Editorial

Early Aspects: Aspect-oriented Requirements Engineering and Architecture Design

Software development is a discipline that is continuously evolving. New techniques, methodologies and tools continue to appear and shape the way we modularise and compose software systems. An interesting characteristic of this continuously evolving discipline is the reverse introduction of the techniques. Historically, most new development techniques are introduced at the programming level and their concepts subsequently travel up the development life cycle to be applied at the earlier stages, for example, requirements engineering, analysis and design. We have seen this evolution trend, in the 1970's, with structured techniques and, in the late 1980's and early 1990's, with object-oriented techniques. For example, object-oriented concepts were initially introduced by the programming language SIMULA-67 and are now applied throughout the software life cycle with standard notations, such as the Unified Modelling Language (UML), available for modelling, analysis and design.

Aspect-oriented software development (AOSD) [1] is another step towards achieving improved modularity during software development. AOSD focuses on a specific type of concerns, crosscutting ones, which span traditional module boundaries (for example, classes and objects in an object-oriented decomposition). Examples of such crosscutting concerns include distribution, persistence, security and real-time behaviour. The modules encapsulating such crosscutting concerns using an AOSD technique are referred to as *aspects* while the composition process, which integrates the aspects with other modules in the system, is referred to as *weaving*.

The reverse introduction trend is also evident in this newest addition to the set of software development techniques. AOSD was introduced at the programming level with AspectJ [2–4] playing a key role in its early adoption by researchers and practitioners; the notion of aspect-orientation is rooted in earlier work on composition filters [5, 6], adaptive programming [7, 8] and subject-oriented programming [9]. The concepts are now moving beyond programming and being applied at earlier development stages. For instance, a number of aspect-oriented requirements engineering, analysis and design approaches have been proposed (see, for example, [10–16]).

The term *early aspects* [17] refers to the set of techniques focusing on separation of crosscutting concerns before the detailed design (and subsequently the implementation) is derived. It, therefore, encompasses techniques focusing on aspects at the requirements engineering and architecture design stages as well as mechanisms to model, analyse and compose such aspects. Furthermore, the traceability of aspects modularised at this early stage to later artefacts (at the design and implementation level) is also an important area of consideration. Three workshops, one at each of the AOSD conferences so far, have explored a number of these issues. A fourth workshop is due to be held in conjunction with the Object-Oriented Programming Languages, Systems and Applications Conference (OOPSLA) in October 2004. The upcoming International Conference on Requirements Engineering in September 2004 has a paper session dedicated to aspect-oriented requirements engineering.

This Special Issue of IEE Proceedings Software includes four very interesting papers focusing on a range of key topics in the early aspects space namely, aspect-oriented requirements engineering, aspects at the architecture and early design stages, and feature composition and architecture variability using aspect-orientation.

Whittle and Araújo discuss that interaction of crosscutting requirements with other requirements must be understood from the very early stages of software development. Otherwise such interactions are discovered later on in the development life cycle leading to increased development costs arising from rectifying actions that must be taken. They argue that the solution lies in aspect-oriented requirements engineering and propose explicit modelling of crosscutting requirements as well as their composition with other requirements so that the whole requirements set can be validated. They focus on a specific requirements modelling mechanism, scenario-based modelling. Crosscutting scenarios are separated as aspectual scenarios and represented using interaction pattern specifications, which are UML sequence diagrams extended with generic role elements. Non-aspectual scenarios are represented as UML sequence diagrams. The two are composed together to produce state machines that can be executed to understand the composed behaviour. The composition mechanism and the resultant state machines are key features of the paper as they make it possible to identify erroneous requirements through analysis of the behaviour of the composed specification. The paper also includes a detailed case study applying the authors' approach to a sub-system of a tool from the air traffic control domain.

France, Ray, Georg and Ghosh introduce an aspect-oriented modelling approach from which they produce a composed aspect-oriented architecture model. Their approach is characterised by the existence of a base architecture which they call the primary model and a set of aspect models. The primary model is described using UML diagrams, and, in this paper, the authors use classifier and interaction diagrams. Aspect models, on the other hand, represent generalised solutions for crosscutting concerns and are specified as patterns. Patterns are described using UML model templates which must be instantiated before they can be composed with the primary model. The instantiation is accomplished by binding the template parameters to application-specific values. Finally, the aspect models are composed with the primary model by following a set of composition directives. A composition directive may specify, for example, the order in which several aspect models are composed with the primary model. The aspect-oriented architecture is, therefore, composed of a primary model, aspect models and the bindings needed to instantiate them to an application-specific context, and a set of composition directives that specify how the instantiated aspect models are composed with the primary model. The approach finishes with an analysis model that takes the resulting composed aspect-oriented architecture model and identifies conflicts that may arise during composition. The authors also show how conflicts can be resolved using composition directives.

