Patterns in Model Engineering 2015 – a Workshop Summary

Richard F. Paige[1], Eugene Syriani[2], Huseyin Ergin[3] and Steffen Zschaler[4]

[1] Department of Computer Science, University of York, UK.
richard.paige_at_york.ac.uk
[2] Department of Computer Science and Operations Research, University of Montreal, Canada.
syriani@iro.umontreal.ca
[3] Department of Computer Science, University of Alabama, USA.
hergin@crimson.ua.edu
[4] Department of Informatics, Kings College London, UK.
steffen.zschaler@kcl.ac.uk

Abstract. The *Patterns in Model Engineering* (PAME) workshop[5] was held on 21 July 2015 as part of the *Software Technologies: Applications and Foundations* (STAF) conference, in L'Aquila, Italy. The workshop focused on identification, analysis and presentation of *patterns* across all aspects of modelling and Model-Driven Engineering (MDE), including patterns for modelling, metamodelling, transformation, and in constraints. The workshop featured three invited presentations by Jordi Cabot (ICREA, Spain), Daniel Varro (BME, Hungary) and Antonio Cicchetti (MDH, Sweden), five full papers, and a significant discussion and debate about the roles that patterns can play in modelling. This paper summarises the workshop discussion and highlights some of the key research challenges in the field.

1 Introduction

The field of Model-Driven Engineering (MDE) has evolved tremendously since its first steps 2002. Current development activities in MDE span very different techniques such as metamodeling, model transformation, domain-specific modeling, model evolution, model verification and validation. Various languages and tools have been proposed to implement the artifacts produced by these activities. Having been applied in a variety of different application domains, good practices and idioms have been proposed for some of these languages in order to improve the quality of modeling artifacts produced. Some language-specific idioms can be re-used and applied across different languages, to then become generalized to design patterns. Although some preliminary work on design patterns in modelling and MDE[6] [1] has been proposed, this is an area of MDE that still needs

[5] http://www-ens.iro.umontreal.ca/~syriani/pame2015/index.html
[6] https://people.irisa.fr/Jean-Marc.Jezequel/enseignement/DPApplicationWithMDE.pdf

to be thoroughly investigated. With the growing maturity of MDE, it is high time to further investigate the discovery, definition, purpose and application of design patterns in model engineering activities.

The *PAME'15* workshop co-located with *STAF'15* in L'Aquila, Italy, was intended to be the first forum for practitioners in MDE to discuss the patterns that occur often during the different modeling activities. The aim of the workshop was to provide an arena for proposing and discussing good practices, patterns, pattern-based modeling, as well as the start of an initiative to develop a vocabulary and parts of a common structure for discussing and describing relevant problems and their solutions in modelling and MDE in the form of patterns.

This paper constitutes a summary of the workshop presentations as well as a brief overview of the discussion that took place in the second half of the workshop.

The workshop was organised in an unusual distributed fashion: one organiser was physically present, while the other three organisers connected via Skype, and actively engaged in both the discussion and the presentations.

This summary commences with a brief overview of the program and then the presentations given; each presentation is outlined in the order in which it was given. Some of the key questions that arose in the discussions for each presentation are highlighted. A short summary of each invited talk is also presented. The final section outlines the key general questions and research challenges that were identified.

2 Program and Presentation Summary

The program for *PAME'15* is given below.

- 11:00-11:10 Intro
- 11:10-12:30 Paper presentations:
  - Adolfo Sánchez-Barbudo Herrera: *Auto-generation of model visitor frameworks*
  - Vadim Zaytsev: *Two-Faced Data*
  - Sahar Kokaly, Zinovy Diskin, Tom Maibaum and Hamid Gholizadeh: *Elementary Model Management Patterns*
  - Kevin Lano and Sobhan Yassipour Tehrani: *Design Patterns for Model Transformations: Current research and future directions*
  - Huseyin Ergin and Eugene Syriani: *A Unified Template for Model Transformation Design Patterns*
- 12:30-14:00 Lunch
- 14:00-15:30 Invited talks:
  - Daniel Varro: *Patterns and styles for incremental model transformations*
  - Antonio Cicchetti: *On the need for a bottom-up approach to metamodelling*
  - Jordi Cabot: *Who needs languages when you have patterns?*
- 15:30-16:00 Coffee break
- 16:00-18:00 Discussion

2.1 Auto-generation of model visitor frameworks

Adolfo presented an example of using a pattern in MDE; he talked about uses for a Visitor pattern in building tools for MDE (e.g., constraint languages, transformation languages) and how the need for something like a Visitor has arisen in his doctoral research. He then presented a framework to support automatic generation of visitor frameworks for EMF. The motivation is that while it appears straightforward to apply the visitor pattern to models in EMF, it turns to be as tedious as performing it in a traditional programming language. For example, every *X EClass* requires an *accept EOperation*; a visitor *EClass* requires a *visitX EOperation* for every *X EClass*. If the number of involved metamodels grows large, the shortcomings associated with the visitor pattern hinder its adoption.

