Towards Knowledge Assisted Agile Requirements Evolution

Manish Kumar, Nirav Ajmeri
Tata Research, Development and Design Center (TRDDC)
A Division of Tata Consultancy Services
54 Hadapsar Industrial Estate Pune 411013, India
+91 20 6608 6441
{manish9.k, nirav.ajmeri}@tcs.com

Smita Ghaisas
Tata Research, Development and Design Center (TRDDC)
A Division of Tata Consultancy Services
54 Hadapsar Industrial Estate Pune
411013, India
+91 20 6608 6434
smita.ghaisas@tcs.com

ABSTRACT

This paper presents work on a recommendation system for Knowledge assisted Agile Requirements Evolution (K-gileRE). We treat requirements engineering as a special case of knowledge engineering and emphasize the fact that providing a domain knowledge edge can impart agility to the requirements definition exercise. The approach differs from existing agile methods in that it seamlessly incorporates a domain knowledge base into an agile requirements definition framework and explicitly provides to requirement analysts, relevant online domain specific recommendations based on underlying ontologies. The framework presents a 'domain knowledge seed' to requirement analysts. The seed provides a view of core features in a given domain and associated knowledge elements such as business processes, rules, policies, partial data models, use cases and test cases,. These in turn are mapped with agile requirements elements such as user stories, features, tasks, product backlog, sprints and prototype plans. The requirement analyst can evolve the seed to suit her specific project needs. As she modifies and evolves the seed specification, she receives domain-specific online recommendations to improve the correctness, consistency and completeness of her requirement specification documents and executable models. Using the domain knowledge seed as a point of departure provides a jump-start to her project. Each exercise of requirements definition thus becomes an evolution from the seed instead of the traditional 'clean slate' Requirements Engineering (RE) that typically starts from the scratch. Hence, the term K-gileRE. We elaborate how K-gileRE helps in practicing the essence of agile doctrines while defining software requirements by providing just-in-time recommendations.

Categories and Subject Descriptors

D.2.1 [**Software**]: Software Engineering – *Requirement specification,* D.2.13 [Reusable software]: Domain Engineering, D.2.13 [Reusable software]: Reuse models, H.3.5 **[Information Storage and Retrieval**]: Online Information Systems

General Terms

Management, Documentation, Design, Experimentation

Keywords

Domain-specific recommendations, Knowledge assisted Agile, Collaborative and semantic requirements definition,

1. INTRODUCTION

The agile movement has made a significant change of stance in terms of embracing pragmatic variants of their original recommendations. Industry veterans [1, 2] take cognizance of the need for agile to evolve and embrace ground realities of software developments. This has been due to the realization that adoption depends largely on how well a method (agile or otherwise) can support the real-life issues involved in software development. As a result, we see a lot of agile research and literature focusing on adapting the original agile doctrines to suit practical situations. 'Agile requirements' is one such point of focus [3] wherein we notice a change of stance from emphasis on an entirely code-driven development to a need to have in place at least a 'lightweight' requirements specification. We address this need by devising a Knowledge assisted Agile Requirements Evolution (K-gile RE) framework. The framework presents a 'domain knowledge seed' that can be evolved into a specification (document+ executable models). A requirement analyst who works on the seed uses the online domain-specific recommendations offered by K-gileRE as she evolves the seed to suit her project.

K-gileRE framework treats requirements engineering as a special case of knowledge engineering. It integrates four different knowledge contexts (Environmental, Generic requirements, Agile requirements and the Problem Domain) in the form of four ontologies. We employ mechanisms to specify semantic mappings of conclusions drawn from instance of one ontology to elements in other ontologies and provide recommendations based on the integrated inference. The online just-in-time recommendations help the requirement analyst in improving completeness, correctness and consistency of her specifications by providing an in-built and explicit domain knowledge value. The recommendations may be specific to a singular context or span the four knowledge contexts when necessary in response to actions of the requirement analyst. For example, if a requirement analyst selects some features from the domain seed and attempts to modify them in the context of her project, she would be presented with business rules, in the given geography e.g. 'Pension rules in Europe' (Environmental context and Problem domain context). If she selects features that complement each other but decides to associate them with different sprints, she would receive recommendations to rearrange them (Problem Domain context and Agile Requirements context). If she selects conflicting features in a given domain, she would be alerted about the inconsistency of her selection (singularly the Problem Domain context). We present examples to illustrate our approach and also discuss how it supports agile doctrines.

