Mehran University Research Journal of Engineering and Technology

https://doi.org/10.22581/muet1982.2202.13

2022, 41(2) 135-145

Ontological automation of software essence kernel to assess progress of software project

Farooq Ali [a,*], Asif Raza [b], Muhammad Munwar Iqbal [a], Tahira Nazir [a]

[a] *Department of Computer Science, University of Engineering and Technology, Taxila, Pakistan*

[b] *Department of Computer Science and Information Technology, University of Mianwali, Pakistan*

[*] Corresponding author: Farooq Ali, Email: farooq.ali@uettaxila.edu.pk

Received: 24 December 2019, Accepted: 16 December 2020, Published: 01 April 2022

KEYWORDS

Alpha

Essence

Activity Space

Software Health

ABSTRACT

Managing a software project with a large number of requirements is a challenging task, a lot of effort and time is required to measure the software's progress and health. To tackle this, software development organizations look for different development processes that would best assist them. Recently, software essence is used to measure the progress and health of software development. However, these are applied manually to assess the development and fitness of the system. Therefore, it is very difficult for software organizations to access the required information quickly. The aim of this study is the automation of Software Essence Kernel through Ontology Development to quickly measure the progress and health of the system. Experimental work is done with a hypothesis that if software essence kernel is automated, then one can quickly measure the progress and health of the system. The results of this study argue that software health of the system can be quickly and easily measured if the ontology is automated.

1. Introduction

The software development method contains a large number of practices, procedures, and techniques which semantically help in the development of projects. In past decades, different software development methods were proposed including model-driven to agile approaches. Currently, software essence is used to measure the health of software projects [1]. However, these essences are applied manually to measure the health of a software project [1-4]. The software essence is significant in numbers which were discussed in subsections. To maintain large data, ontology is one of the best options to use. [5-6] used ontology for different purposes. Therefore, we have used ontology to maintain software essence and by using the same ontology, we have done reasoning to measure the health of software projects.

1.1 Essence of Software Engineering

The essence is the foremost and well-known output from the Software Engineering Method and Theory (SEMAT) community. As with the development of software techniques, the way the people work with software methods has been completely changed [1-2]. So, in order to deal with it, the Software Engineering Method and Theory (SEMAT) community was set up by Bertrand Meyer, Ivar Jacobson, and Richard Soley in September 2009. The SEMAT community-produced it's the most distinguish output named as Essence [2]. There were the two main goals of SEMAT community; (1) Searching a kernel of extensively approved elements, and (2) Describing a concrete theoretical basis. The aim of this community was to introduce essence language and essence kernel that can easily be scaled, learned,

used, modified and extended. Moreover, these languages and kernel can help the users to quickly describe the basics and fundamentals of their existing used methods. Furthermore, the introduction of essence language will assist the people to compare, evaluate, analyse, simulate, adapt, and compose their methods with each other and with academics' researchers as well. The Essence specification defines a domain-specific language (i.e., the Essence language) and a kernel (i.e., the Essence kernel) in order to support the definition and the enactment of methods in the context of software engineering endeavours. While the language defines the theoretical and conceptual base to define and describe software engineering methods, practices and kernels, the kernel aims to provide a set of essential and universal elements that form the common ground of software engineering.

The Essence kernel captures the idea of a common ground based on the things we always have, the things we always do, and the skills we still need in order to conduct software engineering endeavours. The objective of the kernel is to provide a set of universal elements to define, use and adapt methods and practices supporting the daily practices of the software development team in a dynamic way while fostering communication and collaboration. To establish a common ground, the essence kernel addresses the three areas of concern which are customer, solution and endeavour. Moreover, within these areas of concern, the kernel defines activity spaces, alphas and competencies.

1.1.1 Areas of concern

The kernel is structured alongside three areas of concern: The solution area of concern, the customer area of concern and the endeavour area of concern [3]. The solution area of concern addresses the specification, design and implementation of the software system. Within this area, the team establishes a shared understanding of the requirements and implements, builds, tests deploy and support the software system. The customer area of concern deals with the concrete utilization and operation of the software system to be developed. Within this domain, the team understands the opportunity to build software, i.e., the value the software system provides to the other stakeholders. The domain area of endeavour engages in the concerns of the team and the management of its work. Within this area, the team is formed and work is being planned and organized; the team advances the job in alignment with the agreed working approach.

1.1.1.1 Alphas: Alphas can be regarded as one of the most central concepts of the kernel. Alphas symbolize the essential items to work with while conducting a software engineering endeavour. As such, alphas form the base of the usual ground for the illustration of methods and practices of software engineering. The kernel defines a collection of seven alphas as shown in Fig. 1. Each alpha belongs to a specific area of concern. The customer domain area specifies Opportunity and Stakeholders' alpha. Within the solution domain area, the Software System alpha and Requirements are defined. The domain area of endeavour includes the Team, the Way of Working and the Work alphas. During the execution of an endeavour, the team progresses the alphas from an initial state to a target state. Therefore, each alpha contains a set of standardized states. For example, the Opportunity alpha progresses through the states identified, value established, the solution needed, addressed, viable and benefited accrued.

