A Semantic and Collaborative Platform for Agile Requirements Evolution

Nirav Ajmeri, Riddhima Sejpal, Smita Ghaisas
Tata Research, Design and Development Centre (TRDDC)
A Division of Tata Consultancy Services
54-B, Hadapsar Industrial Estate, Pune – 411013, India
e-mail: {nirav.ajmeri, riddhima.sejpal, smita.ghaisas}@tcs.com

Abstract— The characteristics of web-based and community-oriented social software are very useful in the context of software engineering in general and requirements engineering in particular. Their ease of use, transparency of communication, user orientation, self organization and emergent nature resulting from a continual social feedback are particularly relevant to an agile requirements definition exercise. The reason is that agile requirements are inherently meant to be collaboration-intensive. However, while the benefits of social platforms are valuable, they are necessary and not sufficient in themselves for making the exercise effective. The emerging social software engineering discipline is about enabling community-driven creation, management and deployment of software by applying methods, processes and tools in online environments. In this paper, we report our work on a semantic and collaborative platform that combines the virtues of social software principles and the semantic web concepts to enable knowledge-assisted agile requirements definition.

Keywords- Social software engineering; collaborative and semantic requirements definition; semantic assistance

I. INTRODUCTION

With agile development becoming increasingly mainstream, there is naturally a lot of emphasis in recent years on agile requirements. Contrary to the earlier leanings on an entirely code-centric development, agile veterans now advocate at least lightweight requirements definition in agile projects [1]. Requirements definition however continues to pose a challenge as much to agile development projects as to traditional ones [2]. Moreover, most large projects involve stakeholders from various geographies. As a result of the distributed teams, there is very little or no opportunity for co-located discussions among them. This jeopardises the success of a project because agile requirements definition needs intensive collaborations among stakeholders [3]. Respondents to a survey cite communication as the top agile challenge. They express that it is extremely important that the distributed teams 'use the same language' while defining requirements. [3].

While the problem of collaboration has been addressed by various tools [4], the communication 'using the same language' is difficult to achieve unless we equip requirement analysts with a platform that, in addition to supporting collaboration; also seamlessly incorporates domain knowledge. The knowledge should be visible, accessible and structured in a way to be amenable to tailoring to suit specific project needs.

In the work reported here, we combine useful concepts from the web 2.0 stream and the semantic web stream [5, 6] with an aim to enable an agile requirements identification, discussion and definition. We term this approach Knowledge assisted Agile Requirements Evolution (Kgile-RE).

A requirement analyst using K-gileRE is presented with a domain knowledge corpus comprising core user stories and associated knowledge elements such as business rules, processes, use cases, data models, prototypes and possible sprint plans. In collaboration with the customers, domain experts and other requirement analysts, she can modify and enhance the knowledge elements to suit her specific project needs. As she defines (new)/ alters (existing) knowledge elements (for example, identify a new use case or modify an existing business rule), the text is parsed for detection of new business terms. The detected terms are matched with concepts from in-built ontologies and assistance is provided on aspects such as synonymy of terms and complementary as well as conflicting nature of knowledge elements. Each new exercise of requirements definition thus, becomes an evolution of a generic structured domain knowledge corpus tailored to suit specific projects, as opposed to the traditional 'clean slate' approach. Hence the term Knowledge assisted Agile Requirements Evolution (Kgile-RE) in contrast to the clean slate Requirements Engineering (RE). The context - sensitive assistance (based on the ontologies and inference rules that operate on them) serves as a moderation mechanism. This complements the collaborative identification, discussion and definition of requirements facilitated by social software engineering platform [7-10].

The paper continues into Section II with an overview of the platform and a usage illustration. In Section III we present the results of an experiment and discuss how our approach combines the best of both- web 2.0 and the semantic web worlds. Section IV discusses related work and Section V presents conclusions.

II. THE KGILE-RE APPROACH

In this section we introduce the Kgile-RE approach, present details of its model and illustrate its usage.

A. An Overview

The Kgile-RE platform classifies knowledge using ontologies and their instances [5, 6]. Semantic assistance to requirements definition is derived from four types of ontologies and inference rules.

