Agile software development methodology, an ontological analysis

David Parsons
Massey University, Auckland, New Zealand
d.p.parsons@massey.ac.nz

Abstract

Agile methods have emerged in recent years as a new paradigm in software development, promising to free the process of building software systems from some of the constraints of more traditional approaches. However the plethora of overlapping methods makes it difficult to identify the core features of an agile approach that transcends any particular method and provide us with an overarching methodology. This paper takes an ontological approach to analyzing the core components of an agile methodology based on an analysis of existing literature related to agile ontology. The intent of this ontology is to assist our understanding of the kernel of software engineering theory that underlies agile methodology.
.

Keyword(s)*: agile methods, ontology, software engineering*

1. Introduction

In recent years we have seen a number of different approaches to bringing software development forward by applying both new technologies and new engineering and lifecycle management practices. In particular, we have seen the emergence of agile methods as a major new paradigm for the management of software development projects. In this paper, we take an ontological approach to an analysis of agile methods and attempt to identify an overarching ontology than may help us to understand how various agile practices may be successfully integrated within a broader agile methodology.

The mid 1990s saw the emergence of a number of informal analysis and design approaches that were later categorized as agile methods [1]. These methods place emphasis on being flexible to changes in requirements and working in collaboration with customers and other stakeholders. However there is evidence that teams adopting agile methods are unsure about the relationship between a given method and the set of techniques and processes that it may or may not involve [2]. This may be because there are a large number of agile methods, each specifying a particular set of techniques, both engineering and managerial, as opposed to supporting a more general understanding of what constitutes an agile methodology. Ivar Jacobson recently stated that we need a theory of software engineering built around a kernel of software development [3]. This work is somewhat less ambitious but addressing the same issue, stated by Jacobson as; "With the kernel in place all methods can be described in a uniform way, as specializations or extensions to this kernel."

It seems that representation of such a kernel should be done in a reasonably formalized way. To this end we utilize ontology as a means of providing this formalization. The basis of our proposal in this paper is therefore a general ontology-based representation for agile methods. This ontology is intended to provide an analytical model to represent the core relationships between the various components of agile methods. In this paper we consider previous work on ontologies for software development and introduce an ontology for agile methods, based on an analysis of a number of published agile methods.

2. Ontologies for software development

In attempting to provide a more formalized way of analyzing agile methods, one approach that may prove fruitful is to consider the use of ontologies as an analytical tool. Some of the literature suggests that agile methods have an ontology, though as yet this has not been formally published so is merely implied.

The benefits of an ontology include the ability to categorize the key components of the entities of interest and the relationships between them. Ontologies have been widely explored in software engineering (e.g. [4],[5],[6],[7]). There have also been a number of papers relating to application ontologies within specific agile projects. Mishali and Katz [8] explicitly refer to an ontology of XP in driving the architecture of their

Eclipse plug-in, though this ontology is not formally expressed. This plugin supports XP from the perspective of software process aspects. The aspects are seen as a way of implementing an ontology that is semantically congruent with the various practices of XP. The prototype targets certain practices, such as enforcing a test first policy.

Clearly there is a relevant body of work that may contribute to an understanding of agile method ontologies. So far, however, a more general ontology of agile methods has not been proposed. The motivation for this paper, therefore, is to propose some underpinnings for such an ontology and attempt to map it to subsets of existing software engineering ontologies.

It is important to specify why one is attempting to build, modify or apply an ontology and what kind of ontology is therefore required. Happel and Seedorf [7] categorise ontologies using two dimensions, one that distinguishes development time from run time, and another that differentiates software and infrastructure. In this paper we are concerning ourselves with development time infrastructure, so are focused on what Happel and Seedorf [7] call 'ontology-enabled development' which uses ontologies at development time to support developers with their tasks. In attempting to propose an ontology for agile infrastructure, we are therefore concerning ourselves with a subset of the possible ontologies that might apply to software engineering in general.

