Crowd Sourcing Rules in Agile Software Engineering to Improve Efficiency using Ontological Framework

Himanshu Pandey[1], Dr. Santosh Kumar[2], Dr. Manuj Darbari[3]

[1](*Research Scholar, MUIT, Lucknow, India)*
[2](*Associate Professor, MUIT, Lucknow, India)*
[3](*Associate Professor, BBDNITM, Lucknow, India)*

Abstract

Business Rule Management System provides the necessary seeds for the planning, implementing, verifying and validating the Agile Requirements. The BRMS model needs to be modified in a way that organizational growth runs parallel with the intrinsic expansion in the number of User Requirements in Agile Development. This growth in Requirements or Rules in Agile Software Development is an obvious overhead that needs to be managed properly considering its sprint nature. A Semantic approach is followed by design and maintenance of an Ontology called RAgile. The ontology is developed in 'Protégé 5 having inherent capability f Ontology Merging in case of disparate Rule files. User requirements that are drawn into the Rules or Policies depend upon the features users expect of the Agile System.

Keywords: crowd sourcing rules, agile software, engineering, improve efficiency

I. Introduction

The Three quality models of Agile Development[1]: The stakeholder model proposed in the introduction sits between internal and external quality as a source of uncertainty linked to the user and their device(s). If the QA/QC framework is aimed at producing metadata about spatial data quality in the form of the ISO 19157 (the producer quality model), this process requires other types of quality elements. Table 1 describes an overview of quality elements that are considered as part of the QA process, with a focus on active volunteers.

Table no 1: Three Quality Model of AGILE

Quality element	Definition
Vagueness	Inability to make a clear-cut choice (*i.e.*, lack of classifying capability)
Ambiguity	Incompatibility of the choices or descriptions made (*i.e.*, lack of understanding, of clarity)
Judgement	Accuracy of choice or decision in a relation to something known to be true (i.e., perception capability and interpretation)
Reliability	Consistency in choices / decisions (*i.e.*, testing against itself)
Validity	Coherence with other people's choices (*i.e.*, against other knowledge))
Trust	Confidence accumulated over other criterion concerning data captured previously (linked to reliability, validity and reputability)

II. Literature Review

Crowdsourcing is a means of data collection has produced previously unavailable data assets and enriched existing ones, but its quality can be highly variable [1]. This presents a state where challenges to potential end users are concerned with the V&V and QA Activities of the data collected. Being able to quantify the uncertainty, define and measure the different quality elements associated with crowdsourced data, and allows methods for dynamic assignment and enforcement of Agile Rules using the concept of merged Ontology which is the scope of this paper. Types of crowdsourcing range from highly organized methods of harnessing the collective power of the crowd, for example Amazon's Mechanical Turk (Kittur, et al. 2008) and other monetary reward based schemes (Horton and Chilton, 2010), to volunteered geographic information (VGI) such as Open StreetMap (Haklay and Weber, 2008).[1]

Ontologies: The word ontology was taken from philosophy where it means "study of the nature of being". The most common definitions state that an ontology is a specification of a conceptualization [7] or that an ontology is the shared understanding of some domain of interest [14]. Ontologies provide domain representation for multi-agent systems. It defines everything comprehensively in the domain. Ontology contains classification, properties, objects, literals and most importantly relationships between individual elements. Ontology provides the vocabulary for the messages passed between communicating agents. It specifies meaning to agent communication. This makes it easy to combine and add heterogeneous agents at runtime in order to function together even if they are unknown to their peers. The ontological support in the multi agent system proffers reasoning. XML provides syntax. RDF(S) provides basic relational language and simple ontological primitives. OWL offers powerful decidability in an ontology language. But SWRL, Semantic Web Rule Language combining OWL and RuleML extends OWL. In this implementation of MAS, Protégé 5.0 is used as an ontology design toolkit.