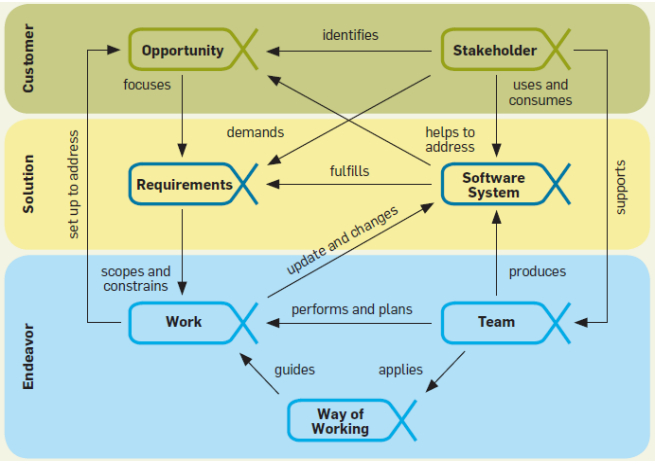


Fig. 1. Alphas for software essence kernel [2]

Alpha states allow us to assess the progress and health of the alphas during the execution of the endeavour. Furthermore, the states allow to determine where the team currently stands and how much work is required to complete the project. The assessment of each state is supported by a defined checklist associated with each state. For example, regarding the alpha Opportunity and its state Value Established, a checklist consisting of five items summarizes the required criteria to achieve that state.

1.1.1.2 Activity spaces: Activity spaces represent placeholders for the essential activities in software engineering endeavours [3]. Each activity space has a set of objectives and is related to certain kernel alphas which are required to achieve. Activity Spaces describes the challenges a team may face while development and maintenance of software systems. Moreover, activity

spaces define the things that will required to complete them. Further, they balance the alphas and present an activity-based vision on software engineering. Activity spaces against area of concerns are shown in Fig. 2.

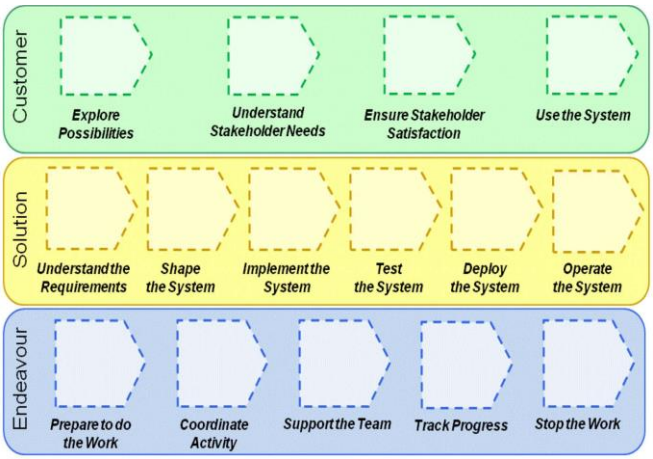


Fig. 2. Activity Spaces against area of concerns [2]

1.1.1.3 Competencies: The kernel competencies complement the kernel alphas and activity spaces with the significant competencies required to conduct software engineering endeavours. The defined kernel competencies consist of communication, engineering and management capabilities. The following image depicts the specified competencies across the three areas of concern.

Each competency can be assessed by five levels of achievement that build upon each other. Team members with competencies at level 1 ('Assists') demonstrate an essential understanding of required perceptions and can track instructions. At level 2 ('Applies'), the concepts are applied in simple contexts based on first experiences. Level 3 ('Masters') defines the competency to apply the concepts in most circumstances. Team members possessing this competency are considered to have enough knowledge skills to perform tasks with no supervision. Competencies at level 4 ('Adapts') allow judging how and when to apply the concepts in additional multifaceted contexts. Competencies at Level 5 ('Innovates') represent recognized experts who extend concepts, apply them to new contexts, and inspire others. Table 1 shows the essence area of concerns along with their respective alpha, activity spaces, and competencies.

1.2 Ontology

Ontology is "a collection of concepts and categories in a domain area. It represents inter-relations and the properties of concepts and categories." Many authors define the number of approaches to create an ontology for different purposes. Most common approaches are top-down, bottom-up and combination development. In 2009 Yajing Zhao and latterly Hans-Jörg Happel and Stefan Seedorf discussed these approaches and stated that it is difficult for software engineers to access the required information manually if the information is in a large amount. Furthermore, they state that semantic web techniques e.g. ontology can help to maintain a large amount of data [4].

Table 1

Essence Kernel [2]

Area of Concern	Alphas	Activity Spaces	Competencies
Customer	- Opportunity - Stakeholders	- Explore possibilities - Understand stakeholders needs - Ensure stakeholder satisfaction - Use the system	Stakeholder Representation
Solution	- Requirements - Software system	- Understand the requirements - Shape the system - Implement the system - Test the system - Deploy the system - Operate the system	- Analysis - Development - Testing
Endeavour	- Work - Team - Way of working	- Prepare to do the work - Coordinate activity - Support the team - Track progress - Stop the work	- Leadership - Management

1.3 Research Question

The research question of this study is 'how to automate Software Essence Kernel to measure the health of

software projects?'. To answer this question, we have divided the work into following two objectives.