Based on the environmental parameters selected by the requirement analyst, a suitable Knowledge Corpus (henceforth referred to as KC) is made available to her. For example, a requirement analyst working on Life Insurance domain in Asia-pacific region for the customer ABC Insurance is presented with a KC different from an analyst working on same domain but in Europe region for another customer XYZ Insurance. Here, we address the fact that stakeholders may need to conform to different laws of land, may be used to different terminologies depending on where they are located, and they may have different organizational policies even though the problem domain is the same. A requirement analyst is not necessarily a domain expert, yet needs to take into account these factors while defining requirements.

Kgile-RE provides a semantic assistance to the analyst while evolving over the KC. The generic requirements definition assistance is based on the rules defined in the method published elsewhere [11]. The concepts in the Generic Requirements Ontology are mapped with concepts in the Agile Requirements Ontology which represents agile requirements context. The domain specific assistance is provided on aspects such as synonymy of terms, complementary and conflicting nature of features selected, relevant business rules in the selected geography, customer-specific business policies derived from project executed for the same customer earlier, interactions of the selected domain with other domains etc. Lexical decomposition techniques are used to resolve requirements descriptions (input by users) into constituent terms. Each detected term acts as pointer to concepts in the domain ontology. For example, the Problem Domain Ontology contains the 'Synonym' relationship among certain terms. If the analyst uses a synonymous term, she receives a recommendation to replace it with the most commonly used term. Thus, if the requirement analyst enters a feature- 'Verify customer's details and send notification to the insured', the Kgile-RE platform will prompt her that 'Customer' and 'Insured' are synonymous terms, but 'Insured' is the commonly accepted term.. Similarly if a term 'Adjudicator' is used in the specification, there is an alert indicating that 'Arbitrator' is the most commonly accepted term. The details of different ontologies and inference rules are described in section II B.

The collaborations between various stakeholders are facilitated on a web2.0 platform. This architecture is selected because agile requirements definition is a highly interactive process and web2.0 provides architecture of participation.

B. The Kgile-RE Model

The four ontologies in K-gileRE, - 'Environmental Context Ontology', "Generic Requirements Ontology', 'Agile Requirements Ontology' and 'Problem Domain Ontology' are created using RDF-OWL schema [5, 6]. Figure 1 shows partial example instances of the ontologies depicted using the UML class diagram notation.

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 abstractions ***Actor, Action, Domain, LineofBusiness***, ***Customer, Geography,*** are used to capture the information.

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*** are used to capture this information.

3) Generic Requirements Ontology: The KC 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. [11 and references therein].

4) Agile Requirements Ontology: This contains abstractions specific to the agile requirements, e.g. ***UserStory, Feature, ProductBacklog , Sprint*** and so on.

C. Examples of mappings between the elements of different ontologies

- The ***BusinessEvents*** (e.g. Claim submission), ***BusinessActions*** (e.g. Investigate Claim) and ***BusinessDecisions*** (e.g. Adjugation) in the Problem Domain Ontology are represented as ***BusinessProcess*** (e.g. Claims Handling) in the 'Generic Requirements Ontology'.
- ***BusinessGoals*** (e.g. Reduce Costs) in the Generic Requirements Ontology are designed to deliver ***BusinessValue***, (e.g. Profit margin) a concept in 'Problem Domain Ontology'
- ***BusinessConstraint*** (e.g. a New legislation) in the Problem Domain Ontology in maps to ***Validation*** (e.g. Verify conformance to rule) in Generic Requirements Ontology.