AGILE Development: XP Extreme Programming

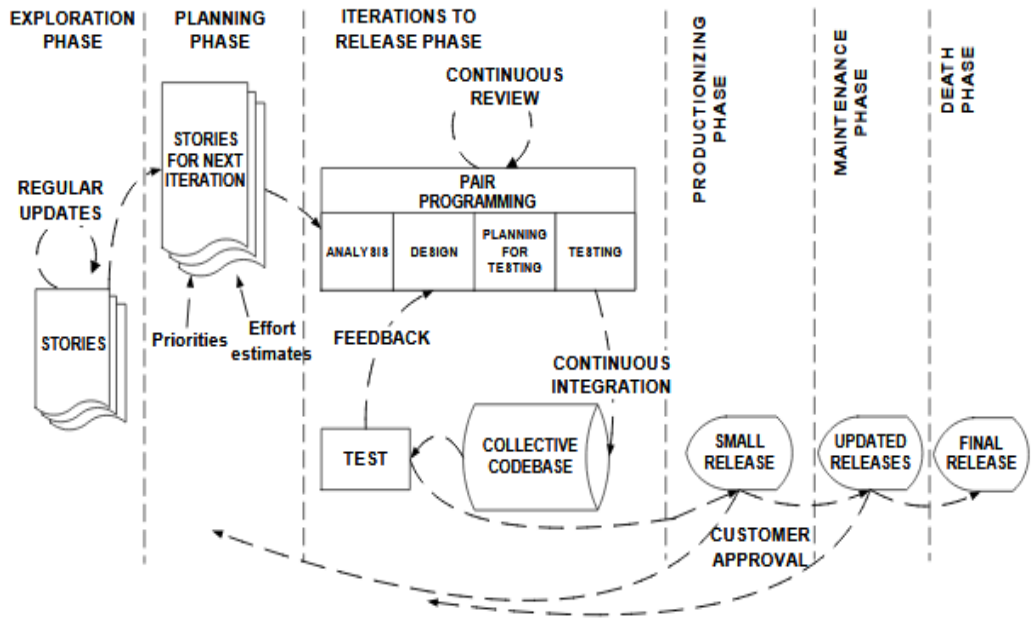

Figure no 1: XP Programming in Agile

BRMS: The business rules approach has a much broader scope than the expert systems approach [2].
Business Rules are defined as: "a formal way of managing and automating an organization's business rules so that the business behaves and evolves as its leaders intended" [et. al.] von Halle 2001.

Roles and responsibilities: There are six identifiable roles in Scrum that have different tasks and purposes during the process and its practices: Scrum Master, Product Owner, Scrum Team, Customer, User and Management. In the following, these roles are presented according to the definitions of Schwaber and Beedle(2002).

III. Research Work

Major Issues in Agile Development and Crowd Sourcing are:

Issue 1: Organizations need to know which business rules they are using, and whether they are using them consistently.
Issue 2: Organizations need to describe the business rules that are embodied in their information systems in a way that all stakeholders can understand, and need a way of ensuring traceability between those rule descriptions and the actual implementations of the rules.