The paper continues into section 2 on illustration of the K-gileRE model.

2. The K-gileRE Model

The four ontologies in K-gile RE 'Environmental Context Ontology', 'Agile Requirements Ontology' and 'Problem Domain Ontology' are constructed using the grounded theory [4] and implemented using RDF-OWL schema [5]. Fig 1 shows partial example instances of the ontologies

2.1 Environmental Context Ontology

This ontology is designed to capture the environment in which software requirements are to be defined. For example, a requirement analyst may want to capture requirements for a Claims module of a Life Insurance application for a customer ABC Inc. in the Asia-Pacific geography. The concepts , ,'Actor', 'Action', 'Domain', 'LineofBusiness', 'Customer' and 'Geography', are abstractions used to capture the information.

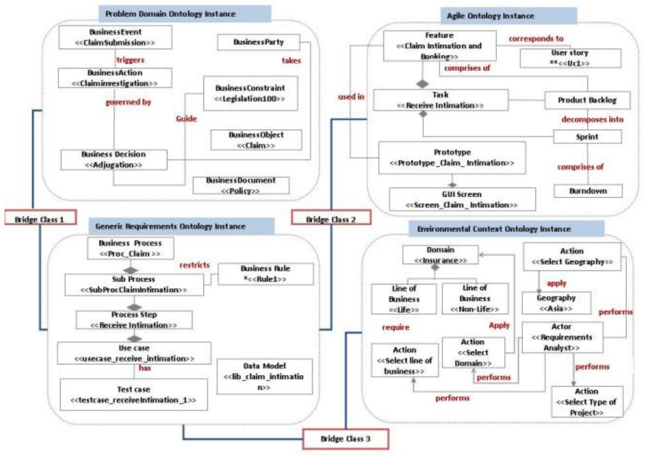
Problem Domain Ontology Instance
BusinessEvent <<ClaimSubmission>>
triggers
BusinessAction <<Claiminvestigation>>
governed by
Business Decision <<Adjugation>>
BusinessParty
takes
BusinessConstraint <<Legislation100>>
Guide
BusinessObject <<Claim>>
BusinessDocument <<Policy>>
Agile Ontology Instance
Feature <<Claim Intimation and Booking>>
corresponds to
User story **<<Uc1>>
comprises of
used in
Task <<Receive Intimation>>
Product Backlog
decomposes into
Sprint
Prototype <<Prototype_Claim_ Intimation>>
comprises of
GUI Screen <<Screen_Claim_ Intimation>>
Burndown
Bridge Class 1
Bridge Class 2
Generic Requirements Ontology Instance
Business Process <<Proc_Claim >>
restricts
Business Rule *<<Rule1>>
Sub Process <<SubProcClaimIntimation>>
Process Step <<Receive Intimation>>
Use case <<usecase_receive_intimation>>
has
Test case <<testcase_receiveIntimation_1>>
Data Model <<lib_claim_intimation>>
Environmental Context Ontology Instance
Domain <<Insurance>>
Line of Business <<Life>>
Line of Business <<Non-Life>>
Action <<Select Geography>>
apply
Geography <<Asia>>
performs
require
Apply
Actor <<Requirements Analyst>>
Action <<Select line of business>>
Action <<Select Domain>>
performs
performs
performs
Action <<Select Type of Project>>
Bridge Class 3

Figure 1 Example knowledge base instances and bridge classes that refer to them for context-specific recommendations

2.2 Problem Domain Ontology

This ontology provides abstractions to capture the essence of the problem domain. For example, consider the following scenario- 'In event of death of a policyholder, a beneficiary may submit a claim request.' The abstractions such as 'BusinessEvent', 'BusinessType', 'Party', 'BusinessAction' let one capture this information.

2.3 Requirements Ontology

The domain seed that we present to the requirement analyst is built around abstractions that capture requirements definition elements such as business goals, features, business processes and sub-processes, business constraints (laws of the land, organizational policies), use cases and business entities. The Requirements Definition ontology provides for abstractions that let one capture and organize requirements in terms of these elements and their relationships. This ontology is derived from our previous work. [6 and references therein]

2.4 Agile Requirements Ontology

This contains concepts specific to the agile requirements, e.g. 'UserStory', 'Feature', 'ProductBacklog' and 'Sprint' and so on.

