

A Multi-Dimensional, Model-Driven Approach to Concern Identification and Traceability

Awais Rashid¹, Ana Moreira², Joao Araujo², Peter Sawyer¹, Américo Sampaio¹

¹Computing Department, Infolab21, Lancaster University, Lancaster LA1 4WA, UK

²Dept. Informática, FCT, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal

1. Introduction

The *separation of concerns* principle [3], i.e., modularising concerns so that they may be realised and reasoned about in isolation, is a fundamental principle in software engineering. Recent years have seen increasing interest in aspect-oriented software development (AOSD) techniques [1, 4]. These focus on treatment of crosscutting concerns, i.e., concerns of a broadly scoped nature such as security, distribution and dependability. Multi-dimensional separation of concerns approaches [5, 6, 10], on the other hand, do not make a strong distinction between crosscutting and non-crosscutting concerns in a system and instead treat all concerns symmetrically. The notion of multi-dimensionality thus supports AOSD as one can potentially choose any arbitrary set of concerns as a base on which to observe the crosscutting influence of another set of concerns [6]. Whatever the chosen approach for separation of concerns or its degree of dimensionality, concern identification remains a difficult and challenging task. Similarly, tracing concerns through their refinement and mapping across various software development stages also poses significant challenges. In this paper, we outline an approach for concern identification and traceability. The approach employs model-driven techniques to map a problem description to recurring concern patterns as well as maintain the traceability of the concerns thus identified through to implementation.

We have based our concern identification and traceability approach on the multi-dimensional concern model for requirements engineering we presented in [6]. In this model, the concern space at the requirements-level is perceived as a hypercube (cf. figure 1). A concern is a coherent collection of requirements. All concerns are treated uniformly and no base-aspect distinctions are made though the model supports compositional projections of concerns on other concerns (block arrows in figure 1) to identify their influences hence catering for crosscutting effects.

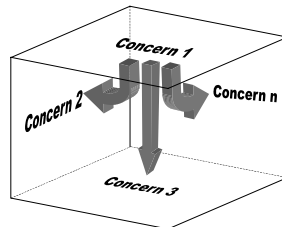


Figure 1. Concerns in a system represented as a hypercube

2. Concern Patterns in the Problem Space

Concern identification begins very early on in the development lifecycle, i.e., during requirements elicitation and subsequent analysis. However, concern identification at this stage is hindered by the problems of synthesising a coherent baseline of requirements from sources such as interview transcripts. These are frequently imprecise, full of apparent contradictions and missing essential information. Furthermore, the number of concerns to be identified in a system can be potentially very large [9]. Recent work in AOSD has employed natural language processing (NLP) techniques to identify concerns in a system [7, 8]. Here, we take an alternative concern identification approach based on the observation that certain concerns appear time and again during system development [6]. A catalogue of non-functional concerns has been provided by [2] but in [6] we extended the notion of such a catalogue to typical functional concerns. Examples of such repeatedly appearing functional and non-functional concerns include registration, ordering, billing, booking, mobility, availability and security. Some of these recurring concerns may span domains but most are likely to be domain-specific. Registration is an example of such a domain-specific concern. The semantics of registration would be similar for the domain of on-line shops but differ from registration of aircrafts for the aviation domain.

These recurring concerns can be observed as patterns of requirements and mined for (note this can be done using NLP techniques as mentioned above) and documented as abstract concern models. This results in a repository of concern models. When identifying concerns in requirements for a new

application (or reengineering an application for that matter), one can mine for concrete instances of the abstract concern models from the repository and automatically categorise relevant requirements into suitable concerns. One can also observe additional patterns and add further abstract concern models to the repository. This is shown in figure 2.

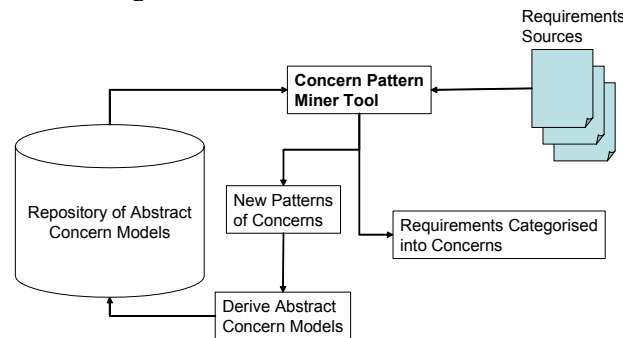


Figure 2: Mining for Recurring Concern Patterns in Requirements Sources

3. Mapping Concern Patterns to Application Space and Implementation

The mapping from the abstract concern models to their concrete occurrences in the application space can be perceived as model transformation. The Concern Pattern Miner tool (cf. figure 2) acts as the model transformer converting abstract concern models to more concrete ones. At the same time, the tool contributes to the extension of the repository hence supporting the creation of new reusable assets (as in product line engineering).

As shown in figure 3, we take a similar model driven approach to maintaining the traceability of the concrete concern models thus obtained. A composition model describes how a set of concrete concern models influences or constrains another set of concern models. As mentioned earlier, this is realised via projections of the former on the latter ([6] presents the notion of *compositional intersection* to limit the number of such projections to the essential ones). The concrete concern models and the composition model are used as input to the Solution Space Mapping Tool which maps the concrete concern models to corresponding architectural choices. This mapping is based on the information and interactions derived from the composition model as well as domain knowledge about the architectural choices driven by a particular abstract concern model. At the same time, it maps the interactions and trade-offs (finalised after negotiation with the stakeholders) to guard conditions which are subsequently reified to act as constraints over the implementation both statically and at runtime. This ensures that the interactions among concerns and trade-offs are preserved by the operational system.