A question that was raised in discussion is whether the example Adolfo presented was indeed an MDE pattern; the question was whether the automatic application of visitors to EMF could be better solved through MDE language support, e.g., Rascal or Stratego, which provide automated rule scheduling.

2.2 Two-faced data

Vadim presented a particular modelling pattern, *two-faced data*. The motivation for this pattern comes from the challenge of linking a conceptual model and the underlying domain: ideally, the relationship between a conceptual model (e.g., the grammar or metamodel that engineers will use and implement) and the underlying domain should be bijective. If you make your grammar/metamodel too close to the desired conceptual representation, you risk making it ambiguous, inefficient for parsing and/or not user friendly for the language users. If you make it too close to the desired way of writing and reading sentences in the language, you risk overburdening your model traversals with unnecessary details concerning a particular textual representation.

Questions that arose in the discussion related to the scope of the two-faced data pattern, e.g., can two different misinterpretations of data be counted as two-faced?

2.3 Elementary Model Management Patterns

Zinovy presented basic building-block patterns for model management, from which more complex and expressive model management patterns could be constructed. He presented a summary of a library of these basic design patterns, which focus on making mappings between model elements explicit, and which are defined as sets of links between model elements rather than single links between models. Moreover, the algebraic operations in the patterns generate models as well as explicit traceability maps. He presented a selection of static and dynamic patterns (including several for model merge).

The talk was somewhat contentious, as Zinovy made the statement "Patterns are only good if they are formal". There was some debate on this; Steffen Zschaler

questioned this, stating that a lot of the usefulness of the Gang of Four patterns derives precisely from the fact that they aren't formal. Another question focused on the nature of a pattern in Zinovy's library: the talk seemed to argue that patterns are hierarchical encapsulations of model management behaviour (e.g., analogous to procedures in a programming language).

2.4 Design Patterns for Model Transformations: Current research and future directions

Sobhan presented his paper, which focuses on a survey of patterns in the model transformation literature. He identified a number of categories of relevant patterns: Rule modularisation patterns (e.g., Phased Construction; Structure Preservation); Optimisation patterns (e.g., Unique Instantiation; Object Indexing); Model-to-text patterns (e.g., Model Visitor; Text Templates); Expressiveness patterns (e.g., Simulating Multiple Matching) and Architectural patterns (e.g., Filter before Processing).

The audience thought that this was a useful contribution, and found it interesting that many of the patterns identified have already found their way into existing model transformation languages. There was discussion as to what point a pattern becomes a language feature. Also, there were several questions about validating or evaluating the usefulness of said patterns.

2.5 A Unified Template for Model Transformation Design Patterns

In the last non-invited peer reviewed talk, Huseyin (over Skype) presented a proposal for a unified template to describe model transformation design patterns; the proposal was based on the analysis of existing model transformation design pattern studies. He also presented an example that instantiates the pattern, using the *top-down phased construction* design pattern.

Questions were asked about how to judge the effectiveness and utility of a design pattern template in MDE, for example, should it favour formality (e.g., should there be a metamodel?) over understandability. Questions about stress-testing the template also arose, e.g., could it be used to express negative patterns (such as NACs)?

2.6 Invited Talk: Patterns and styles for incremental model transformations

Daniel Varro gave his invited talk on patterns and styles, particularly focusing on incremental model transformations, in which patterns play a significant role (e.g., for specifying and identifying that part of the model that has changed, to which the incremental transformation must be applied).

There was substantial discussion on the distinction between a pattern and a style. One view was that styles were more coarse-grained patterns; a question about whether there was an important distinction in the context of MDE led to

numerous follow-up questions. Another related question was whether styles were just classifications of different approaches to a problem (e.g., less formal than patterns). In this sense, Daniel's talk presented a number of different styles for incremental transformation; this is somewhat different from the notion of style in software architecture.

2.7 Invited Talk: On the need for a bottom-up approach to metamodelling

Antonio Cicchetti gave his invited talk focusing on bottom-up approaches to metamodelling, i.e., using examples and lightweight customisation approaches to build metamodels. Antonio's argument was based on the observation that no-one uses modelling languages (and hence metamodels) as they are defined off-the-shelf; companies want customisations for languages (otherwise they may abuse them). Patterns for languages and transformation development can help provide the necessary flexibility so that language customisation is easier to make.