These two first papers share the idea of employing specification patterns, described using UML model templates, to represent aspects. (This is also proposed in Clarke's work [18].) The major difference in the way the two papers use patterns is that while Whittle and Araújo allow concrete and role elements to coexist in a UML model template, France *et al.* impose that all model template elements must be generic. Moreover, the composition process proposed by France *et al.* is subsequent to the instantiation process and the result is still the same type of UML diagram, whereas in Whittle and Araújo's work the instantiation process generates a set of sequence diagrams which are then merged together into a set of state machines by using a synthesis algorithm.

The last two papers, on the other hand, focus on feature modelling and variability. While Jansen *et al.* focus on making features first class abstractions in software product families, Pratap *et al.* devote their attention to providing variability in middleware platforms.

Jansen, Smedinga, van Gurp and Bosch focus on product development in software product families in their paper. A software product family (SPF) represents a family of related products that share common features and business goals, and that are developed from a common set of core assets in a prescribed way. Software product family development primarily aims at systematic reuse and likewise improvements in productivity, time to market, and product quality. A key challenge in SPFs is to define the commonality and variability for the set of products that are required to develop the required products. Products in this context are essentially defined as a set of features. The authors claim that features are examples of early aspects because they have to be identified early on during the commonality and variability analysis, and are scattered over various products. Unfortunately, current SPF approaches do not adopt features as first class representations. Changing or adding features to the SPF, therefore, implies a tedious and error-prone process and seriously complicates product derivation. To solve the problem, the authors propose a role-based first-class representation of features, thereby explicitly separating features from products. To provide the 'weaving' of the features and as such to derive the products the authors describe a composition algorithm and elaborate on the possible composition problems.

Pratap, Hunleth and Cytron highlight the customisability problem in middleware platforms. They argue that existing customisation approaches such as macros and strategy and template method patterns have a number of shortcomings and propose an alternative approach to developing customisable middleware. The proposed framework involves the use of aspects to introduce features in an incremental fashion. Most interestingly, however, the authors highlight a number of architectural considerations in the development of their framework. Firstly, they discuss the issue of automatic validation of feature combinations and propose the use of a feature registry to maintain all relationships and meta-data pertaining to each feature. Secondly, they take testing into account within the architecture of their framework and use aspects to automatically run and upgrade tests relating to the current feature set in the system. Lastly, they discuss and present a number of aspect-oriented design patterns identified during the development of their framework. The paper also includes some experimental results comparing the authors' approach with traditional middleware approaches. These measurements demonstrate that key architectural considerations based on aspect-oriented development concepts can go a long way towards improving the overall footprint and performance as features can be selectively enabled or disabled depending on the specific requirements of the application.

This Special Issue would not have been possible without the time and effort devoted by the various reviewers: Uwe Assman, Elisa Baniassad, Gordon Blair, Marcelo Campo, Siobhán Clarke, Paul Clements, Fernando Figueroa, Lidia Fuentes, Jeff Gray, John Grundy, Charles Haley, Juan Hernandez, Shmuel Katz, Robin Laney, Mira Mezini, Gail Murphy, Bashar Nuseibeh, Peter Sawyer, Ian Sommerville and Stanley Sutton Jr. We also wish to thank Lee Baldwin, Stuart Govan and Shirley Rossall at IEE for their help throughout the preparation of this Special Issue. Last, but not least, we would like to thank all the authors who submitted papers to the Special Issue.

Although the papers in this Special Issue provide an interesting insight into the topic of early aspects and address some important research issues pertaining to early aspect composition, there are a number of other key research issues outstanding that need to be addressed. One of these is traceability. Aspects at earlier development stages do not necessarily map on to aspects at the design and implementation level while new aspects might arise during these later development stages. It is important that one clearly understands how early aspects filter down the development stages and how one can verify that the resultant system satisfies and preserves the properties specified by early aspects. Another key area is that of early aspect identification. What makes a good early aspect and how does one identify these early aspects in the plethora of requirements documentation available, e.g., interviews with stakeholders, users, ethnographic studies, etc? There is also a need to clarify the relationship between early aspects research and existing research in requirements engineering and architecture design. Some of these techniques, for example, goal-oriented approaches [19] and architectural styles [20], also attempt to tackle the problem of crosscutting concerns. It is, therefore, important to understand what lessons early aspects techniques can learn from these approaches and what interesting ideas can be injected into these approaches from the work on early aspects. The above, non-exhaustive, list of outstanding research issues shows that this Special Issue has just scratched the surface. We, therefore, hope that a number of similarly exciting approaches addressing the issues above will appear over the next few years. Whatever shape or form those approaches take, it is clear that early aspects tackle a very important problem at the requirements engineering and architecture design stages and can provide improved support for requirements engineers and architects to reason about their specifications and architectures.