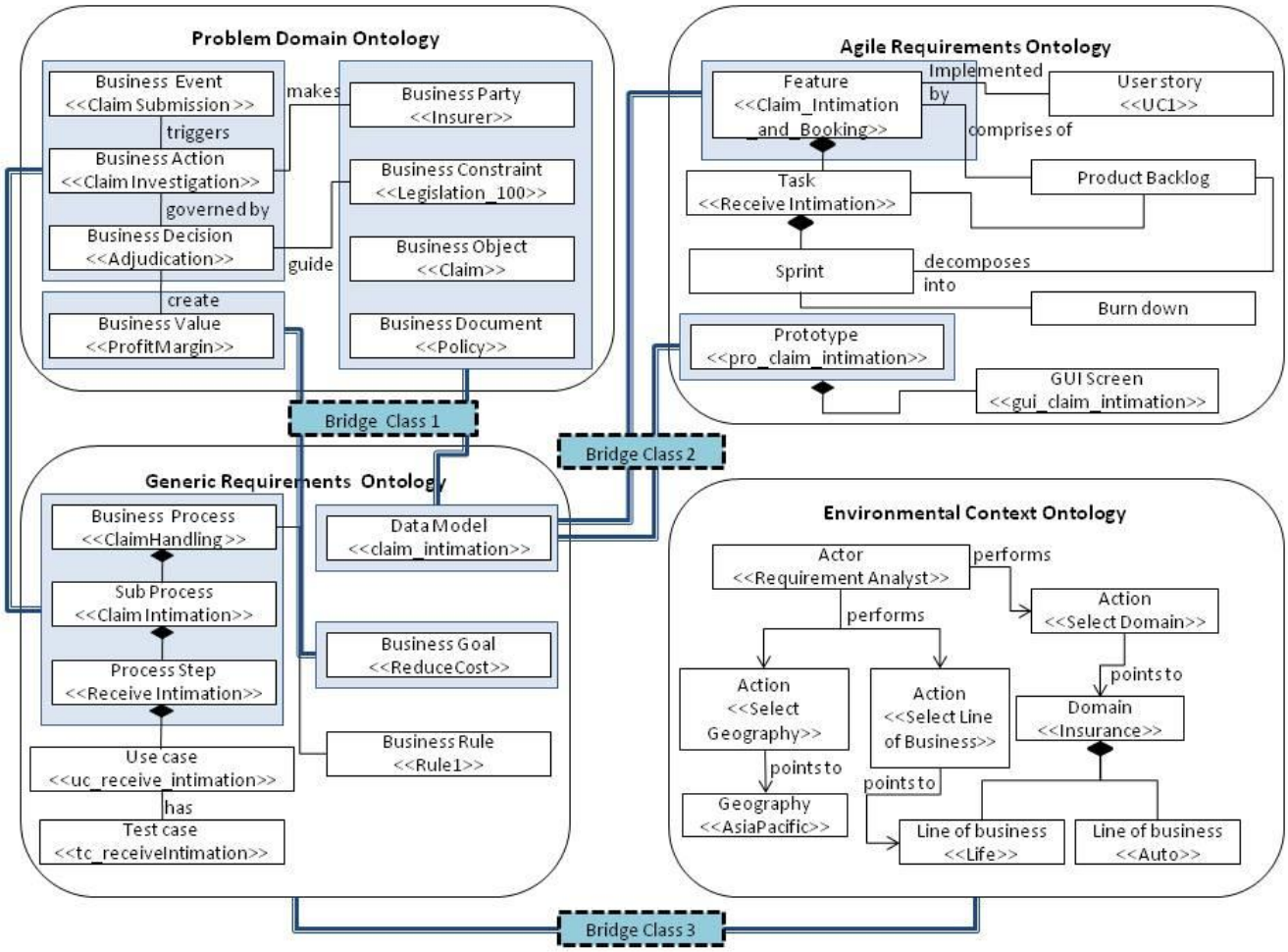

Figure 1. Example knowledge base instances and bridge classes that refer to them for context-specific recommendations

- The ***BusinessParty*** (e.g. Insurer), ***BusinessObject*** (e.g. Claim), ***BusinessDocument*** (e.g. Policy) from the Problem Domain Ontology contribute to ***DataElement*** 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.
- ***UserStory*** and ***Task*** in Agile Ontology map to ***UseCase*** in Generic Requirements Ontology

Requirement definition for each Module is divided into sets of Sprints to be executed in specific time frames. Each ***Sprint*** is composed of ***Features***. While implementing BusinessConstraints need to be taken into account. Further, each ***Feature*** consists of ***UserStories*** captured during the Analyst – Stakeholder interactions. ***UserStories*** are associated with ***UseCases*** and ***TestCases***. A Sprint consists of ***Tasks*** associated with Features and is further mapped with ***ProductBacklog*** and ***Burndown*** which can be displayed graphically in K-gileRE. A ***UserStory*** is indirectly mapped to ***BusinessProcess*** through the mapping between ***Feature*** and ***SubProcess***. This serves as a reference for system testing.