1. To query the current status of the software project in an automatic manner.

2. Through reasoning, quantify the progress of software development.

The rest of the paper is organized as follows. In section 2, we have discussed related work. In section 3, the research methodology and research process are presented. In section 4, results and analysis are discussed. Section 5 is about the conclusion and future work of this study.

1.4 Research gap analysis

Literature shows that using ontology one can easily maintain a large amount of data. Naveen Malviya [13] and P. K. C. M. Wijewickrema [14] build ontologies to maintain data of different projects and proves that one can easily do reasoning and measure using it. Iaakov Exman [24] proposed an ontology of software kernel essence (SEMAT) but the ontology build by him is on the abstract level. He claimed that one can check the completeness of the project using the same if it extends it by adding sub-alphas.

No one yet has created ontology on low-level details to verify laakov Exman's claim. Ontology at a low level still requires attention.

2. Related Work

In 2009, Yajing Zhao [4] discussed different methodologies for Ontology-based software engineering. He stated that it is difficult for software engineers to access the required information manually if the information is in a large amount. Furthermore, Yajing Zhao stated that, by using Semantic web techniques, one could formalize a large informational data. By doing this, probability of availability, accessibility and reusability of informational data can be improved.

Hans-Jörg Happel and Stefan Seedorf [9] also provided a brief explanation of diverse ontology-based approaches in Software Engineering. They discussed the benefits of ontologies in software engineering and then a skeleton for categorizing the application of ontologies in Software Engineering is proposed. Some advantages discussed in this paper are; (1) Ontologies provide great flexibility as it is well compatible to merge information from different sources, (2) Ontologies offer a way to catch knowledge concerning some specific problem domain. iii) Ontologies provide background knowledge of the project that helps novels to query the system.

Research Studies discussed above, shows that it is difficult for software engineers to maintain huge data. To maintain large amount of data, ontologies are helpful. It supports that ontologies can be used to maintain a large amount of data.

P. K. C. M. Wijewickrema and R. C. G. Gamage [14] created an ontology for automatic document Dewey decimal classification system which splits information into subjects with a numeric identifier. They reduced the rapid growth of text-form information by converting the text in an organized form i.e. ontology. They classify their text manually and automatically, then they compare the results of both techniques. By comparing results, they said that both human and machine users would extract required information quickly and in the right way using the same ontology.

In 2011, Naveen Malviya, Nishchol Mishra, and Santosh Sahu [12] developed a university ontology. They stated that with the help ontology, one can view key concepts and their relationship with information. They focused on how ontology creates. Furthermore, they stated that one of the uses of ontologies is to properly specify the conceptualization and interrelations between instances and properties of Concepts. The main components used to develop ontology are classes, objects, relations and attributes. Classes are the elements that conceptualize components of a domain. Classes are usually organized in taxonomies that are associated to each other through relations, which can be taxonomic (thus defining the type of inheritance among superclasses or subclasses) or non-taxonomic (which can define any other type of relationship, such as part-of, cause-effect, etc.). Objects are the specific entities of a domain and are represented as instances of classes; they may have specific properties that are represented by the attributes of these classes. They proposed essential steps concerning screenshots display with mainly advance tools for ontology creation and editing i.e. protégé tool.

This paper proposed a procedure to develop ontology using protégé tool and advocates that one can use protégé tool to develop ontologies. Iaakov Exman [24] automates an ontology for software essence kernel (SEMAT). He performed an experiment to analyse that "In which logic and up to what degree does the kernel and its alphas, truly represents the essence of software engineering?" He stated that the kernel alphas can be

sighted as top-most ontology, certainly the Essence of Software Engineering. Further, he developed a top-level alpha ontology and claimed that one can ensure the completeness of software using the same. By extending the approach using sub-alphas and other Kernel entities one can check completeness of system and measure the health of the project.

3. Research Methodology

In this section, we have discussed hypotheses of study, research process, and data acquisition process.

3.1 Hypothesis

If the software essence kernel is automated, then one can quickly measure the progress and health of the ontology.

3.2 Research Process

First of all, we have collected data from Essence Kernel Alpha states and activity spaces (Fig. 3). Then ontology of software essence kernel was developed using the standard methodology of ontology construction and protégé tool [16]. After building the ontology, the results were validated by using Apache Jena Fuseki server through SPARQL queries. These queries were executed using rule-based inference [22].