3. An ontology of agile methods

The relationship between ontologies and agile methods appears in the literature from time to time. For example Knublauch [9] suggests that ontology driven development should be more applied in the agile domain, and asserts that, with the correct tools, ontologies can be a powerful support for agile methods, in particular for generating test methods and supporting stakeholder involvement. Thus far, however, no single generic ontology of agile methods has been proposed. Therefore we have begun to propose such an ontology, based on an analysis of a number of commonly used agile methods. We took seven agile methods and attempted to summarize their terminology, illustrated with some key examples. Table 1 shows, for each of the seven methods, key terms used, along with indicative examples

The purpose of this exercise was to identify the commonality (or otherwise) of a representative number of agile methods to explore the viability of building an ontology that might apply across all agile methods. The purpose of the examples was to enable us to filter the

Agile Microsoft Solutions Framework	
Principles	Open communications, empowered team members, clear accountability and shared responsibility
Mindsets	Focus on Business Value, Teams of Peers, Internalize Qualities of Service
Agile UP	
Phases	Inception, elaboration, construction, transition
Disciplines	Model, implementation, test, project management
Philosophies	Simplicity, tool independence
Crystal Clear	
Properties	Frequent delivery of usable code, reflective improvement, osmotic communication
Strategies	Incremental Rearchitecture, Information Radiators.
Techniques	Daily Stand-up Meetings, Side-by-Side Programming, Burn Charts.
DSDM	
Principles	User involvement, empowered project team, frequent delivery of products, testing throughout the project life-cycle
Techniques	Timeboxing, MoSCoW, testing, workshop
eXtreme Programming (XP)	
Values	Communication, simplicity, feedback, courage, respect
Activities	Coding, testing, listening, designing
Techniques	Pair programming, test driven development, continuous integration, collective code ownership
Feature Driven Development	
Activities	Plan by feature, design by feature, build by feature
Best practices	Domain object modeling, develop by feature, individual code ownership, visibility of progress and results
Scrum	
Techniques	Team creation, backlog creation, project segmentation, scrum meetings, burn down charts
Phases	Review release plans, sprint, sprint review, closure

Table 1: Key concerns of agile methods

various terms used in the seven methods so that we could identify synonyms. This approach follows Happel and Seedorf [7], where their ontology classification is illustrated by exemplars. Where synonyms were identified, one term was chosen to subsume the others. Where possible, the chosen term was the most commonly used of the synonyms. The chosen terms were;

- Technique
- Phase
- Principle
  - Subsumes property, value
- Activity
  - Subsumes discipline
- Practice
  - Subsumes mindset, philosophy, strategy

In general it seems that an agile method will have some guiding set of principles that underpins its approach. It will also have high level activities, supported by management and engineering techniques. These will be organized under the umbrella of a set of practices. There may also be the concept of phases within the overall process. Within the detail of the various methods, the instantiation of techniques, for example, may vary widely. Engineering focused methods like eXtreme Programming (XP) will promote a specific set of techniques, whereas other methods, such as Scrum, do not concern themselves so much with engineering practices as with project management processes. Common ideas emerge from many methods, including testing, communication and visibility of progress. Incompatibilities are few and far between, with individual code ownership in Feature Driven Development being one of the few examples, contrasting with the common code ownership promoted by most other methods. This however has no impact on the overall ontology, since these are simply different instantiations of technique.

From this analysis we built an initial ontology of agile methods that attempts to encompass the various characteristics of commonly used methods. This ontology is shown in Figure 1. In our generic ontology for agile methods, a software system consists of a set of features built within activities that are part of a development process. That process will be guided by the principles of a particular method. Various techniques are used to carry out the activities (they will vary between methods) but these techniques will be either engineering or management oriented. The engineering techniques will include spatial considerations (co-location, pair programming etc.) and lingual issues (languages and tools). The management technique may address social issues such as active stakeholder involvement, sustainable pace and activities such as stand up meetings and retrospectives.

