A METHODOLOGY FOR CREATING AN IEEE STANDARD 830-1998 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT *

Robert A. Elliott, Sr., Ph.D.
Department of Mathematics and Computer Science
Mississippi Gulf Coast Community College
Gulfport, MS, USA
(228) 897-3818
robert.elliott@mgccc.edu

Edward B. Allen, Ph.D.
Department of Computer Science and Engineering
Mississippi State University
Mississippi State, MS, USA
(662) 325-7449
edward.allen@computer.org

ABSTRACT

Software intensive systems are developed to provide solutions in some problem domain and software engineering principles are employed to develop and implement such systems. Software engineering principles should enhance development and production of software artifacts and yet the artifacts often lack in quality. Crucial in the development process are requirements engineering activities and methods for software documentation. This research focused on requirements engineering activities, software requirement documentation and a new approach incorporating ontology engineering principles. The resulting system utilized the benefits of intelligent reasoning to elicit, automatically verify, extract and document software requirements. The resulting Software Requirements Specification documents produced from within this environment conformed to Standard 830-1998 promulgated by the Institute of Electrical and Electronics Engineers (IEEE).

---

1 INTRODUCTION

This research was motivated by a need for automated assistance in elicitation, verification, and documentation of software requirements that conforms to Standard 830-1998 promulgated by the IEEE.[1]

Requirements engineering is defined as a process of obtaining and detailing the goals of a proposed software system. In the requirements engineering process there are several related manual activities that are error prone. The activity of requirements elicitation does not involve computer based execution of information gathering, computer based processing or computer based management. Some requirements engineers may not effectively conduct interviews to gather the user requirements or the requirements engineers may have a different interpretation of responses from stakeholders.

Once information is gathered, different problems occur as the requirements are manually distributed to the development teams in the software development process. Software development teams may be located in different parts of the organization's physical location or in a different part of the world. The distributed nature of development teams make it difficult to effectively communicate requirements engineering process activities and data since most of the elicitation process is not completely electronic.

2 REQUIREMENTS ENGINEERING

A problem associated with requirements engineering occurs in the requirements verification activity. This activity involves making sure that a user requirement is accounted for in the software system, and assuring that the requirement is feasible and is not in conflict with other user goals. The magnitude of software requirements makes manual verification of software requirements extremely difficult when done in traditional ways.

Formal methods of software verification are extremely difficult and requirements engineers tend to avoid formal methods unless the methods are embedded in software tools [2]. Our research methodology used an electronic method to verify certain characteristics of requirements and was faster and more efficient than a manual verification process that is normal in requirements engineering practice.

A product of requirements engineering is a software requirements specification (SRS) document that describes the proposed software system. The IEEE Standard 830-1998 [8] contains guidelines for the format and creation of a SRS document. The creation of a software requirements specification document is often an error prone manual process. The key elements of the document are the software requirements that have been elicited, analyzed, specified, verified, validated and documented. These requirements should be accurate and reusable as they are distributed within an organization. Ontology engineering methods are key components used in this methodology and aided in the requirements process.

[1] This paper is based on research by Elliott at Mississippi State University [4].

3 ONTOLOGY

According to Gruber [7], an ontology is a "specification of a conceptualization". This is one of the more popular definitions of an ontology in the scientific literature. According to Noy [11], an ontology is "an explicit description of a domain." This means that the concept or concepts are clearly defined. Calero et al. [1] state an ontology is "the attempt to formulate an exhaustive and rigorous conceptual schema within a given domain, a typically hierarchical data structure containing all the relevant entities and their relationships and rules (theorems, regulations) within that domain." This definition is birthed from the term in philosophy, where ontology means "the study of being or existence as well as the basic categories thereof" [3].

Ontology in computer science has its roots in philosophical ontology [9]. Cocchiarella[2] defines formal ontology as the formal properties and the classification of the entities of the world (physical objects, events, etc.), and of the categories that model the world (concept, property, etc.). This definition provides a theoretical foundation in ontology engineering.