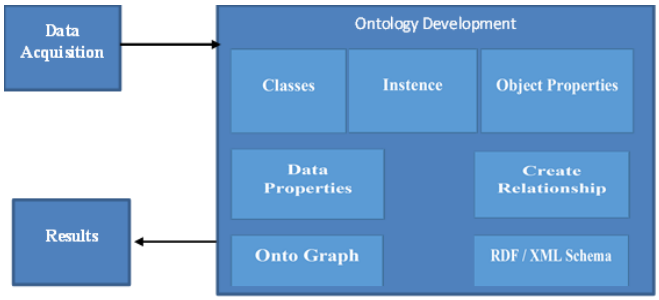


Fig. 3. Research Process for automation of software essence

3.3 Data Acquisition

We have collected all data for Essence Kernel Alpha states and Checklists the Alpha's states and Checklists from Semat book [23]. Table 2 shows the Kernel Alpha states, and Table 3 shows the completion criteria for activity spaces. Moreover, a completed ontological graph is shown in Fig. 2.

Table 2 shows the Alpha classes, their cross-ponding subclasses and the description. It shows that every class is a significant task or an activity which is subdivided into subclasses, tasks or activities. Every subclass has a description. One can see that Stakeholder is a class having subclasses or tasks like recognized, represented, involved and satisfied in use. Table 3 shows the completion criteria against each activity space. We have divided the evolution of the system among area of concerns. The division was done on the basis of activity spaces. There are 4, 4 activity spaces for customers and endeavour so 30 percent is given to each of them. The solution has six activity spaces so 40% was given to it. They are subdivided according to completion criteria.

4. Results and Validation

We have created an ontology-based on kernel alpha states. First of all, to create ontology, we have created classes, then subclasses of classes and individuals for classes and lastly, object property of individuals. Protégé tool was used to create an ontology. Fig. 2 shows a complete ontology graph. The Protégé is also an editor used for ontology having two languages; (1) OWL, (2) RDFS.

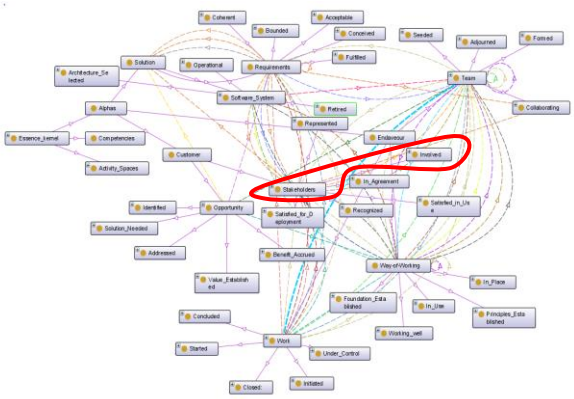


Fig. 4. Complete Ontology graph for software essence

Fig. 4 shows a complete ontology graph against table 2. It shows activities and sub-activities in a graphical view. Furthermore, it shows that Stakeholders of a project required to be recognized and have representation, must be involved in project development, must-have in the agreement of project and at last must be satisfied to end product. Similarly, every class has its subclasses or sub-activities. To link an individual with an object property, we used Boolean data type properties. In our proposed Ontology, Individuals of a stakeholder's subclass i.e. Involved are development team, investor, maintenance team and support team, their data properties were declared as Boolean, means their value will be true or false. True in case, when an individual is involved otherwise the value of object property remains false. Furthermore, we have applied different Sparql quires and have achieved one activity space "Explore_Possibilities" as the completion criteria for Explore_Possibility in Table 3 is to recognize all stakeholders and value established for the opportunity. We have run 8 queries to validate our results as shown in Table 4.

Table 2

Kernel alpha states of software essence

Stakeholders	Opportunity	Requirements	Software system	Team	Work	Way of working
Recognized: The stakeholders are recognized.	Identified: Social, business opportunity or commercial, is discovered and it can be completing by a software solution	Conceived: Requirements for software solution has been approved.	Architecture selected: A design is chosen that will help to address the key risks relevant to organizational constraints.	Seeded: Mission of solution team is clear, and they know how necessary to expand the team is in place.	Initiated: The work has been demanded.	Principles established: Principles and constraints form the way-of-working are recognized.
Represented: The processes for including the stakeholders in software project should be approved and the stakeholder representatives should have been selected.	Software solution required: Need of a software solution is verified	Bounded: The aim scope, and extent of the new software solution are clear.	Demonstrable: Working software is accessible which shows that the selected architecture is suitable for purpose and testing	Formed: Team has been inhabited with adequate dedicated people to begin the mission.	Prepared: The preconditions for beginning the work has been encountered	Foundation established: Tools and practices to form the foundation of the method of working are chosen and prepared for use
Involved: The recognized stakeholders are energetically participating responsibly in project work.	Value established: Most suitable and successful solution has been identified.	Coherent: Requirements offer a reliable explanation of the vital features of the new solution.	Usable: The system is useful and validates all the quality characteristics of a software system.	Collaborating: Team members are doing work collectively like a group.	Started: The work is progressing	In use: Some members are using and adapting, the way-of-working.
In-agreement: Stakeholders representatives are agreeing.	Viable: A solution which can be generated cheaply, quickly and enough successfully tackle the opportunity has been approved	Acceptable: Requirements that address a system which is appropriate to stakeholders	Ready: Whole solution is recognized for deployment.	Performing: Team members are working efficiently and effectively	Under control: work is going well, risks under control, productivity levels are enough to get an acceptable outcome	In Place: The team is utilizing the way of working to achieve their work.
Deployment Satisfaction: Nominal hopes of the stakeholder should have been achieved.	Addressed: The approved solution apparently completes the identified opportunity.	Addressed: Stakeholders requirements have been achieved	Operational: In an operational environment, the system is in use.	Adjourned: The team is adjourned and not responsible for taking out its mission.	Concluded: The work to generate the results has been achieved.	Working well: The method of working is doing well for the team.
User satisfaction: The system has successfully met nominal stakeholder requirements.	Benefit accrued: Effective usage and trade of the addressed solution is generating	Fulfilled: Addressed requirements satisfy the requirements of software solution	Retired: No support for the system		Closed: All remaining maintenance tasks have been completely, and work has been formally ended.	Retired: The way of working is no longer in use by the team