2.5 Mappings between the elements of different ontologies

The 'Business events' (e.g. Claim submission), Business Actions (e.g. Investigate Claim) and 'Business Decisions' (e.g. Adjugation) in the 'Problem Domain Ontology' map to 'Business process' (e.g. Claims Process) in the 'Generic Requirements Ontology'.

'Business Goals' (e.g. Reduce Costs) in the 'Generic Requirements Ontology' are designed to deliver 'Business Value',(e.g. Profit margin) a concept in 'Problem Domain Ontology'

'Business Constraint' in the 'Problem Domain Ontology' (e.g. a New legislation) in maps to 'Validation' (e.g. Verify conformance to rule) in 'Generic Requirements Ontology'.

The 'Business Party', (e.g. Insurer), 'Business Object' (e.g. Claim) and 'Business Document' (e.g. Policy) from the "Problem Domain Ontology' correspond to 'Data Elements' in the 'Generic Requirements Ontology'.

'Feature' (e.g. Claim intimation and booking) in 'Agile Requirements Ontology' maps to 'SubProcess' (Claim Intimation process). in 'Generic Requirements Ontology'.

'User Story' and 'Task' in Agile Ontology' maps to 'Use Cases ' in "Generic Requirements Ontology'

'Data Elements' (e.g. Corresponding to Claim intimation) in 'Generic Requirements Ontology' can be used to create 'Prototypes' (e.g. Claim intimation screens) in 'Agile requirements' context.

Thus, through its linkage with the 'Generic Requirements Ontology', the Agile Requirements Ontology' has mappings to the 'Problem Domain Ontology'.

Requirement definition for each Module is divided into sets of 'Sprints' to be executed in specific time frames. Each sprint consists of 'Features' which are restricted by certain 'Business Constraints'. Further, each feature maps to 'User Stories' captured during the Analyst – Stakeholder interactions. 'User Stories' are associated with 'Use Cases' and 'Test Cases'. A 'Sprint' consists of tasks associated with 'Features' and is further mapped with 'Product Backlog' and 'Burndown' which can be displayed graphically in K-gileRE. Additionally, a 'User story' is mapped to a 'Business process' relevant to the selected 'Feature'. This serves as a reference for system testing.

The framework facilitates evolution of the domain seed into project-specific requirements by providing online semantic recommendations that are based on the underlying ontologies and their instances. This is achieved by employing the 'Bridge classes' and inference rules written in the Semantic Web Rule Language (SWRL). The 'Bridge classes' specify semantic mappings of conclusions drawn from one ontology to elements of another ontology.

For example, the 'Actor' (e.g. requirement analyst) performs functions like 'Select domain', 'select geography' and so on. Based on the selection, the K-gile RE framework draws logical conclusion about what modules should be presented to her. If she has selected 'Insurance', 'Life', 'Asia' and 'ABC Insurance', the K-gileRE framework presents to her modules like 'Claim',' Reinsurance' and features such as 'Claim initiation', 'Waiver management' and so on inferred by the 'Bridge classes'. She can select to work with features relevant to her project. If the requirement analyst selects to work with conflicting features (such as 'Claim intimation for death due to unnatural cause' together with 'Document waiver management') , the Bridge classes traverse the ontologies, sense rules that specify the conflicting nature of the features and provide an alert stating so.

3. K-gileRE Usage illustration

A requirement analyst starts with selecting environmental parameters and is presented with a core set of features from a domain knowledge seed that matches the parameter selection. As she selects to work with features, she receives recommendations about their complementary or conflicting nature. The associated user stories, use cases and tasks are also displayed. She can make a selection from these, edit the elements as necessary to suit her project needs and form a product backlog and sprints thereafter. If interdependent tasks are included in separate Sprints, she would receive an alert stating so and can make an informed decision about rearranging them. As she selects a feature to modify (or to directly include in a Sprint without modifications) she receives recommendations regarding applicable business rules, data models, and glossaries and so on.