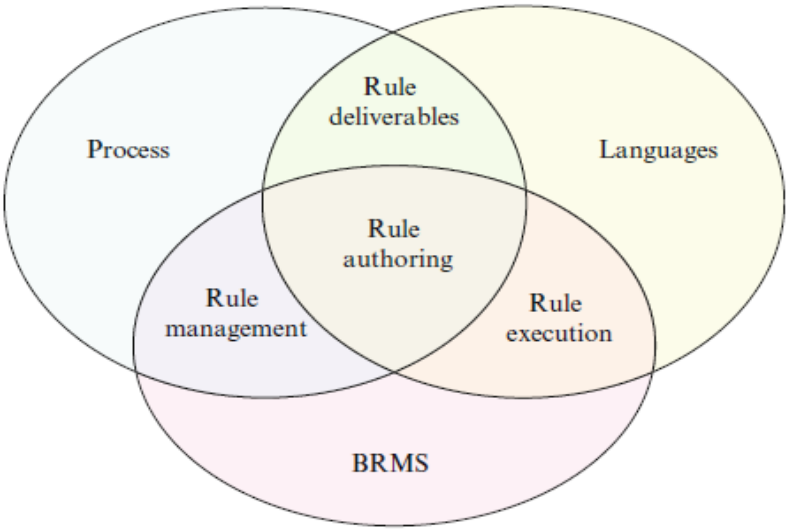

Figure no 2: BRMS as Venn Diagram

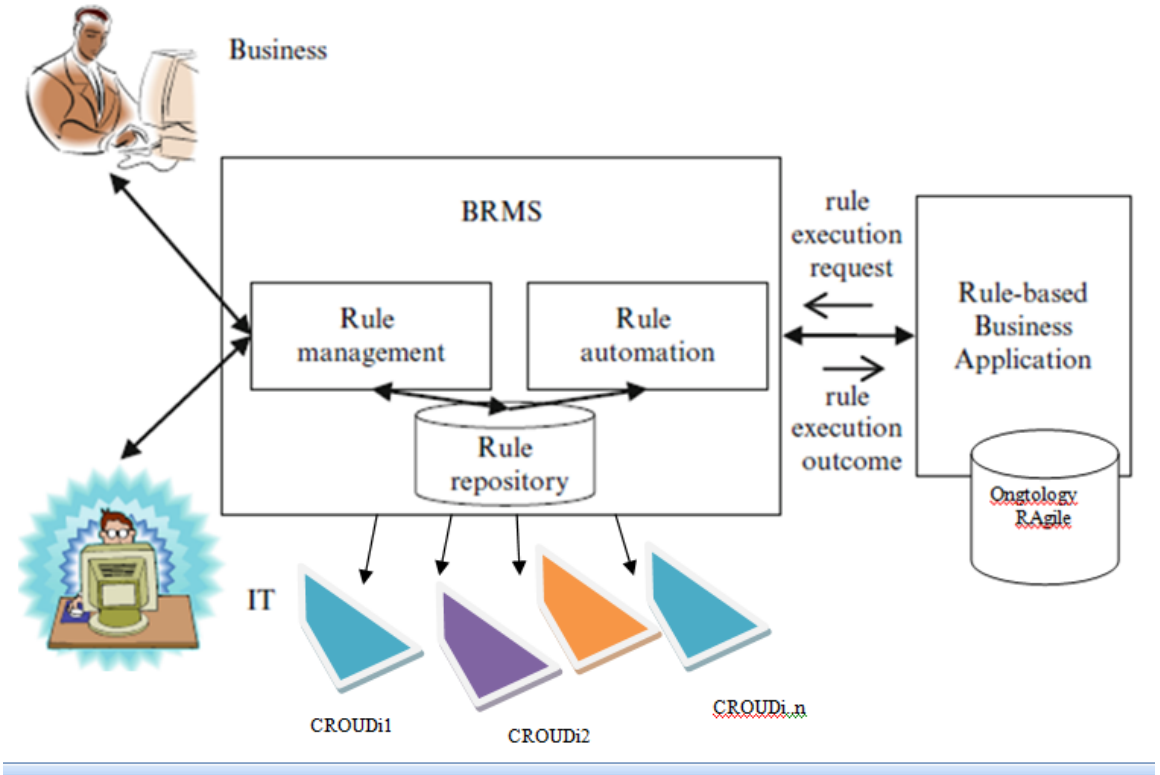

Figure no 3: OPFACS (Meta Model)

Case Study: Genomics & DNA Sequencing et al [Kant] 2017

As a case study for implementing inter-agent communication, Dna pattern search in existing varied and heterogeneous Genome Repositories is chosen.
From the viewpoint of a computer science researcher the important concerns regarding sequencing a DNA are:
The Genome contains the DNA and the whole genetic structure. This genetic structure keeps the complete information necessary for an organism to live its life. This genetic material is similar in many organisms. Biologists and Life Science's experts sequence DNA in the form of sequences of four characters, A, C, T and G. This is done in order to represent a DNA programmatically.

1. Thymine (T)
2. Cytosine (C)
3. Guanine (G)
4. Adenine (A)

A DNA structure is made up of the combinations of these four elements. Since the DNA of an organism is similar to other organisms, conditions arise when Biologists look for similarity in DNAs like in areas like Pharmacy. There are requirements when a particular extract of a DNA has to be searched in disparate and heterogeneous data sources ranging from plain text files to plethora of databases acting as repositories of fully sequenced DNAs. The full discussion on DNA Sequencing is out of the scope of this paper. As a researcher, my quest deals only with inter-agent communication focusing validated policies and verified message exchange between agents.
The above case study is used to implement the agile rules as shown below:
As part of the Architecture Design, policies stating the rules that the agents must stick to, are also derived. The policies through requirements stated in OPFACS can be stated as:

R1: Maximum Time Limit: 30ns
R2: ONLY A, C, T, or G characters can be used to represent a DNA Pattern.
R3: The search string cannot contain white spaces, hyphens or any special character or numbers.
R4: Max Length of a search pattern can be set but assumed to be 1024 chars.
R5: Break-up size of search pattern can be set but assumed to be 11 chars.
R6: Patterns not conforming to P1, P2, P3 and P4 will be immediately discarded.
R7: Total time from query submission to result display can be maximum 8.00 secs.
R8: Queries from Agile to crowds failing to meet R6 will be held for resubmission.