AWAIS RASHID
ANA MOREIRA
BEDIR TEKINERDOGAN

IEE Proceedings online no. 20041027
doi: 10.1049/ip-sen:20041027

References

1 AOSD, 'Aspect-oriented software development', http://aosd.net, 2004
2 AspectJ Team, 'AspectJ Project', http://www.eclipse.org/aspectj/, 2004
3 Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M.A., Palm, J., and Griswold, W.G.: 'An Overview of AspectJ', *Lect. Notes Comput. Sci.*, 2001, **2072**, pp. 327–353
4 Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J., and Irwin, J.: 'Aspect-oriented programming', *Lect. Notes Comput. Sci.*, 1997, **1241**, pp. 220–242
5 Aksit, M., Bergmans, L., and Vural, S.: 'An object-oriented language-database integration model: the composition-filters approach', *Lect. Notes Comput. Sci.*, 1992, **615**, pp. 372–395

6 Bergmans, L., and Aksit, M.: 'Composing crosscutting concerns using composition filters', *Commun. ACM*, 2001, **44**, (10), pp. 51–57
7 Lieberherr, K.J., Orleans, D., and Ovlinger, J.: 'Aspect-oriented programming with adaptive methods', *Commun. ACM*, 2001, **44**, (10), pp. 39–41
8 Lieberherr, K.J., Silva-Lepe, I., and Xiao, C.: 'Adaptive object-oriented programming using graph-based customization', *Commun. ACM*, 1994, **37**, (5), pp. 94–101
9 Harrison, W.H., and Ossher, H.: 'Subject-oriented programming (a critique of pure objects)'. Proc. ACM SIGPLAN Conf. on Object-oriented Programming, Languages, Systems and Applications (OOPSLA), 1993, ACM, SIGPLAN Notices, **28**, (10), pp. 411–428
10 Baniassad, E.L.A., and Clarke, S.: 'Theme: an approach for aspect-oriented analysis and design'. Proc. Int. Conf. on Software Engineering (ICSE), 2004, pp. 158–167
11 Clarke, S., and Walker, R.J.: 'Towards a standard design language for AOSD'. Proc. 1st ACM Conf. on Aspect-oriented software Development, 2002, pp. 113–119
12 Haley, C., Laney, R., and Nuseibeh, B.: 'Deriving security requirements from crosscutting threat descriptions'. Proc. 3rd ACM Int. Conf. on Aspect-oriented Software Development (AOSD), 2004, pp. 112–121
13 Katara, M., and Katz, S.: 'Architectural views of aspects'. Proc. 2nd ACM Int. Conf. on Aspect-oriented Software Development, 2003, pp. 1–10
14 Moreira, A., Araújo, J., and Brito, I.: 'Crosscutting quality attributes for requirements engineering'. Proc. 14th ACM Int. Conf. on Software Engineering and Knowledge Engineering (SEKE), 2002, pp. 167–174
15 Rashid, A., Moreira, A., and Araujo, J.: 'Modularisation and composition of aspectual requirements'. Proc. 2nd ACM Int. Conf. on Aspect-oriented Software Development, 2003, pp. 11–20
16 Sutton, S.M., and Rouvellou, I.: 'Modeling of software concerns in cosmos'. Proc. ACM Int. Conf. on Aspect-oriented Software Development, 2002, pp. 127–133
17 EarlyAspects, 'Early aspects: aspect-oriented requirements engineering and architecture design', http://early-aspects.net, 2004
18 Clarke, S., and Walker, R.J.: 'Composition patterns: an approach to designing reusable aspects'. Proc. Int. Conf. on Software Engineering (ICSE), 2001, pp. 5–14
19 Lamsweerde, A.: 'Goal-oriented requirements engineering: a guided tour'. Proc. 5th Int. Symp. on Requirements Engineering, 2001, pp. 249–261
20 Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M.: 'Pattern-oriented software architecture: a system of patterns' (Wiley, 1996)

Awais Rashid is a senior lecturer in the Computing Department at Lancaster University in the United Kingdom. His principal research interests are in aspect-oriented software engineering, aspect-oriented databases and object data management. He is particularly interested in composition and traceability issues in aspect-oriented requirements engineering, combining different aspect-oriented approaches during system development and application of aspect-oriented techniques in databases, software product lines and safety critical systems. He has published actively on these topics. He is coordinating the European Network of Excellence on Aspect-Oriented Software Development and is the co-editor-in-chief of Transactions on Aspect-Oriented Software Development.

Ana Moreira is an assistant professor in the Computer Science Department at Universidade Nova Lisboa, Portugal. Her main research areas are aspect-oriented software development, object technology, requirements engineering, and formal description techniques. Currently she is interested in investigating how aspect-orientation can be used during the early activities of the software development process. She has been actively involved in several scientific events in her topics of interest. She is a member of the editorial board for the Springer-Verlag journals 'Software and Systems Modeling' and also the upcoming 'Transactions on Aspect-Oriented Software Development'.

Bedir Tekinerdogan is an assistant professor at University of Twente in the Department of Computer Science. His current research is on aspect-oriented software architecture design, quantitative evaluation of software architectures, multidimensional separation of concerns using design space modelling and aspect-oriented domain analysis. He has served on the program and organising committees of several workshops on the topics of aspect-oriented software development.