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. We define rules that refer the ontology-instances and provide recommendations based on the integrated inference thereof. This helps the requirement analyst in improving completeness, correctness and consistency of her specifications as a result of an in-built and explicit domain knowledge value. The recommendations may be specific to a singular ontology or span the four ontologies when necessary; in response to actions of the requirement analyst. For example, if a requirement analyst selects 'Europe' as the

geography for a 'Claims Handling' application to be developed, she would be presented with features and user stories relevant to 'Insurance Claims Processes' from the KC. As she starts to modify them in the context of her project, she would be presented with business rules, in the given geography e.g. 'Claims rules in Europe' (Environmental Context Ontology and Problem domain Ontology). If she selects features that complement each other but decides to associate them with different sprints, she would receive a recommendation to preferably rearrange them in the same sprint (Problem Domain Ontology and Agile Requirements Ontology). If she adds a new feature to the Product backlog upon the Customer's suggestion, and it happens to conflict with an already selected feature in a given domain, she would be alerted about the inconsistency of her selection (singularly the Problem Domain Ontology). We present examples to illustrate our approach and also discuss how it supports agile doctrines in Section II E.

D. Knowledge Creation Process

The ontologies discussed above have been a result of an iterative and a continual process that involves (1) exploring available resources such as documents, web-sites, (2) identifying and extracting various knowledge elements from the resources (3) Analyzing the extracted knowledge and (4) representing the knowledge in the form of instances of the concepts and their relationships in the ontologies in a machine readable format. We provide the role of 'Domain contributor' for this purpose. We have attempted to partially automate the process and have identified the points of human intervention.

K-gileRE employs a web crawler that explores various resources on the web and detects terms and key phrases specific to a given domain (such as Insurance in our example). We will refer to these domain specific terms as 'Concept_instances' because they are instances of concepts in the ontologies. For example, a term as ***Policy*** is a concept_instance of the concept ***BusinessDocument*** in the Problem Domain Ontology. The concept_instances are then parsed to detect similarity mappings. The techniques employed are lexical similarity [12], semantic similarity [13], direct string matching [14] and ontological structure based mapping [15]. We also perform 'Complementarity mapping' to help identify associations between concept_instances. Relations like subclass, super class, equivalent, part-of, and concepts related with each other by minimum cardinality of one on both sides are considered. The associations between concept_instances are subject to refinements by human intervention.

The Domain contributor is presented with concepts from the Problem Domain Ontology and is required to map the concept_instances with the concepts. (E.g. ***Insurer*** is a concept_instance of the concept ***BusinessParty***). If there are some concept_instances that do not seem to be instances of any existing concepts in the ontology, the Domain contributor can identify new concepts. Thus the ontology itself evolves and gets refined and enhanced in the process of evolving the KC. After the newly identified concepts are reconciled with the ontology and concept_instances are mapped to concepts, the domain contributor is presented with the resultant structured knowledge and is required to identify associations between concepts and concept_instances if necessary. The concept_instances are reviewed for similarity mapping again after this step.

The detected key phrases are presented to Domain Contributor so that she can identify these as one of the following: (1) relations between concepts (e.g. ***Insured*** has ***Policy*** (2) features (e.g. ***Claim intimation*** is followed by ***Claim scrutiny***) (3) user stories ***("As an Insurer, I want to have Claim Intimation & Booking feature with automated agreement verification…. ")*** (4) use cases (e.g. ***Scrutinizer prepares a report to be reviewed by insurer***) (5) business constraints (***A claim must be made only against a valid policy***). If a key phrase is identified as a feature, the Domain contributor is asked to specify complementary feature(s) and conflicting feature(s) from a list of available features in the existing KC. She can also add new complementary/ conflicting features to the KC. If a key phrase is identified as a use case, the Domain Contributor specifies actor(s) from the available list or adds new ones to the KC. She also identified 'includes' and 'extends' use cases for a given use case from the KC or adds new ones. If a business constraint is identified, the features, user stories, use cases that are affected by the constraint are specified.

This is followed by a machine readable representation of the knowledge. The structural part is captured as RDF-OWL schema while the behavioral part is represented by various semantic rules that operate on the ontologies. Both the schema and the SWRL rules can be specified using a UI that incorporates structural and behavioral patterns. For example, the relationships between concepts are specified by identifying the source concept (e.g. Policyholder), destination concept (e.g. Policy) and association(s) (e.g. ***has***) while the rules are captured using placeholders (e.g. for ***if-then-else***).

K-gileRE incorporates a process for refining the (thus) structured knowledge by way of directly adding concept_instances, associations, new concepts, rules as well.

The Knowledge must be reviewed for its correctness and currency. The role 'Domain Curator' facilitates this. The activities in this role are outlined briefly below.