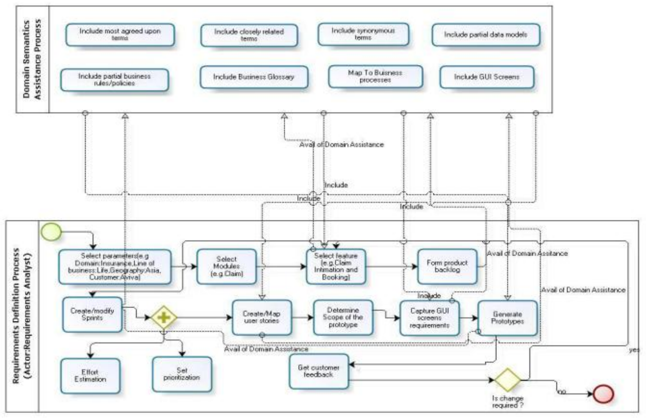
Domain Semantics Assistance Process
Include most agreed upon terms
Include closely related terms
Include synonymous terms
Include partial data models
Include partial business rules/policies
Include Business Glossary
Map To Business processes
Include GUI Screens
Avail of Domain Assistance
Include
Include
Include
Include
Requirements Definition Process (Actor:Requirements Analyst)
Select parameters(e.g Domain:Insurance,Line of business:Life,Geography:Asia, Customer:Aviva)
Select Modules (e.g Claim)
Select feature (e.g.Claim Intimation and Booking)
Form product backlog
Avail of Domain Assitance
Avail of Domain Assistance
Create/modify Sprints
Create/Map user stories
Determine Scope of the prototype
Capture GUI screens requirements
Generate Prototypes
Avail of Domain Assistance
Effort Estimation
Set prioritization
Get customer feedback
Is change required ?
yes
no

Fig 2. Domain knowledge assisted agile requirements evolution

Fig 2 illustrates the process while Table 1 highlights some of the agile requirements related activities and the domain specific recommendations available in K-gileRE. She can include the recommended elements in her requirement specification and models and act on the alerts provided by K-gileRE.

Table 1: Requirements definition and domain-specific Assistance

Requirements definition activities	Domain- specific assistance	Example(s)
Select environmental parameter	A 'domain knowledge seed' relevant to the selected parameters is presented	<u>Parameters:</u> Domain (e.g. Insurance), line of business (e.g. life), geography (e.g. Asia) and customer (e.g. ABC), <u>Domain knowledge seed:</u> presents Modules such as Claims, Riders, Maturity
Editing elements such as User Story from the seed	Recommendations to include Features that would help in implementing the user story, adherence to terminology, detection of new terms and recommendations to include them in glossary and data models, recommendations to specify associations between terms .	<u>User story text:</u> "As an Insurer, I want to have Claim Intimation & Booking feature with automated agreement verification in my insurance Application so that the verification process gets completed within 2 days."
Select features (from the domain knowledge seed) relevant to project	Recommendations to include business rules/policies relevant to features,, Business Glossary, Business Process, , Include Closely Related Terms	<u>Selected Feature:</u> Claim intimation and booking <u>Business Terms :</u> Assignee, <u>Rules:</u> Laws of the land with respect to claims, in Asia, <u>Policies</u> of the selected company (ABC) ,conflicting features
Form product backlog and sprints thereafter	Recommendations to include inter-dependant features in the same sprints, Splitting of a feature	'Claim intimation' and 'Claim review and inspection' may be included in the same Sprint.
Generate prototype	Typical screens, partial data models , use cases	<u>Recommendations:</u> Sample screens depicting the 'Claim intimation'

Requirements definition activities	**Domain- specific assistance**	**Example(s)**
		activities, data models (e.g. consisting of Claim, Policy, Agent)

The requirements analyst can generate structured requirements specification documents and partial domain models intermittently. She can refine partial data models and use them for downstream development using industry standard model-based development tools. The analyst can either work on the 'text' or 'diagram' mode and import/ export (to /from either text or diagram mode) using industry standard data exchange formats (e.g. XMI, XPDL). As she works on her textual specification of a User story or a feature, the framework detects new business terms/ key phrases that may appear in the description. For example, if the term 'Customer' appears in some feature description, K-gileRE prompts that 'Insured' is a commonly accepted term. She can make an informed decision about replacing 'Customer' by 'Insured' or retaining it as it is. K-gileRE framework supports generation of low (UI layouts only) and hi-fidelity (UI layout +functionality) prototypes that can be used for verification and validation by the customer and for achieving iterative and frequent feedback. This helps in refining requirement documents and models incrementally.