There were questions about who the intended users of metamodelling patterns would be: the original language designer, a language engineer, a software engineer? There were also questions about the most appropriate ways to define metamodel design patterns – how would they appear? How would they be distinguished from metamodels?

2.8 Invited Talk: Who needs languages when you have patterns?

Finally, Jordi Cabot gave his invited talk on the importance of patterns over languages. He argued for mechanisms to observe what patterns users of modelling languages apply (e.g., when creating models or transformations), how they are applied (e.g., manually, automatically, to specific parts of a model) and when they are applied. All of this information can potentially help improve modelling languages.

This concluded the formal presentations for the workshop, and after a short refreshment break, we moved to the discussion phase.

3 Discussion

To kick off the discussion, the organisers collected a set of provocative questions, some or all of which would be considered over the rest of the afternoon. The collected questions, proposed by both the audience (and authors not present) and the organisers, are as follows.

- Why should design patterns in model engineering be useful? Do we find patterns because we have to?
- Claim: design patterns are "workarounds for language omissions". If this is true, do we need them in MDE given that we can easily fix the language? Are they perhaps better used to indicate the need to improve our language, i.e., a *language bad smell*?

- Why use visitors instead of using a model transformation language? More generally, why use "design pattern" as they are applied in programming in an MDE context?
- What are MDE-specific patterns? Must they be tied to a specific technology platform like EMF?
- Are patterns only good if theyre formal? Or, to ask the question in a different way: Do we need pattern description languages specifically for MDE? If so, what should they look like?
- What are the analogies to design patterns, architectural styles, enterprise integration patterns, ... (if any) in model engineering? What other levels are there?
- Are there domain-specific patterns (and how do we find them)?
- How do patterns interoperate with domain-specific transformation languages (and how do we describe them there)?
- What are the obstacles to the application, discovery and use of patterns in model engineering?
- Where does a pattern language start and a systematic literature survey end?

The discussion session was wide ranging, touching on these and numerous other questions. One observation that was made several times was that patterns in model engineering were concepts that were increasingly turned into language features (over time). What was the process for this? How did a language engineer (effectively, the user of a pattern) decide how and when to convert ("promote") a language pattern into a full-fledged language feature? While there were rules of thumb (e.g., when we find ourselves writing the pattern in 20% of our model management code), the conclusion was that it has to be driven by the needs of language engineers and language users – and we are not yet very good at understanding these needs.

Another discussion focused around whether *domain analysis* is effectively the search for domain patterns (which could thereafter be encoded as language patterns). The view was that this was a too-strict interpretation of domain analysis, which also involves understanding vocabulary/concepts, pattern composition, and classification (though searching for domain patterns is an important part).

A third general point for discussion considered whether patterns in modelling and MDE played the same role as in programming. The reason this question came up is that, at least in theory, metamodels are easier to change than programming languages, and hence an application of a pattern could easily be a change to a metamodel or addition of language feature. One point raised in the discussion was that if patterns are language-specific then we will need libraries of patterns for different (families of) languages, e.g., OO, dynamically typed languages, etc. In essence the language itself would be the subject of a domain analysis. A point on which there was some consensus was that there is value and purpose in capturing knowledge of the community in the form of patterns. So even if patterns in modelling are transformed to language features quickly, expressing said patterns *outside of a language context* may provide positive (and negative!) value.

Finally, there was a discussion on whether patterns must be formal – i.e., do we need pattern languages specifically for MDE. There was no consensus here but a general view was that formality (e.g., via templates or formal semantics or metamodels) was not always useful – what was needed was precision in definition of scope and application of patterns.

There was, at the end, a consensus call for more examples of patterns in modelling, and a view that a wiki collecting such patterns would be useful.

4 Conclusions

The workshop was judged to be very successful by all 20-odd participants, and there was a clear desire to run a similar workshop, focusing more on collating and documenting interesting patterns, at a future event. To this end, the organisers have proposed to run *PAME'16* either at a future STAF conference, or co-located with MoDELS.

References

1. Zinovy Diskin and T. S. E. Maibaum. Category theory and model-driven engineering: From formal semantics to design patterns and beyond. In *Proceedings Seventh ACCAT Workshop on Applied and Computational Category Theory, ACCAT 2012, Tallinn, Estonia, 1 April 2012.*, pages 1–21, 2012.