This paper tries to implement the issues discussed above as part of the framework developed by the authors and implement the Agile Rules for crowd sourcing.

Issue1 (Organizations need to know which business rules they are using, and whether they are using them consistently.) This Issue is Implemented Using Requirements V&V

V&V

R1: Maximum Time Limit: 30ns
RULE(?R) ^ hasCPUTIME(?R, ?t) ^ swrlb:lessThan(30^^xsd:short, ?t) -> valRULES(?R, ?t) ^ VALIDATEDRULES(?R)

In the above code SWRL Formal language is used to test, verify and validate the Agile Rules/Requirements in Crowd Outsourcing. R, one of the instances of RULE Class is testified with any range of valid and invalid rules and then only the successfully verified and validated rules are selected and pushed to the val_RULES Property in the Ontology designed specifically for this. The above RULE example R1 only considers a single rule that: "One of Agile Processes need 30 nano seconds as maximum to get executed"

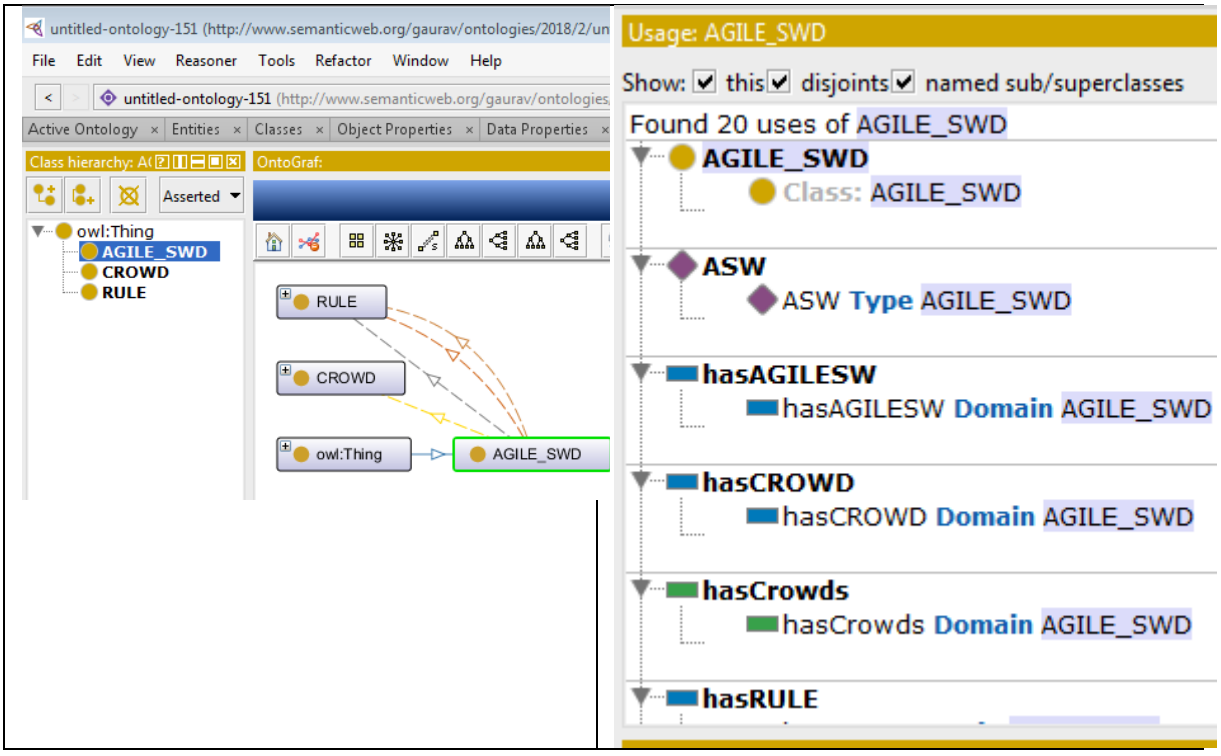

