IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 6, Ver. V (Nov.-Dec. 2016), PP 57-60
www.iosrjournals.org

A Feature- Driven Ontology for Agile FDD Software Development

Dr Farheen Siddiqui

Abstract: *Agile FDD methods have evolved and practiced in recent years as a new way in software development, leading to minimum dependency on the process of building software systems from some of the limitations of waterfall like approaches. However the excess of overlapping procedures makes it difficult to utilize important features of an agile FDD approach to facilitate any particular method and provide us with an overarching methodology. This paper takes an ontological approach to analyzing the core components of an feature driven development (FDD) based on an analysis of existing literature related to agile FDD ontology. The purpose of this ontology is to deepen the understanding of the most important part of software engineering theory that underlies agile FDD methodology.*

Keyword(s): *feature driven development, agile FDD methods, ontology, software engineering*

I. Introduction

In recent times number of different approaches came up bringing software development forward by applying both new technologies and new engineering and lifecycle management practices. In particular, agile FDD methods emerged 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 FDD methods and attempt to identify an overarching ontology than may help us to understand how various agile FDD practices may be successfully integrated within a broader agile FDD methodology. The mid 1990s saw the emergence of a number of informal analysis and design approaches that were later categorized as agile FDD 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 FDD 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 FDD 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 FDD 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 this paper utilizes 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 FDD methods. This ontology is intended to provide an analytical model to represent the core relationships between the various components of agile FDD methods. In this paper we consider previous work on ontologies for software development and introduce an ontology for feature-driven methods, based on an analysis of a number of published agile FDD methods.

II. Software development with ontologies

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 FDD 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 FDD method ontologies. So far, however, a more general ontology of agile FDD 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. This paper focus on FDD ontology, 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.

III. An ontology of agile FDD 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 FDD domain, and asserts that, with the correct tools, ontologies can be a powerful support for agile FDD methods, in particular for generating test methods and supporting stakeholder involvement. Thus far, however, no single generic ontology of agile FDD methods has been proposed.Therefore this paper intend to propose such an ontology, based on an analysis of a number of commonly used agile FDD methods. This paper targets concepts of agile FDD methods and attempted to summarize their terminology, illustrated with some key examples. The purpose of this exercise was to identify the commonality (or otherwise) of a representative number of agile FDD methods to explore the viability of building an ontology that might apply across all agile FDD methods. In general it seems that an agile FDD 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 initial ontology of agile FDD methods is built that attempts to encompass the various characteristics of commonly used methods. In our generic ontology for agile FDD 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. The following XML is owl rendering of developed ontology:

```
<?xml version="1.0"?>

<!DOCTYPE Ontology [
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.semanticweb.org/lenovo/ontologies/2016/11/FDD"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    ontologyIRI="http://www.semanticweb.org/lenovo/ontologies/2016/11/FDD">
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <Declaration>
      <Class IRI="#Action"/>
   </Declaration>
   <Declaration>
      <Class IRI="#Feature-list"/>
   </Declaration>
```

```
  <Declaration>
    <Class IRI="#Model"/>
  </Declaration>
  <Declaration>
    <Class IRI="#Object"/>
  </Declaration>
  <Declaration>
    <Class IRI="#Result"/>
  </Declaration>
  <Declaration>
    <Class IRI="#feature"/>
  </Declaration>
  <Declaration>
    <ObjectProperty IRI="#features"/>
  </Declaration>
  <SubClassOf>
    <Class IRI="#Action"/>
    <Class IRI="#feature"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Object"/>
    <Class IRI="#feature"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Result"/>
    <Class IRI="#feature"/>
  </SubClassOf>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#features"/>
    <Class IRI="#Feature-list"/>
  </ObjectPropertyDomain>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#features"/>
    <Class IRI="#feature"/>
  </ObjectPropertyRange>
</Ontology>
```

IV. Summary and Future Work

This paper describes an ontology for agile FDD methods to propose an analytical framework for understanding how an overarching agile FDD 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 FDD software development that may assist us in ensuring that the relationships between agile FDD 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 FDD ontology, and the formalization of support tools for agile FDD 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 FDD methods is that this ontology might be leveraged in supporting tools for agile FDD 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 FDD software development techniques. Future work will address this aspect of ontology, with the intention of creating such support tools.

References

[1] J. Highsmith, Agile FDD 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 FDD 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-theoryfor-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 FDD 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://p