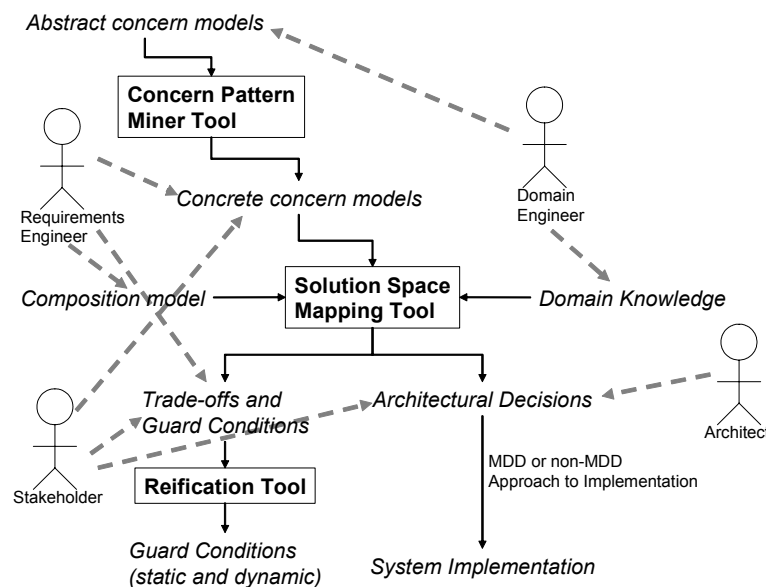


Figure 3: Model-Driven Approach for Concern Traceability

It is worth noting that the very nature of the concerns at the requirements-level means that the process cannot be fully automated. Instead it has to be driven by the skills and knowledge of the engineer at

several points (as shown by the actor roles¹ and dashed grey arrows in figure 3). This is essential and highly beneficial as it ensures that the needs and viewpoints of the stakeholders are reflected effectively in the trade-offs established as well as the architectural choices made. Further more, some steps need to be driven by the requirements engineers and architects or involve input from and negotiations with the stakeholders.

4. Conclusion

We have introduced a model driven approach to requirements engineering whereby abstract concern models can be transformed to their concrete manifestations in a given problem space. The concrete concern models can in turn be transformed into architectural decisions and guard conditions governing the system implementation. One of the key contributions of the approach is the inherent support for traceability. The various transformation tools are intended to be open in that they can be queried for the *path* from the input models to the output models. This makes it possible for the requirements engineer to trace the relevant abstract concern models from the repository through the various transformations. One of the key questions that we need to address is that of model categorisation. Is it essential to categorise our models as Computation Independent Models (CIMs), Platform Independent Models (PIMs) and Platform Specific Models (PSMs)? Or is it more pragmatic to have an approach where the boundaries between model categories are less rigid and models can exist across boundaries. This is an interesting notion to explore especially in COTS-based development where platform-specific issues often need to be taken into account during requirements engineering. We aim to address this question in our future work as well as continue the implementation of the various transformation tools mentioned in this paper.

Acknowledgements: This work is supported by EPSRC Grant MULDRE (EP/C003330/1) and Portuguese FCT Grant SOFTAS (POSI/EIA/60189/2004).

References

- [1] AOSD, "Aspect-Oriented Software Development", <http://aosd.net>, 2005.
- [2] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*: Kluwer, 2000.
- [3] E. W. Dijkstra, *A Discipline of Programming*. Englewood Cliffs, NJ: Prentice Hall, 1976.
- [4] R. Filman, T. Elrad, S. Clarke, and M. Aksit eds., "Aspect-Oriented Software Development": Addison-Wesley, 2004.
- [5] A. Moreira, J. Araujo, and A. Rashid, "A Concern-Oriented Requirements Engineering Model", International Conference on Advanced Information Systems Engineering (CAiSE), 2005 (Accepted to Appear).
- [6] A. Moreira, A. Rashid, and J. Araujo, "Multi-Dimensional Separation of Concerns in Requirements Engineering", International Conference on Requirements Engineering (RE), 2005, IEEE Computer Society (Accepted to Appear).
- [7] A. Sampaio, N. Loughran, A. Rashid, and P. Rayson, "Mining Aspects in Requirements", Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, held in conjunction with AOSD Conference, 2005.
- [8] A. Sampaio, A. Rashid, and P. Rayson, "Early-AIM: An Approach for Identifying Aspects in Requirements", International Conference on Requirements Engineering (RE), 2005, IEEE Computer Society (Accepted to Appear).
- [9] S. M. Sutton, "Concerns in a Requirements Model - a Small Case Study", Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, held in conjunction with AOSD Conference, 2003.
- [10] P. L. Tarr, H. Ossher, W. H. Harrison, and S. M. Sutton, "N Degrees of Separation: Multi-Dimensional Separation of Concerns", International Conference on Software Engineering (ICSE), 1999, ACM, pp. 107-119.

¹ Note that the same actor role can be played by multiple persons or the same person can play multiple actor roles.