Figure no 4a: Basic Rule Ontology

Figure no 4b: The Agile SWD Class

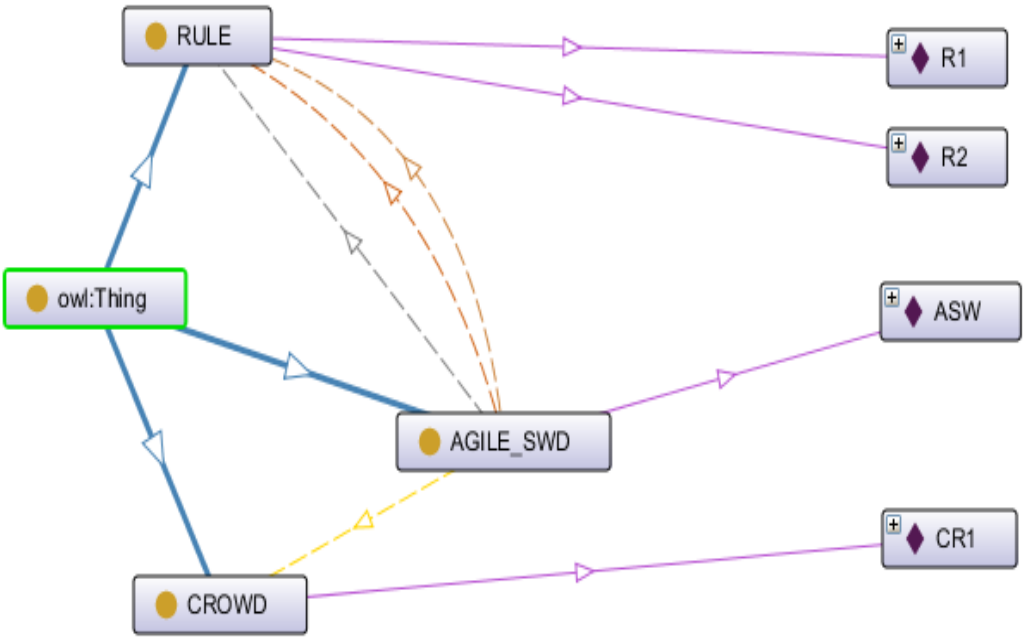

Figure no 5: The above ontology is expanded to include Agile Rules and its relationship with Rules

The above ontology is expanded to include Agile Rules and its relationship with Rules. It is pertinent to mention that these Rules are in fact derived from the requirements of the end users and hardcoded in the ontology for validation purpose. These requirements are extracted from the SRS itself and verified first at the manual level then injected as rules in to the ontology for Validation Purpose. Rule R1 is one of the 8 Rules that are inferred by the DROOLS reasoner that comes along with Protégé 5. The reasoner continuously checks for any discrepancy in the formulated rules. Only the individuals that abide by the Rule Engine Inference mechanism are moved to the VALIDATEDRULES Class. This parallel checking is conducted like a chronograph and new rules might be coming on the fly and validated accordingly as in the figure below.