During the requirements definition exercise, a requirement analyst can consult experts using semantically enriched collaborations. For example, if she starts a discussion forum on Life insurance rules, she is presented with a set of relevant posts available on the topic such as Rules for 'ABC Inc', Rules in APAC, other experts' opinions and so on.

K-gileRE framework starts with a seed requirement specification that can be evolved into one that suits specific project needs, hence the term- Requirement Evolution as opposed to a clean slate Requirements Engineering. The framework is built on web 2.0 architecture of participation to leverage its collaborative aspects for requirements definition, which is inherently a collaboration- intensive process.

Knowledge creation and selection are achieved by providing the roles: Domain Knowledge Contributor and Domain Knowledge Curator. We do not discuss these here.

4. Validation

Using K-gileRE framework a knowledge base comprising of 300 'Claims' features, 3269 business concepts and relations, 822 business rules along with exceptions and over-rider scenarios and a glossary explaining the concepts was created. This work was done using the Knowledge Contributor role in K-gileRE framework by 3 domain experts and 2 domain curators from the Domain Competency Group of the Insurance Industry Solutions Unit in our organization.

We used as our reference for this validation, a requirements document the requirement analysts had prepared. The project had a product-backlog of 170 features organized into sprints in consultation with customers. We selected one of the Sprints consisting of 10 features for our experiment. We compared these with the ones present in the knowledge base incorporated in K-gileRE. We selected 10 features that matched closely in functionality from the knowledge base and modified these to match the project needs (with the document as our reference) While we did this exercise, we received several recommendations from the K-gileRE framework. To understand the effectiveness of the framework, we recorded recommendations related to (1) missing elements such as business rules corresponding to a feature (2) inconsistencies (such as conflicting features) (3) terminology suggestions (4) corrections (such as modifications, deletions and additions) to the 'seed' presented. This was done in order to identify possible gaps in a seemingly complete requirements document.

We accepted and acted on some of the recommendations and had to reject some in consultation with the requirement analysts who had actually interacted with the customers. The elements mentioned in recommendations were displayed for inclusion and could be edited to suit the specifics of the project. For example, if a recommendation was regarding business rules relevant to a selected feature, then the corresponding rules were displayed and one could select to include some (or none) from these depending upon the project specifics. Table 2 summarizes the observations.

Table 2: K-gileRE effectiveness

Knowledge element selected/ edited	**Number of recommendations displayed by K-gileRE**	**Recommendation details**	**Number of recommendations accepted and acted upon**
Features	40	Complementary features, conflicting features, missing business rules, suggestions for Sprint formations	28
Use cases	12	Relevant business rules	7
Business concepts	20	Relationships between the concepts suggested	15
Synonym usage	15	Most accepted term in place of the synonym	12
Business term to be included in project glossary	75	Related terms to be included along with selected term	50

Though the experiment is small in size, it brings out the potential strength of the method and framework. We are aware that the results presented here are only indicative and we need to test this approach on field in a large project. We will be taking up this exercise next. We find that this approach has the potential to improve several desirable properties of requirements. We realize that the effectiveness of this approach will be largely dependent on the quality of domain knowledge seed that we are able to provide. Also, this method would require a mindset change for a larger adoption in any organization. To address this need we have adopted a hybrid approach; one that lets the requirement analysts work in their natural mode – that of textual specifications and have provided only a light-weight formalism for a semantic assistance. The requirement analysts are not required to learn any new visual notation, or a mapping technique or modeling tool to be able to use K-gileRE.

It is obvious that the framework can be used in the context of traditional requirements as well, since it also incorporates Generic Requirements Definition Ontology. One of the suggestions from a reviewer of this work has been that we need to explore use of concept maps or other lightweight modeling techniques for modeling domains. Performance and scalability of this approach will also need to be addressed for it to be deployed in large projects.