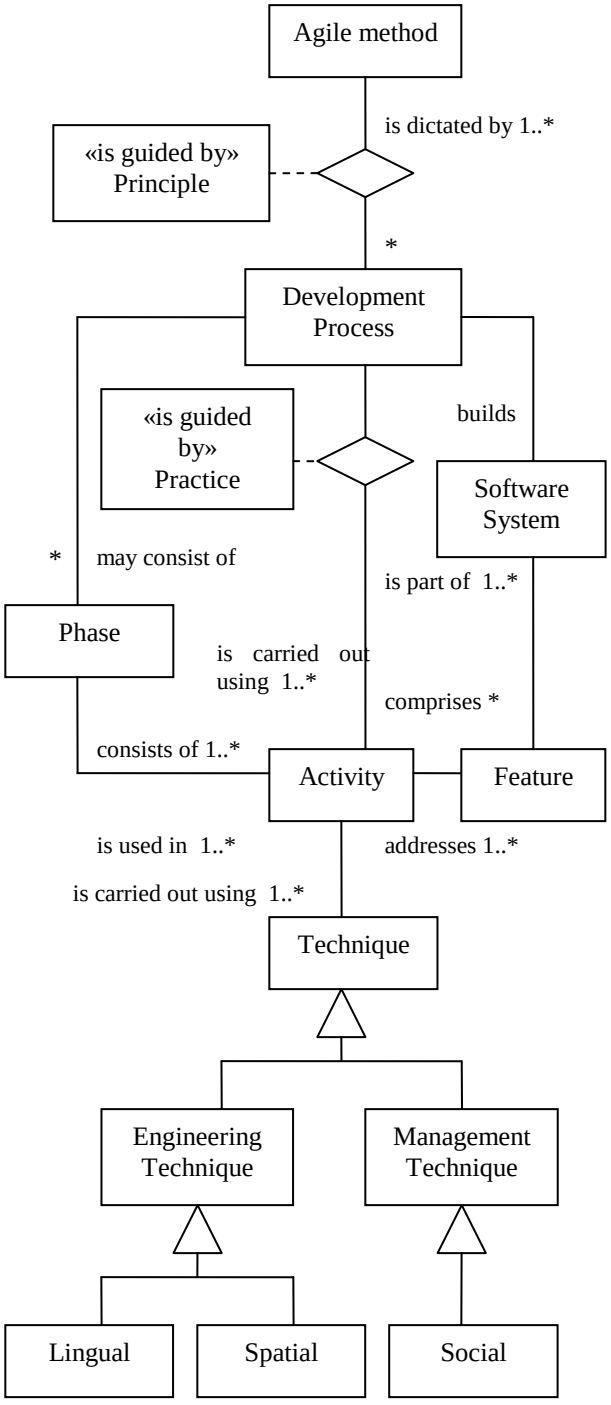

Figure 1: A generic ontology of agile methods

4. Summary and future work

In this paper we have described an ontology for agile methods to propose an analytical framework for understanding how an overarching agile methodology is constructed. This work is preliminary in nature and has yet to be exercised by empirical study. However it represents a first step in formalizing a kernel of agile software development that may assist us in ensuring that the relationships between agile practices and processes are properly understood by practitioners and may therefore be implemented in an effective way. The ways in which such an ontology may be used could include ontology mappings between a chosen method and the generic ontology, to indicate to what extent a given method encompasses the overall agile ontology, and the formalization of support tools for agile software development.

One of the practical aspects of an ontology is that it provides a formal specification that may be used in software tools. Therefore the value of creating a generic ontology for agile methods is that this ontology might be leveraged in supporting tools for agile software development. Tools such as Jena [10] and Protégé [11] give the opportunity to create ontologies in various representations such as the Resource Description Framework (RDF) or Web Ontology Language (OWL) that could enable the reasoning capabilities within these tools to be utilized in guiding the adoption of agile software development techniques. Future work will address this aspect of ontology, with the intention of creating such support tools.

References

[1] J. Highsmith, *Agile software development ecosystem*, Boston: Addison-Wesley, 2002.

[2] D. Parsons, H. Ryu and R. Lal, R. "The Impact of Methods and Techniques on Outcomes from Agile Software Development Projects" in IFIP 8.6 Conference: Organisational Dynamics of Technology-based Innovation: Diversifying the Research Agenda, McMaster, Wastell, Ferneley and DeGross (eds.), Springer, 2007, pp. 235-249.

[3] I. Jacobson, 2009 In need of a theory for software engineering, http://ivarblog.com/2009/05/29/in-need-of-a-theory-for-software-engineering/

[4] B. Marick, "Methodology Work Is Ontology Work", ACM SIGPLAN Notices, 39(12) pp. 64 – 72, 2004.

[5] P. Wongthongtham, E. Chang, T. Dillon and I. Sommerville, "Software engineering ontologies and their implementation", in Kobol, P. (ed), IASTED International Conference on Software Engineering (SE), pp. 208-213, Innsbruck, Austria, ACTA Publishing, 2005.

[6] M. Leppänen, *Towards an Ontology for Information Systems Development*, Via Nova Architectura, 2006.

[7] H. Happel and S. Seedorf, "Applications of Ontologies in Software Engineering", Proceedings of the 1st international conference on Theory and practice of electronic governance, Macao, China, pp. 5-11, 2007

[8] O. Mishali and S. Katz. "Using Aspects to Support the Software Process: XP over Eclipse". Proceedings of AOSD 06, Bonn, Germany, 2006.

[9] H. Knublauch. "Ramblings on Agile Methodologies and Ontology-Driven Software Development", Workshop on Semantic Web Enabled Software Engineering, Galway, Ireland, 2005.

[10] Jena – A Semantic Web Framework for Java http://jena.sourceforge.net/

[11] The Protége Ontology Editor and Knowledge Acquisition System, http://protege.stanford.edu/