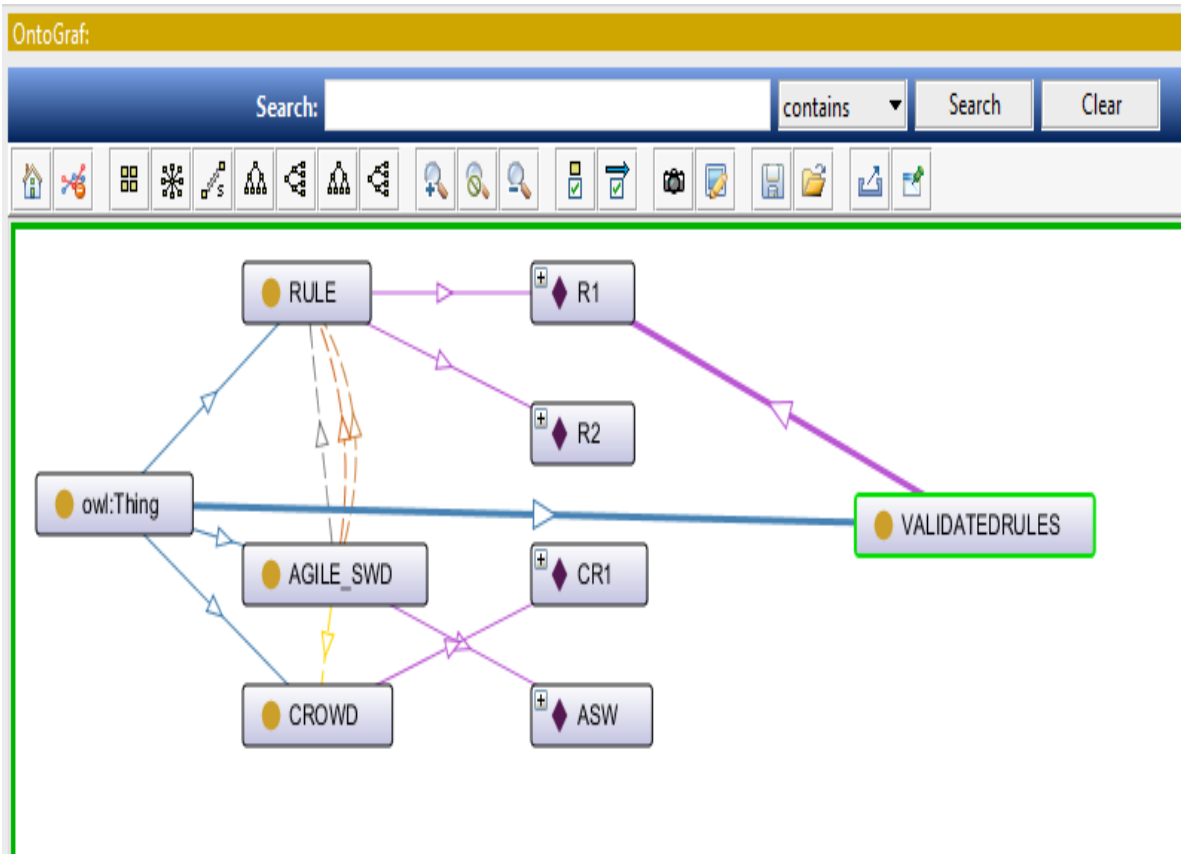

Figure no 6: Validation of Rules.

Table no 2: Status of issue 1 & 2

Issue 1	Implemented Using Requirements V&V	Implemented
Issue 2	BY Use of A Meta Model termed OPFACS (Open Process Framework for Agile Crowd Sourcing.	Implemented

Merged ontology

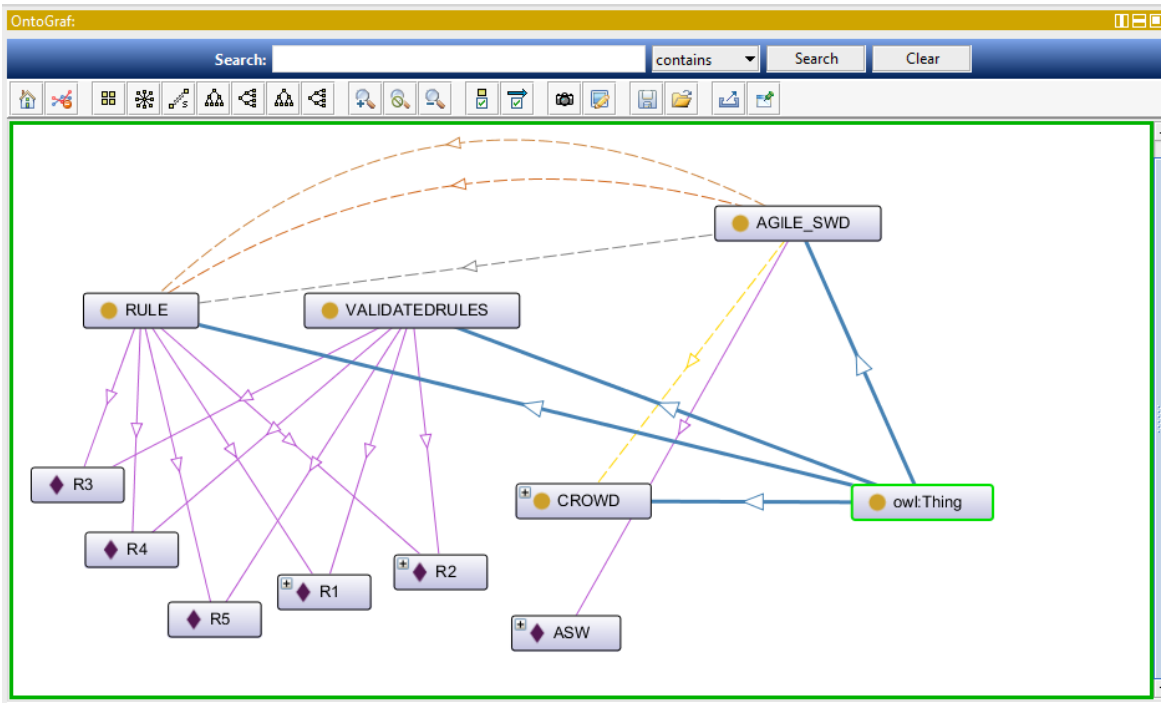

Figure no 7: Merged ontology with newly discovered and validated rules

Rules represented as Requirements are dynamic entries by the SRS (modification) from end user, namely, R3, R4, R5 are added to the ontology. Onto Graph is used for representing the relationships between classes, properties and individuals.