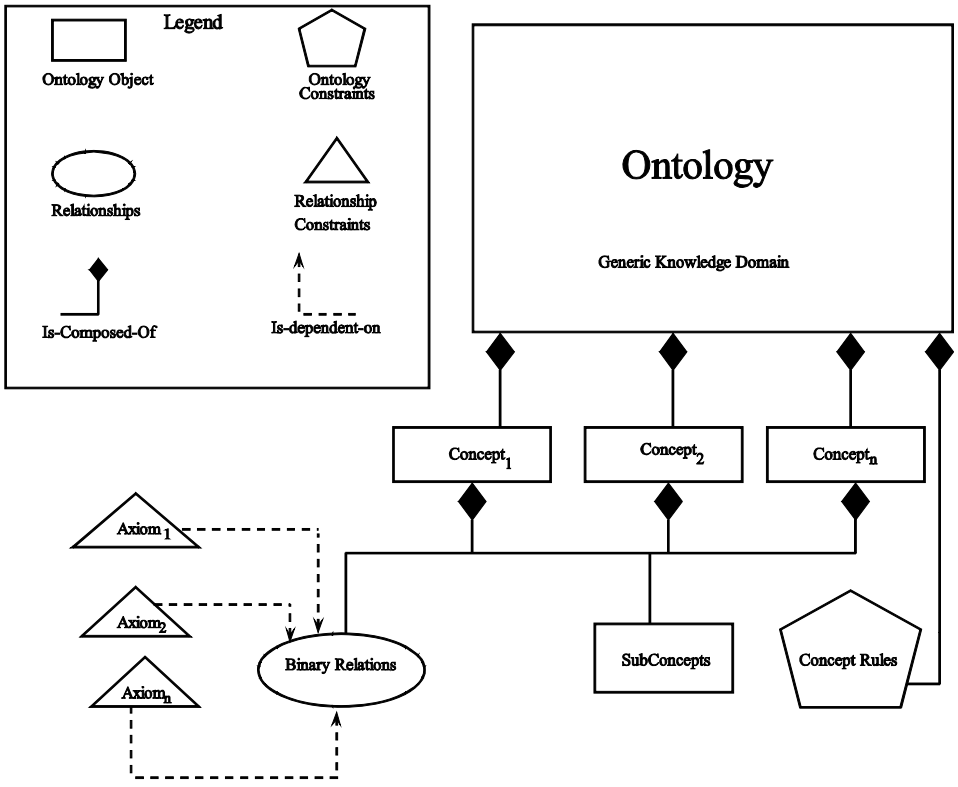

Figure 1

4 ONTOLOGY ENGINEERING

The ontology development process, the ontology life cycle, the methods and methodologies for building ontologies, tool suites and languages are activities that refer to ontology engineering [6]. An ontology includes a vocabulary of terms and

specifications of their meaning. An ontology formally defined describes the concepts in a machine-readable form and is defined in such a way that there is no ambiguity within the definitions [10]. Ontologies facilitate knowledge sharing and reuse and allow people, applications or intelligent agents to electronically share this knowledge.

There are rules for combining terms and relations that are defined within the ontology as described in Figure 1. The figure is an ontology model that contains hierarchical classes and subclasses that explicitly define a specific knowledge domain.

The instances define specific members in the ontology that contain real data.

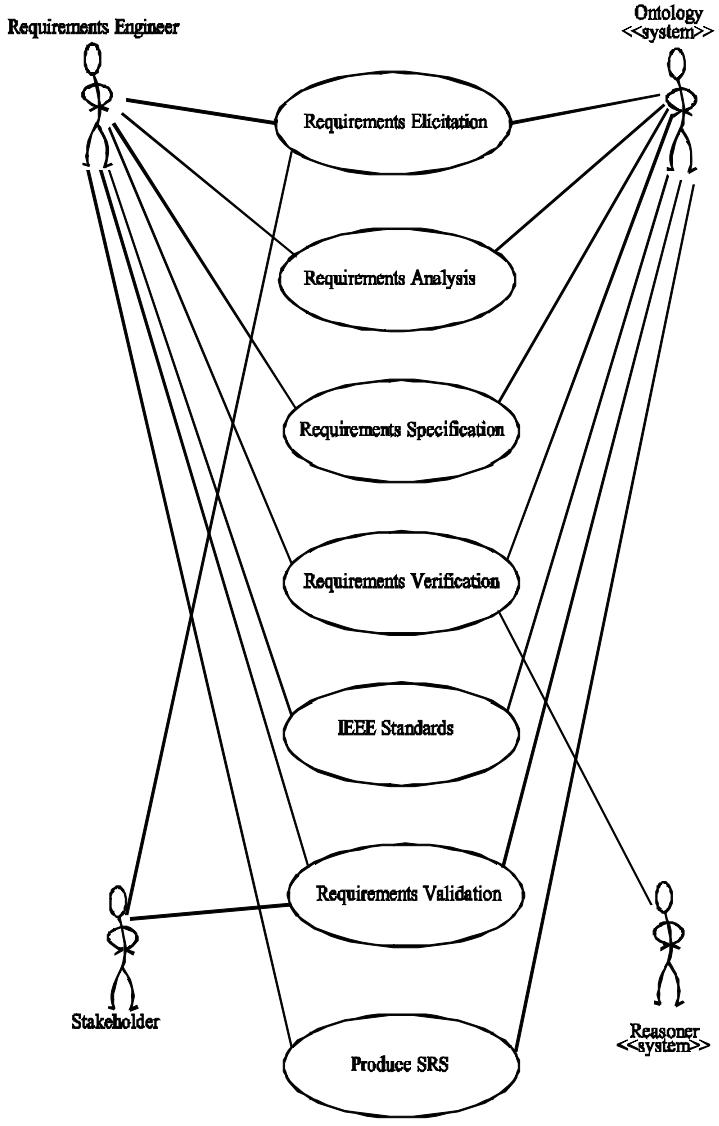

Figure 2

Concepts and subconcepts are depicted as ontology objects in a parent-child relationship. Binary relations show that objects within the ontology can have interactions with one another. The axioms define the rules that govern how objects relate to one another within the ontology system.