Table 3

Completion Criteria for Activity Spaces

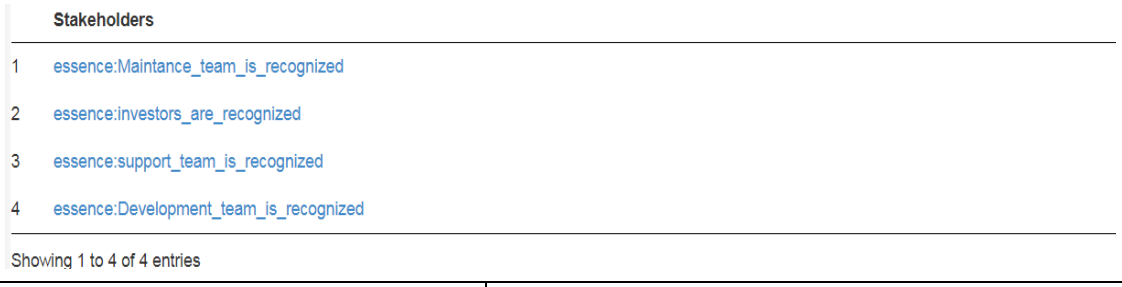
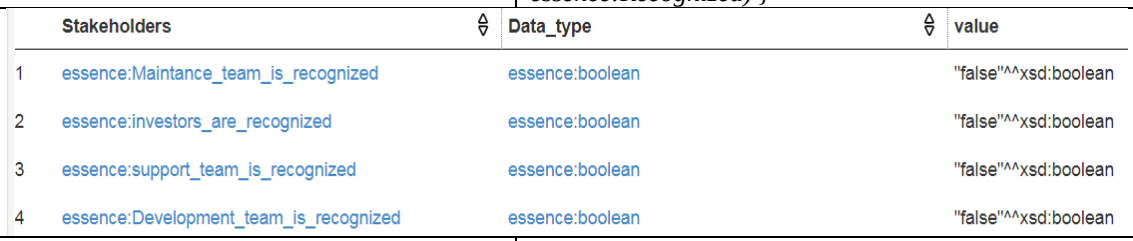
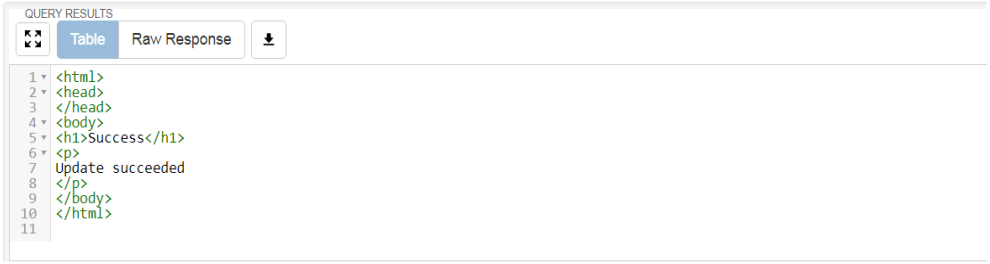
Area of concern with health percentage	Activity Spaces	Completion Criteria	Health Percentage progress while activity space achieved
	Explore possibilities	Stakeholders::Recognized, 3.75 Opportunity::Value established 3.75	7.5
	Understand stakeholder needs	Stakeholders::In agreement, 3.75 Opportunity::Viable 3.75	7.5
Costumer, 30 %	Ensure stakeholder satisfaction	Stakeholders::Satisfied for deployment, 3.75 Opportunity::Addressed 3.75	7.5
	Use the system	Stakeholders::Satisfied in use 3.75, Opportunity::Benefit accrued 3.75	7.5
	Understand the requirements	Requirements::Coherent 2.23	6.67
	Shape the System	Requirements::Acceptable 2.23, Software System::Architecture Selected 2.23	6.67
	Implement the System	Software System::Ready 6.67	6.67
Solution, 40%	Test the System	Requirements::Fulfilled 2.23, Software System::Ready 2.23	6.67
	Deploy the System	Software System::Operational 6.67	6.67
	Operate the System	Software System::Retired 6.67	6.67
	Prepare to do the Work	Team::Seeded 2.23, Way of Working::Foundation Established 2.23 , Work::Prepared 2.23	6.67
	Coordinate Activity	Team::Formed 3.75, Work::Under Control 3.75	7.5
	Support the Team	Team::Collaborating 3.75, Way of Working::In Place 3.75	7.5
Endeavour, 30 %	Track Progress	Team::Performing 2.5, Way of Working::Working Well 2.5, Work::Concluded 2.5	7.5
	Stop the Work	Team::Adjourned 2.5, Way of Working::Retired 2.5, Work::Closed 2.5	7.5