5. DISCUSSIONS AND CONCLUSION

An agile requirements exercise requires the whole development team to collect requirements from the customer [7]. This is expected to reduce the effort involved in sharing knowledge documents and also the probability of misunderstanding. Co-located teams find it relatively easier to acquire knowledge about a problem domain by staying in close contact with the customers. However, geographically dispersed teams form a roadblock to knowledge dissemination. The problem of 'distributed agile' has been addressed by some development environments [8]. But this approach solves the problem only partially. It provides a way for the developer community to come together and interact and also supports the 'governance' part of the development exercise, but does not really equip them with the domain knowledge edge they need. Customer is supposed to be the domain expert who makes decisions [7]. However customer involvement of the level that agile requirements definition exercises advocate, is very difficult to achieve and customers actually expect the vendor organizations to possess the necessary knowledge in a problem domain. Requirements are to be collected using the language of the customer and not a formal language for requirements specification [7]. This reinforces that the requirements analysts needs to be equipped with the knowledge of problem domain, in order to 'speak' the language. If the development team considers a requirement too complex, it is split into simpler ones [7]. It can be easily appreciated that such a splitting would have to be a guided exercise. For example, if a set of inter-dependent or complementary features are 'split 'and included into tasks that belong to different sprints, without the awareness that they need to function together finally, it may be difficult to achieve the desired results. A need for providing explicit and seamlessly incorporated domain knowledge assistance to agile requirements is thus obvious. No agile method, framework or tool currently supports this crucially important need. Methods and techniques to structure domain knowledge and use it in requirements engineering exist ([9 and references therein], but they do not explicitly take into account the agile context. Also, they do not include recommendation mechanisms to achieve an effectual domain knowledge usage.

The online context- sensitive recommendations inferred from the underlying knowledge bases in K-gileRE render a 'paired experience' (analogous to pair programming [10]) while defining requirements, at least partially substituting for a domain expert. K-gileRE achieves Knowledge dissemination essential for agility by facilitating semantically enriched collaborations. It combines benefits of the meritocratic aspects of the semantic web and the democratic aspects of web 2.0.

All agile methods strongly advocate close communication and collaboration. Using K-gileRE, interactions among dispersed teams happen informally, in keeping with the agile culture and doctrines of trust and care for individuals. A virtual 'stand up' meeting among geographically dispersed teams can be easily facilitated.

Agile methods advocate parsimonious documentation and executable requirement models. K-gileRE framework facilitates automated generation of requirements models such as editable and evolvable business process maps, use case models and data models that form inputs to downstream development (e.g. generating code from data models captured as UML class models). The domain knowledge seed provides a jump start for an agile requirements definition. No existing agile method/tool/ framework incorporates this concept.

Scott Ambler's recent APMM-Agile Process Maturity Model [2] refers to 3 levels – level 1 addresses optimization needs of co-located teams. The level 2 process incorporates governance while level 3 takes into account scaling factors such as team size, geographical distribution, regulatory compliance, and environmental complexity. With reference to this model, we have conceptualized an agile approach that is more advanced than a 'Level 3' agile process. Not only do we incorporate the level 3 aspects, but also explicitly provide a crucially important domain value to an agile exercise hitherto left unaddressed by all existing agile approaches.

6. References

[1] Ivar Jacobson, Scaling agility, http://www.ivarjacobson.com

[2] Scott Ambler, On World of agile development http://www.informit.com/articles/article.aspx?p=1380372

[3] Scott Ambler, Agile modeling, ttp://www.agilemodeling.com

[4] Charmaz, K. Constructing Grounded Theory: a Practical Guide through Qualitative Research, Thousand Oaks CA, Sage, 2007

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 5, 2001

[6] S. Ghaisas, A method for identifying unobvious requirements in globally distributed softwareprojects, Lecture Notes in Informatics (LNI) – proc., In Proceedings of SENSE09, Kaiserslautern, Germany, Mar. 2009, pages 297-308, 2009.

[7] Alberto Sillitti, Giancarlo Succi; Engineering and Managing Software requirements: Chapter 14, Pages 309–326, Apr. 2006-340 2008.

[8] See for example, Rational Team Concert, http://www-01.ibm.com/software/awdtools/rtc/

[9] Haruhiko Kaiya, Motoshi Saeki., Using Domain Ontology as Domain Knowledge for Requirements Elicitation, Proc. 14th IEEE International Requirements Engineering Conf. RE 2006, Minneapolis/St. Paul, Minnesota, USA, Sept 11- 15 ,186-195, , 2006

[10] L Williams, R. kessler, Pair programming illuminated, Addison Wessley, 2002