IV. Implementation

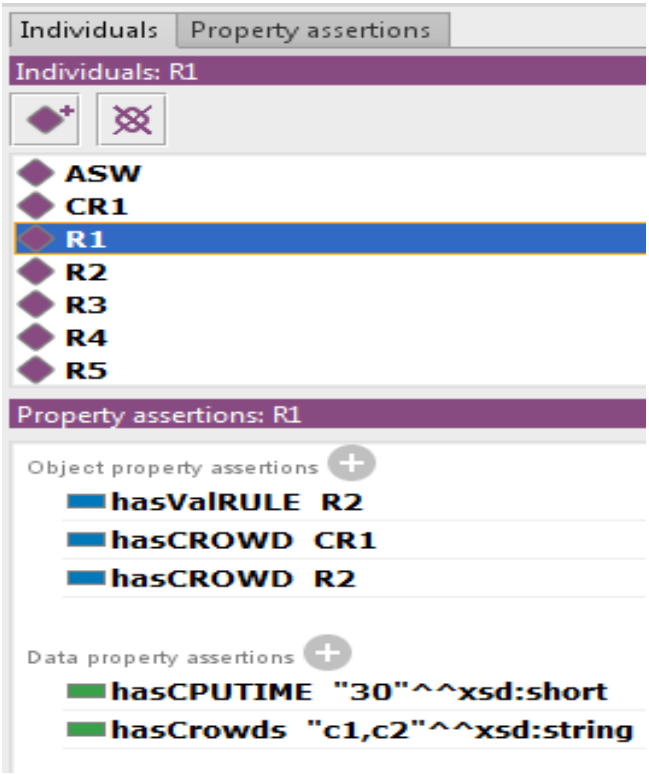

Figure no 7a: Implementation of OPFACS in PROTÉGÉ 5.2 using DROOLS Reasoning.

The above snap shows that Rule R1, derived from first requirement does not have "valRule" Property set for itself. This is because the Estimated CPU Time (t) by the Requirement R is greater than 30 ns.

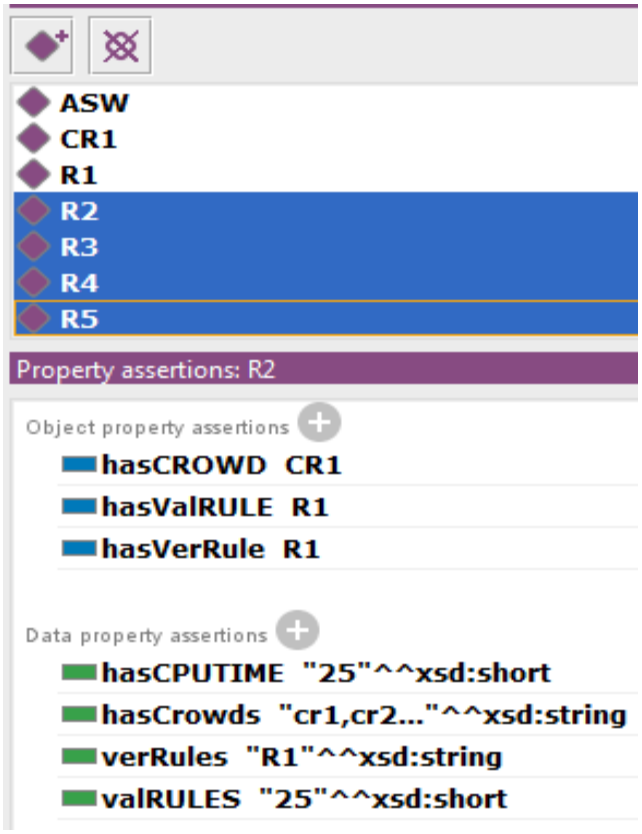

Figure no 7b

On the contrary, the above figure displays R2, R3, R4, R5 requirements / rules and has valRULES property set because they come under the criteria of t being greater than 30.
In short, this implementation selects only those valid rules that have CPU Time less than 30 ns.