5. Results

After developing the ontology graph, we have validated the results using Apache Jena Fuseki server. Google chrome were used as an interface and SPARQL queries were applied to extract the output of ontology. These queries were executed using rule-based inference [22]. We have run 8 queries as shown in Table 4. In this table, queries, description of queries and results from defined ontology are presented. Queries for one activity state are presented i.e. Explore possibilities. Stakeholder and opportunity classes are discussed in the table as the same is the completion criteria for explore possibilities. Furthermore, it shows how one can use different queries to measure the current health of the project and also to update the health of the project.

Table 4

Queries to check and update software health

<table>
<tr><td>Description</td><td>Query</td></tr>
<tr><td>To search what are the instance of recognized class which is a subclass of stakeholder?</td><td>SELECT ?Stakeholders
WHERE {?Stakeholders rdf:type essence:Recognized ; }</td></tr>
<tr><td>Results</td><td> 
</td></tr>
<tr><td>To search which instances of recognized class are identified / recognized?
False value against each instance shows that no instance is identified.</td><td>SELECT ?Stakeholders ?Data_type ?value
WHERE { ?Stakeholders rdf:type essence:Recognized ;?Data_type ?value .
FILTER (?Data_type = essence:boolean && ?value != essence:Recognized) }</td></tr>
<tr><td>Results</td><td> 
</td></tr>
<tr><td>If any instance identified then instance check list is updated.
Delete and insert was used as there is no update keyword options in protégé tool.</td><td>Delete{ essence:Maintance_team_is_recognized essence:boolean essence:Recognized , "false"^^xsd:boolean }
INSERT {
essence:Maintance_team_is_recognized essence:boolean essence:Recognized , "true"^^xsd:boolean
}
WHERE
{ ?asdf rdf:type essence:Recognized; ?Data_type ?value .
FILTER (?value = "false"^^xsd:boolean) }</td></tr>
<tr><td>Results</td><td> 
</td></tr>
<tr><td>To search which instances are updated by query 3?
Results shows that only one value that Maintenance team was only recognized by query 3.</td><td>SELECT ?Stakeholders ?Data_type ?value
WHERE { ?Stakeholders rdf:type essence:Recognized ;?Data_type ?value .
FILTER (?Data_type = essence:boolean && ?value != essence:Recognized) }</td></tr>
</table>

Results

	Stakeholders	Data_type	value
1	essence:Maintance_team_is_recognized	essence:boolean	"true"^^xsd:boolean
2	essence:investors_are_recognized	essence:boolean	"false"^^xsd:boolean
3	essence:support_team_is_recognized	essence:boolean	"false"^^xsd:boolean
4	essence:Development_team_is_recognized	essence:boolean	"false"^^xsd:boolean

To search all instances of Recognized are updated?
After updating all the instances of recognized by using query 3, the value of all instances is 'true'. It means all stakeholders are recognized.

SELECT *?Stakeholders ?Data_type ?value*
WHERE *{ ?Stakeholders rdf:type essence:Recognized ;?Data_type ?value .*
FILTER *(?Data_type = essence:boolean && ?value != essence:Recognized) }*

Results

	Stakeholders	Data_type	value
1	essence:Maintance_team_is_recognized	essence:boolean	"true"^^xsd:boolean
2	essence:investors_are_recognized	essence:boolean	"true"^^xsd:boolean
3	essence:support_team_is_recognized	essence:boolean	"true"^^xsd:boolean
4	essence:Development_team_is_recognized	essence:boolean	"true"^^xsd:boolean

To search value of instance Value-Establish (2nd completion criteria for explore possibilities).

SELECT *?Oppertunity ?Data_type ?value*
WHERE *{ ?Oppertunity rdf:type essence:Value_Established ;?Data_type ?value .*
FILTER *(?Data_type = essence:boolean && ?value !=essence:Value_Established) }*

Results

	Oppertunity	Data_type	value
1	essence:idea_establish_value	essence:boolean	"true"^^xsd:boolean

When stakeholders are recognized and opportunity is created, all the instance of recognized and value established are identified then it means that explore possibilities activity space is completed.