Role: Domain curator
Responsibility: Selecting and refining knowledge.
She can

- view submissions from domain experts
- modify/refine the elements if necessary
- select to accept or reject submissions
- invite discussions and vote on submissions if necessary
- finalize the elements that should reside in the knowledge base.

E. Usage Illustration

A requirement analyst starts with selecting environmental parameters and is presented with a core set of features from a KC that matches the parameter selection. As she selects to work with features, she receives recommendations about

TABLE I. REQUIREMENTS DEFINITION AND DOMAIN-SPECIFIC ASSISTANCE

Requirements definition activities	Domain- specific assistance	Example(s)
Select environmental parameter	A KC relevant to the selected parameters is presented	**Parameters:** Domain (e.g. Insurance), line of business (e.g. life), geography (e.g. Asia) and customer (e.g. ABC), **KC** presents Modules such as Claims, Riders, Maturity
Editing elements such as User Story from the KC	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 .	**User story text**: "As an Insurer, I want to have Claim Intimation & Booking feature with automated agreement verification in my Claim Handling module of Insurance application so that the verification process gets completed within 2 days." **Features:** Claim intimation and booking, Claim review and inspection **New Terms detected:** Verification, **Recommended synonym :** Scrutiny
Select features (from the KC) relevant to project	Recommendations to include business rules/policies relevant to features,, Business Glossary, Business Process, , Include Closely Related Terms	**Selected Feature:** Claim intimation and booking **Business Terms :** Assignee, **Rules:** Laws of the land with respect to claims, in Asia, **Policies** 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	**Recommendations:** 'Claim intimation' and 'Claim review and inspection' may be included in the same Sprint.
Generate prototype	Typical screens, partial data models , use cases	**Recommendations:** Sample screens depicting the 'Claim intimation' activities, data models (e.g. consisting of Claim, Policy, Agent)

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.

Table I 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.

1) Semantically enabled collaboration: During each of the above activities, she can start discussions in the form of informal chats on the selected knowledge elements with her colleagues, experts and seek their opinion on her selections from and refinements on the KC. She can post topics for discussions on semantically enabled forums and subscribe to alerts when others post their opinions on topic of her current interest.

For example if she selects the following rule to be included in her specification:

'If no. of years of premium paid from the date of commencement is equal to 4 years, then policy acquires paid up value'

But she is not sure if this is valid in India, she can start a forum to discuss this with experts. Upon initiating a forum, she will be presented with a set of relevant posts available on the topic. For example, she can view posts related to validity of rules for Life Insurance, Rules for ABC Inc, Rules for India, posts by other experts who contributed Life Insurance rules, rules regarding related terms such as 'date of commencement' 'premium' and select the most suitable thread of discussions in terms of topic, author geography and so on. She can start an entirely new forum as well, if none of the presented ones match her need.

2) Generating and refining artifacts iteratively: The requirements analyst can generate structured requirements specification documents intermittently. She can view Sprints, Product backlogs, Burn down charts. She can populate data models using modeling tools.. The analyst can either work on the 'text' or 'diagram' and import/ export to /from either format. This helps in refining artifacts incrementally.

Starting with a KC for 'Death claim process', we can thus evolve a specification that suits a given project. The evolution is an assisted exercise that helps in adding to or modifying the KC by providing context-sensitive help to a requirement analyst.

It is relevant here to add that not all of the domain knowledge is formalizable in terms of ontologies and the semantic web rule language (SWRL). We therefore use a combination of formalization and human intervention to represent knowledge and enable its reuse during agile requirements definition. Let us consider a small subset of activities and the corresponding K-gileRE responses as an example. Table II illustrates parts of domain –specific assistance that is based on ontologies entirely and parts that require human intervention.

TABLE II. DOMAIN SPECIFIC ASSISTANCE ENABLEMENT - FORMALIZATION+ HUMAN INTERVENTION

Requirements definition activities	Domain- specific assistance	Examples	Formalized/Human intervention	Remark
Select Project Environment Parameter Set	1.List of Available Domains from Knowledge Repository is presented 2. List of Available Geographies from Knowledge Repository is presented	1. Insurance, Banking, Healthcare... 2. Europe, Asia-Pacific...	Formalized	Environment Ontology +SWRL
Select Domain	List of related Line of Business is presented.	Life Insurance, Auto insurance...	Human intervention based	Requirement Analyst is required to select relevant Line of Business as per her project needs.
Select Geography	List of related Customers is presented.	ABC Life, XYZ Auto...	Human intervention based	Requirement Analyst is required to select the appropriate Customer as per her project needs.
Select Customer	A KC relevant to selected parameters is presented	Claim Handling	Formalized	Ontology +SWRL

III. RESULTS AND DISCUSSIONS

In this section we present an example based on the results of an experiment and discussions.