V. Conclusion

The authors in this paper have developed a meta-model named, OPFACS that with the help of semantics and ontological merging proves to implement, verify and validate the Agile Rules. DROOLS Reasoner with Protégé 5.2 is used to model and verify the crowd sourcing for satisfaction and attainment of the Rules in distributed environment. Merging of the ontologies in different versions of the same original RDF/OWL file is done by the Protégé's inbuilt merging tool.

References

[1]. Dignum, V. A (2004) *Model for Organizational Interaction: Based on Agents*, Founded in Logic. PhD thesis, Utrecht University.

[2]. Zambonelli, F., Jennings, N.R., and Wooldridge, M.J. (2001, June) *Organisational Rules as an Abstraction for the Analysis and Design of Multi-Agent Systems.* IJSEKE. 11(3) pp. 303-328,June 2001.

[3]. DeLoach, S. A. (2002) *Modeling Organizational Rules in the Multiagent Systems Engineering Methodology*. Proc of the 15th Canadian Conference on Artificial Intelligence.

[4]. Cossentino, M., Gaglio, S., Garro, A., Seidita, V. (2007): *Method fragments for agent design methodologies: from standardization to research. Intl Jnl of Agent-Oriented Software Engineering*, 1, 91–121

[5]. DeLoach, S.A., Garcia-Ojeda, J.C. (2010*) O-MaSE: a customizable approach to designing and building complex, adaptive multiagent systems*. International Journal of Agent-Oriented Software Engineering. 4, 244–280

[6]. Ferber, J., Gutknecht, O., Michel, F. (2003) *From agents to organizations: an organizational view of multi-agent systems. In: Giorgini, P., Muller, J.P., Odell, J. (eds.), Agent-Oriented Software Engineering* IV, LNCS Vol. 2935, pp. 214-230, Springer, Berlin

[7]. Cossentino, M., Gaud, N., Hilaire, V. Galland, S. Koukam, A. ASPECS: an agent-oriented software process for engineering complex systems. Journal of Autonomous Agents and Multiagent Systems Systems. 20, 260–304 (2009)

[8]. DeLoach, (2008) S.A., Oyenan, W., Matson, E.T. *A capabilities based model for artificial organizations. Autonomous Agents and Multiagent Systems*. 16, 13–56

[9]. DeLoach, (2010) S.A., Miller, M. *A goal model for adaptive complex systems.* International Journal of Computational Intelligence: Theory and Practice. 5, 83–92

[10]. DeLoach, S.A., Valenzuela Jorge, L. (2006) *An agent environment interaction model*. In: Padgham, L., Zambonelli, F. (eds.) Agent-Oriented Software Engineering VII: 7th International Workshop, AOSE 2006. LNCS Vol. 4405, pp. 1-18. Springer: Heidelberg

[11]. Garcia-Ojeda, J.C., DeLoach, S.A. Robby. (2009) *agentTool process editor: supporting the design of tailored agent-based processes.* In: Proceedings of the 2009 ACM symposium on Applied Computing (SAC '09), pp. 707-714, ACM: New York

[12]. Garcia-Ojeda, J.C., DeLoach, S.A., Robby (2009). *agentTool III: from process definition to code generation*. In: Decker, Sichman, Sierra, and Castelfranchi (eds.) Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2 (AAMAS '09), pp. 1393-1394, International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC.

[13]. Candelaria E. Sansores_, Flavio Reyes, Hector F. Gómez_, Juan Pavón† and Luis E. CalderAłn-Aguilera.

[14]. Seidita, V., Cossentino, M., Gaglio (2006) Proceedings of the Federated Conference on Computer Science and Information Systems pp. 675–682BioMASS*: a Biological Multi-Agent Simulation System*, S. A repository of fragments for agent systems design. In: Proc. of the 7th Workshop from Objects to Agents (WOA 2006), pp. 130-137

[15]. Scott A. DeLoach, *Software Engineering for Multi-Agent Systems* IV Volume 3914 of the series Lecture Notes in Computer Science pp 109-125 Engineering Organization-Based Multiagent Systems

[16]. K. Bryson, M. Luck, M. Joy, and D. Jones 2000. *Applying agents to bioinformatics in geneweaver*. In Proceedings of the Fourth International Workshop on Col laborative Information Agents.