SELECT *distinct ?Stakeholders ?value ?Value_Established ?valu ?result*
WHERE*{ ?Stakeholders rdf:type essence:Recognized ;?Data_type ?value .*
FILTER *(?Data_type = essence:boolean && ?value != essence:Recognized) ?Value_Established rdf:type essence:Value_Established ;?Dta_type ?valu .*
FILTER *(?Dta_type = essence:boolean && ?valu !=essence:Value_Established) bind(IF(?value = "true"^^xsd:boolean && ?value="true"^^xsd:boolean, "Completed" , "Not Yet Completed") AS ?result) }*

Results

Stakeholders	value	Value_Established	valu	result
essence:Maintance_team_is_recognized	"true"^^xsd:boolean	essence:idea_establish_value	"true"^^xsd:boolean	"Completed"
essence:investors_are_recognized	"true"^^xsd:boolean	essence:idea_establish_value	"true"^^xsd:boolean	"Completed"
essence:support_team_is_recognized	"true"^^xsd:boolean	essence:idea_establish_value	"true"^^xsd:boolean	"Completed"
essence:Development_team_is_recognized	"true"^^xsd:boolean	essence:idea_establish_value	"true"^^xsd:boolean	"Completed"

To search the achieved activity spaces.
Results shows that only explore possibility is achieved.

SELECT *?Activity ?Data_type ?value*
WHERE *{ ?Activity rdf:type essence:Explore_Possibilities ;?Data_type ?value .*
FILTER *(?Data_type = essence:boolean) }*

Results

	Activity	Data_type	value
1	essence:Achived	essence:boolean	"true"^^xsd:boolean
2	essence:Achived	essence:boolean	essence:Explore_Possibilities

After completion of Explore possibilities, a graph is generated to check the progress in graphical form which is shown in Fig 5.

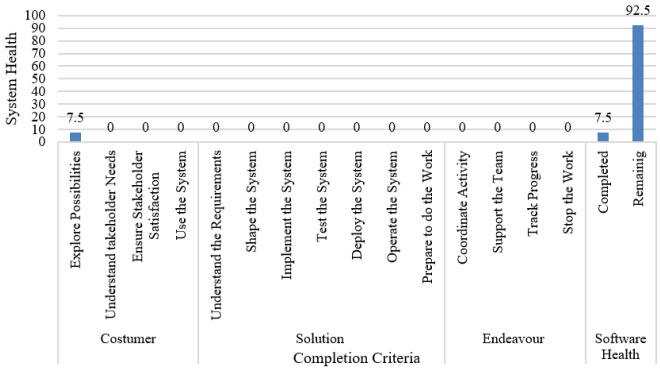


Fig. 5. Software Health after Explore Possibilities achieved

Fig. 5 shows that only one activity space that is Explore possibilities having 7.5% of totals software health is achieved. Furthermore, it shows that 92.5% of work is remaining to complete the project. Similarly, Fig. 6 shows the results when all activity spaces were achieved present in the customer area of concern and Fig. 7 shows the graph of the completed project.

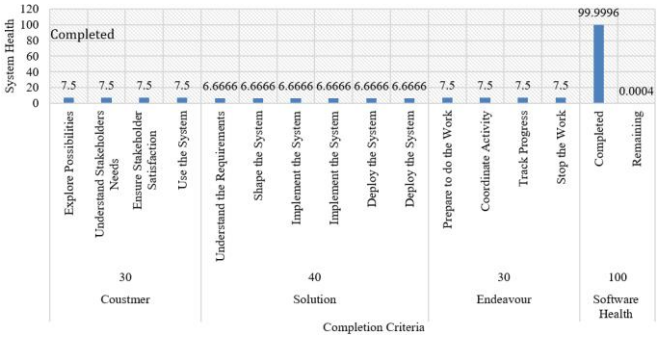


Fig. 6. Software Health after the Customer area of concern completed

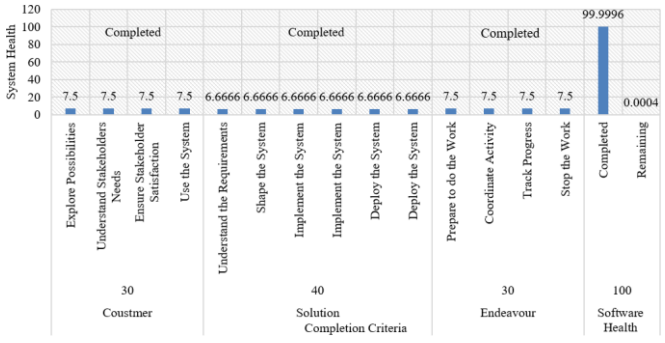


Fig. 7. Software Health after completion of the project

Results show that one can measure the health of software projects using ontology automatically. The results are compiled in seconds using ontology while literature shows that manually it is time-consuming to deal with a huge amount of data. Hence our hypothesis is proved.

6. Conclusions

Software development processes are huge in numbers, and it is difficult to check out the progress of software project manually. Therefore, we have investigated and find the need to automate software essence to measure the health of a software project. We have created an ontology at a low level using protégé tool and measure health using the SPARQL query. Results show that if software essences are automated, then one can successfully measure the health and progress of a software project quickly. In the future, the front-end should be more interactive for end user to improve the quality of software project.