A. *Expirement*

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 by 3 domain experts and 2 domain curators collaboratively from the Domain Competency Group of the Insurance Industry Solutions Unit (ISU) in our organization.

We used as our reference for this validation, a requirements document the requirement analysts had prepared for a 'Claims' application. 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 KC 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 III summarizes the observations.

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 completeness, correctness and consistency of requirements. We realize that the effectiveness of this approach will largely depend on the quality of KC that we would be able to provide.

B. *Discussions*

While adopting the social software principles to a specialized field such as requirements definition, we take into account the following seemingly contrasting aspects in the context of our community of practice comprising requirements analysts (consumers of knowledge), domain experts (contributors and curators of knowledge), project managers and customers.

TABLE III. KGILRE-RE 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

1) Taxonomies and Folksonomies: We use four different ontologies to render a structure to the requirements definition exercise and to provide context-sensitive assistance- (1) Environmental context ontology (2) Generic Requirements ontology (3) Agile requirements ontology and the (4) Problem Domain ontology. The meta- model for each of these presents a distinct context for classification of requirements elements. For example Environmental Context Ontology is a formal specification of ethnic groups to which the user belongs. It consists of concepts like Domain, Line of business, Geography, Customer and Project type. The requirements definition ontology is based on the criteria for completeness, consistency and correctness of requirements specification. This is derived from experiences and best practices in projects. The domain ontology refers to abstractions in a given problem domain such as banking, insurance and therefore contains business concepts, their inter-relations and constraints. It is quite obvious that the ontologies are constructed hierarchically, in a top-down way. In each of this category, apart from the pre-defined taxonomy, a user is at liberty to identify new elements. This constitutes the folksonomy which evolves bottom-up in a community-driven way. If their usage in the community of practice (in this case, stakeholders in requirements definition exercise) is substantial, the elements can be absorbed into the taxonomy. This decision however is made by the knowledge curator in consultation with the community of practice.

2) Self Organization and moderation: A wiki-like platform for entering, editing requirements and for deliberating on them was highly suitable. However unlike the highly democratic communications which the typical social networking mechanisms (e.g. blogs) allow, K-gileRE could not do away entirely with supervision and moderation by requirements experts. The moderation however, had to be in the form of suggestions, alerts and a non-intrusive assistance. We therefore built in a semantic domain knowledge assistance mechanism. As a requirements analyst details a requirement, he receives soft alerts regarding conforming to a common and accepted terminology of the specific domain and/or environment in which he is working, complementary knowledge elements such as laws of the land that apply to the functionality he is illustrating, conflicting nature of certain functionalities, SDLC specific artifacts such as relevant use cases and test cases for the functionality. The decision to modify requirements by acting as per the alerts or ignoring them is thus an informed one. K-gileRE also provides a cumulative report on inconsistencies, variations and over-riders (from the reference domain knowledge base) in the requirements specification instance created by the users for their specific projects.

3) Social feedback: The platform provides for voting and comments on requirements by all stakeholders. We realized that opinions of stakeholders such as domain experts need to be taken into account with a higher weight than less experienced stakeholders in an organization. Thus voting was not found to be the best way to decide for example: whether a given requirement would be considered in the first iteration or in subsequent ones. This seemingly defeats the purpose of a 'social ' platform, but the transparency of discussions bears down heavily on such decisions and since the comments and discussions are for all to see, no single heavy-weight stakeholder can unfairly overrule valid suggestions made by even junior stakeholders, or they would face pressure from other experts in the community. We designed various suitable roles for this purpose with a view to incorporate suitable weights for respective stakeholders. For example a 'Domain expert' role has a higher weight than a 'Requirement analyst' role, but a lower weight than a 'Domain curator' role.

