] M. de Leoni and K. Ostermern, “Variability management with feature-oriented programming and aspects,” in 12th SIGSOFT Newport Beach, CA, USA: ACM, 2004, pp. 127-136.

[28] E. Figueiredo, N. Cacho, C. Sant’Anna, M. Monteiro, U. Kulesza, A. García, S. Soares, F. Ferrari, S. Khan, F. Castor Filho, and F. Dantas, “Evolving Software Product Lines with Aspects: An Empirical Study on Design Stability,” in ICSE Leipzig, Germany: ACM, 2008, pp. 261-270.

[29] B. Odgers and S. Thompsons, “Aspect Oriented Process Engineering,” in ECOOP Workshops, vol. LNCS 1743, A. M. D. Moreira and S. Demeyer, Eds. Lisbon: Springer, p. 295.

[30] A. Charfi and M. Mezini, “AOABPEL: An Aspect-Oriented Extension to BPEL,” in World Wide Web, Springer, 2007, pp. 309-344.

[31] J.-K. Ma, L. Shi, Y.-s. Wang, and H. Mei, “Process Aspect: Handling Crosscutting Concerns during Software Process Improvement,” in ICSP, vol. LNCS 5543, Y.-s. Wang, Ed. Vancouver, Canada: Springer Verlag, 2009, pp. 124-135.

[32] A. Colyer, A. Clement, G. Harley, and M. Webster, Eclipse AspectJ, 1 ed. Hargerstown: Addison-Wesley, 2005.

[33] I. Kiselev, “Aspect-Oriented Programming with AspectJ.” Indianapolis: SAMS, 2003.

[34] O. Armbrust, M. Katahira, Y. Miyamoto, J. Münch, H. Nakao, and A. Ocampa, “Prospecting Software Process Lines,” Software Process Improvement and Pratc., vol. 14, pp. 181-197, 2009.