7. References

[1] W. Dahhane, J. Berrich, T. Bouchentouf, and M. Rahmoun, “SEMAT Essence’s Kernel applied to O-MaSE”, in 2016 5th International Conference on Multimedia Computing and Systems (ICMCS), 2016, pp. 799–804.

[2] J. Pieper, O. Lueth, M. Goedicke, and P. Forbrig, “A case study of software engineering methods education supported by digital game-based learning: Applying the SEMAT Essence kernel in games and course projects”, in IEEE Global Engineering Education Conference (EDUCON), 2017, pp. 1689–1699.

[3] H. Oktaba, “Use of the essence and Kuali-Beh to structure software engineering courses”, in 2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT), 2016, pp. 47–52.

[4] Y. Zhao, J. Dong, and T. Peng, “Ontology classification for semantic-web-based software engineering”, IEEE Trans. Serv. Comput., vol. 2, no. 4, pp. 303–317, 2009.

[5] R. Bouzidi, A. De Nicola, F. Nader, and R. Chalal, “OntoGamif: A modular ontology for integrated gamification”, Procedia Computer Science, Preprint, pp. 1–35, year?.

[6] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, “What are ontologies, and why do we need them?,” IEEE Intell. Syst., vol. 14, no. 1, pp. 20–26, 1999.

[7] J. Brank, M. Grobelnik, and D. Mladenic, “A survey of ontology evaluation techniques”, in Proceedings of the conference on data mining & data warehouses (SiKDD), 2005, pp. 166–170.

[8] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann, “A theoretical framework for ontology evaluation and validation”, Proceedings of the 2nd Italian Semantic Web Workshop, University of Trento, Trento, Italy, 2005, vol. 166, p. 16.

[9] H.-J. Happel and S. Seedorf, “Applications of ontologies in software engineering”, in Proc. of Workshop on Sematic Web Enabled Software Engineering (SWESE) on the ISWC, 2006, pp. 5–9.

[10] J. E. Rogers, “Quality assurance of medical ontologies”, Methods Inf. Med., vol. 45, no. 03, pp. 267–274, 2006.

[11] L. Zeng, T. Zhu, and X. Ding, “Study on construction of university course ontology: content, method and process”, in International Conference on Computational Intelligence and Software Engineering, 2009, pp. 1–4.

[12] Y. Sure, S. Staab, and R. Studer, “Ontology engineering methodology”, in Handbook on Ontologies, Springer, 2009, pp. 135–152.

[13] N. Malviya, N. Mishra, and S. Sahu, “Developing university ontology using protégé owl tool: Process and reasoning”, Int. J. Sci. Eng. Res., vol. 2, no. 9, pp. 1–8, 2011.

[14] P. Wijewickrema and R. Gamage, “Automatic document classification using a domain ontology”, in National Conference on Library and Information Science (NACLIS), Colombo, Sri Lanka, 2012.

[15] G. Bilgin, I. Dikmen, and M. T. Birgonul, “Ontology evaluation: An example of delay analysis”, Procedia Eng., vol. 85, pp. 61–68, 2014.

[16] A. P. Vargas and R. Bloomfield, “Using Ontologies to Support Model-based Exploration of the Dependencies between Causes and Consequences of Hazards.”, in KEOD, 2015, pp. 316–327.

[17] M. Rani, R. Nayak, and O. P. Vyas, “An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage”, Knowledge-Based Syst., vol. 90, pp. 33–48, 2015.

[18] B. B. Duarte, A. L. de Castro Leal, R. de Almeida Falbo, G. Guizzardi, R. S. S. Guizzardi, and V. E. S. Souza, “Ontological foundations for software requirements with a focus on requirements at runtime”, Appl. Ontol., vol. 13, no. 2, pp. 73–105, 2018.

[19] A. Smirnov, A. Ponomarev, N. Shilov, A. Kashevnik, and N. Teslya, “Ontology-based human-computer cloud for decision support: architecture and applications in tourism”, Int. J. Embed. Real-Time Commun. Syst., vol. 9, no. 1, pp. 1–19, 2018.

[20] A. Alobaid, D. Garijo, M. Poveda-Villalón, I. Santana-Perez, A. Fernández-Izquierdo, and O. Corcho, “Automating ontology engineering support activities with OnToology”, J. Web Semant., vol. 57, p. 100472, 2019.

[21] A. A. Alsanad, A. Chikh, and A. Mirza, “A domain ontology for software requirements change management in global software development environment”, IEEE Access, vol. 7, pp. 49352–49361, 2019.

[22] V. JAIN and S. PRASAD, “Evaluation and Validation of ontology using Protégé Tool,” Int. J. Res. Eng. Technol., vol. 4, no. 4, pp. 2321–8843, 2016.

[23] I. Jacobson, P.-W. Ng, P. E. McMahon, I. Spence and S. Lidman, “The Essence of Software Engineering: Applying the SEMAT Kernel”, Addison-Wesley Professional, 2013.

[24] I. Exman, "A Bootstrap Theory: the SEMAT Kernel Itself as Runnable Software," Bootstrap Theory, 2014, pp. 1-8.