4) More than just a Wiki: The platform provides for meaningful collaborations that are semantically enriched as explained earlier in the Section on usage illustration. In the context of agile requirements, we note that it is important to produce executable requirements models [1] in addition to documents. It is possible to derive from our platform; requirements models such as editable and evolvable business process maps, use case models and domain models. These can be imported into tools that incorporate industry standard data exchange formats (e.g. XPDL for business process models) and utilized in downstream development (e.g. generating code from domain models captured as UML class diagrams). A structured requirement specification document can be generated from the user inputs as well.

IV. Related Work

Lohman et al [7] highlight the aspects we discuss in the previous section. They have noted that supervision and moderation by requirements experts remains crucial to a project's success. Their Softwiki platform achieves combination of concepts related to community driven requirements acquisition and management and semantic structuring of knowledge. They mention that the moderation should be unobtrusive. Kgile-RE platform provides for this moderation at two levels (1) At the requirement analyst level, Kgile-RE gives just-in-time alerts and recommendations. This context sensitive assistance is based on the underlying knowledge base comprising of four different ontologies and rules. It explicitly incorporates agile requirements related concepts and rules to enable the assistance (2) At the Domain contributor level, the context sensitive assistance incorporates domain analysis related rules. Additionally, the domain curator role,is empowered to select, moderate and refine the contribution in consultation with the community of experts.

Anna Hanneman et al [8] compare identification of requirements with and without the web 2.0 style elicitation support in their research. Their Bubble Annotation Tool (BAT) provides for "enjoyable and intuitive interactions with the community" collaborating in the requirements engineering processes. They do not however provide any means for semantic and domain specific assistance or agile requirements.

Ankolekar et al [16] have stressed that the semantic web and web 2.0 complement each other and that in fact both communities need elements from the other's technologies to overcome their own limitations. They emphasize that semantic technologies bear a great potential of providing a robust and extensible basis for Web 2.0 applications.

Seater et al [17] propose an approach based on problem frame concept. The problem frame enables depiction of properties and connections of the environment to which the problem is related. But the requirement analyst himself has to come up with the domain assumptions ('breadcrumbs') and thus requires domain expertise.

Kaiya et al [18] discuss use of domain ontologies for verification of inconsistencies. In their approach, requirements concepts are extracted using NLP techniques and the analyst is required to map them to the domain ontologies. The completeness or consistency of requirements specification is verified based on the extent of the mapping. Their work does not take into account the agile requirements context explicitly. The framework does not support collaborative aspects of requirements engineering either.

Our approach presents a semantic and collaborative requirements definition method and platform that uses KC instead of a clean slate as a point of departure. We have seamlessly incorporated various useful aspects form the web 2.0 and the semantic web streams to collaboratively define requirements and provide a context-sensitive just-in-time assistance based on the four knowledge contexts.

V. Conclusion

Kgile-RE bridges the democratic aspects of web 2.0 platform and the semantic web concepts. While collaborative identification, discussion and definition of requirements facilitated by web 2.0 are valuable, we cannot entirely do away with moderation in such a highly specialized exercise. The context- sensitive assistance mechanism based on semantic web concepts serves as a moderation mechanism as well. The decision to modify requirements by acting as per the alerts or ignoring them is an informed one. The four ontologies provide pre-defined taxonomies to facilitate classification of requirement elements. Apart from the pre-defined taxonomies (which are built hierarchically top-down way), a user is at liberty to identify new elements. This constitutes the folksonomy which evolves bottom-up. If their usage in the community of practice (in this case, stakeholders in requirements definition exercise) is substantial, the elements can be absorbed into the taxonomy. Moreover, K-gileRE platform is not just a Wiki. It provides for a semantically enriched collaboration that would foster meaningful and focussed discussions on topics in requirements engineering in general and problem domain such as Insurance in particular. The platform also provides for automated generation of requirements models that form inputs to downstream development (e.g. generating code from domain models captured as UML class diagrams) and documents.

The framework 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 the collaboration mechanisms in 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

Though the work presented here focuses on agile requirements, our interactions with project teams in our organization indicate that the knowledge reuse facilitated by K-gileRE is applicable generically to projects that need domain knowledge assistance. What we do however want to stress on is that we explicitly map domain knowledge elements onto agile requirements elements to incorporate a semantic assistance into the process of agile requirements. The knowledge reuse thus becomes an inherent part of the agile requirements exercise, an aspect not taken up so far by other existing agile methods and frameworks.

We find that this approach has the potential to improve several desirable properties in a requirement specification. This is brought out by our initial experiment. We realize that this approach will depend largely on the quality of KC that we are able to provide and that this would require a mindset change for a larger adoption in any organization.

We have taken up large-scale exercises involving knowledge creation, moderation and consumption using Kgile-RE in Insurance (life and P&C) and Investment

Banking domains. The empirical results will be published soon.

ACKNOWLEDGEMENTS

We would like to thank V.S. Sivakumar of the Insurance ISU (Industry Solution Unit) of Tata Consultancy Services for many meaningful discussions and suggestions on the topic of knowledge reuse. We also thank the anonymous reviewers of this paper for their constructive suggestions.

REFERENCES

[1] Scott Ambler on http://searchsoftwarequality.techtarget.com/news/article/0,289142,sid92_gci1277064,00.html

[2] http://searchsoftwarequality.techtarget.com/generic/0,295582,sid92_gci1351096,00.html

[3] http://searchsoftwarequality.techtarget.com/news/article/0,289142,sid92_gci1277064,00.html

[4] http://www-01.ibm.com/software/rational/jazz/

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

[6] Nigel Shadbolt, Tim Berners-Lee and Wendy Hall, "The Semantic Web Revisited";http://eprints.ecs.soton.ac.uk/12614/1/ Semantic_Web_Revisted.pdf.

[7] S. Lohmann, S. Dietzold, P. Heim and N Heino, A web platform for Social Requirements Engineering, LNI, Proc. Software Engg, Workshopband, 29th IEEE Conf. on Decision and Control, San Francisco, CA, 1990, 500-506. Lecture Notes in Informatics (LNI) – proceedings of SENSE 09 Workshopband, Series of the Gesselshaft fur Informatik (GI), Bonn, 2009 , Vol 150, 309-315

[8] A. Hanneman, C. Hocken and R. Klamma, Community driven Elicitation of Requirements with Entertaining Social Software, LNI, Proc. Software Engg, Workshopband, 29th IEEE Conf. on Decision and Control, San Francisco, CA, 1990, 500-506. Lecture Notes in Informatics (LNI) – proceedings of SENSE 09 Workshopband, Series of the Gesselshaft fur Informatik (GI), Bonn, 2009, Vol 150, 317-328.

[9] B. Decker, E. ras, J. Rech, P. Jaubert, M. Rieth, Wiki based stakeholder participation in Requirements Engineering, IEEE Software, 24(2), 2007, 28-35

[10] J. Whitehead, Collaboration in Software Engineering: A roadmap, Proc.IEEE, Future of Software Engineering, 2007, 214-225

[11] S. Ghaisas, A method for identifying unobvious requirements in globally distributed software projects, Software Engineering in Social Software Environments., Lecture Notes inInformatics (LNI) - proceedings, Series of the Gesselshaft fur Informatik (GI), Bonn 2009, Vol.150 edited by Munch J. and Liggesmeyer P., Mar. 2009, pages 297-308. In Proc. SENSE09, Fraunhofer Institute Experimental Software Engineering, Kaiserslautern, Germany

[12] Kiu CC, Lee CS, OntoDNA: Ontology Alignment Results for OAEI 2007, In proc. 6th International Semantic Web Conference(ISWC) and 2nd Asian Semantic Web Conference (ASWC), 196-205, 2007.

[13] Dao, Simpson. Measuring Similarity between Sentences. http://wordnetdotnet.googlecode.com/svn/trunk/Projects/Thanh/Paper/WordNetDotNet_Semantic_Similarity.pdf

[14] Ghazvinian A, Noy N.F., Jonquet C., Shah N., Musen M.A., What four million mappings can tell you about two hundred ontologies?, LNCS, Vol. 5823, 229-242, 2009

[15] http://www.w3.org/TR/owl-ref/

[16] A Ankolekar, M. Krotzsch, T. Tran, D. Vrandecic, The two cultures- mashing up web 2.0 and the semantic web, Proc., WWW2007, May 2-8 2007, Banff, Alberta, Canada, 825-834

[17] R. Seater, D. Jackson, R. Gheyi, Requirements progression in problem frames: Deriving Specifications from Requirements, Requirements Engineering Journal (RIJ), 12(2) 2007, 77-102.

[18] Haruhiko Kaiya, Motoshi Saeki, Ontology based Requirements Analysis: Lightweight Semantic processing Approach, Proc., International Conference on Quality Software, QSIC 2005.