One of the most important concepts of an ontology is that contained knowledge is agreed upon by the community within the knowledge domain. The concepts are not just the ideas of an individual but of a group of domain experts.

5 METHODOLOGY

A primary goal of this research was to produce an IEEE Standard 830-1998 Software Requirements Specification document. This section describes the methodology in this research process and is depicted in the use cases in Figure 2. Each use case in the methodology is described below.[2]

- Requirements Elicitation Use Case- The requirements engineer will capture software requirements in a raw, informal format and store stakeholder requirements within an ontology.
- Requirements Analysis Use Case- The requirements engineer will (a) take the elicited requirements and employ methods to formally analyze the requirements, (b) update and categorize the requirements as user and system requirements and (c) update the ontology accordingly.
- Requirements Specification Use Case- The requirements engineer will formalize the software requirements and by adding more specific data to each requirement for clarity.
- Requirements Verification Use Case- The requirements engineer will make the necessary adjustments in the ontology to invoke the reasoner that will verify certain requirement characteristics. The relationships in the ontology also will be checked to make sure the relationships can be logically satisfied.
- IEEE Standards Use Case The requirements engineer will populate the IEEE Std 830-1998 data items in the ontology with domain specific information.
- Requirements Validation Use Case- The requirements engineer will run scripts to produce requirement reports this will be checked and approved by stakeholders. The requirements engineer will validate the requirements in the ontology by using the approved stakeholder documents.
- Produce SRS Use Case- The requirements engineer will use scripts to extract requirements from the ontology and produce the software requirements document.

6 CASE STUDY

A retrospective case study which involved software for a financial aid application developed at Mississippi State University was used to apply the methodology. This application provided software to automatically notify staff, students and users of information about financial aid transactions including loans at Mississippi State University (MSU).

[2] This methodology was a poster presentation at the 24th Annual Consortium for Computing Sciences in Colleges: South Central [5].

The requirements for this case study were captured on forms that were designed to notify all entities at the university involved in the financial aid loan process. The forms were designed by ITS staff at MSU and were scanned, and stored directly into the ontology in data elements. Each form was analyzed and contributed to requirement definition for this case study. The research methodology was focused on creating an entire software requirements document including populating all sections of the document.

Methodology

A baseline ontology was modified to include the data items that would be used to adhere to MSUs Financial Aid Office request for automatic email notification to staff and students. This new modified ontology was named WorkFlow and was formed with moderate additions and deletions of data items in the baseline ontology. The following describes a summary of the activities of the use cases in this case study.

- Requirements Elicitation Use Case Execution-Execution of the Requirements Elicitation Use Case in this case study involved processing ten workflow documents and eliciting and defining the requirements. Each requirement that was elicited and defined, was modeled in the ontology and contained fifteen data items in the ElicitedRequirement class object. Functioning in the role of requirements engineer, the researcher populated the data items with data from the workflow documents with real data, and implied data which provided a realistic viewpoint of the problem domain.

Work Flow Ontology New Requirements Table

Condition Number	Condition	Work Flow
1	ENTR-S = “R”	1
2	FSLSIR = “R”	2
3	LDCD-S = “R”	3
4	(LoanName=”Subsidized Stafford Loan Payment” OR “Unsubsidized Stafford Loan Payment”)	4
5	Graduate PLUS Loan Payment	5
6	PLUS Loan Payment	6,7
7	ENTR-G = “R”	8
8	EXIT-S = “R”	9
9	(Fund Code = “STFDS” or “STFDU”) AND Loan Status = “ACPT” AND APPL_STATUS = “Sent” AND (currentDate=submissionDate+14) AND Disbursement 0 “NO” AND LDNumber=APPLNzmber)	10

Table 1

- Requirements Analysis Use Case Execution-Execution of the Requirements Analysis Use Case revealed that several conditions elicited earlier in the process could have been combined in the ontology. Table 1 lists analysis results of the requirements and each condition is listed with the associated Workflow.
- Requirements Specification Use Case Execution- The following snippet XML-style text lists the contents of a specified requirement in theWork-Flow Ontology after execution of the Specified Requirements Use Case. The XML document element is SpecifiedRequirement and its children are composed of all the data items captured from the Elicited Requirements Use Case, the Analyzed Requirements Use Case and the Specified Requirements Use Case. After the execution of this use case, the requirements were suitable for inclusion into the software requirements specification document.

```
< snip >
<WorkFlow>
<!-- WorkFlow Ontology Project-->
<!-- Output Tree Listing - - - -Specified Requirements -->
   <SpecifiedRequirement ID="SpecifiedRequirement_1">
      <RequirementsAnalysisConceptualModelImage01>
            file:/dissertation/WorkFlow/WorkFlowForm1.jpg
      </RequirementsAnalysisConceptualModelImage01>
      <RequirementsAnalysisRequestStatus>
            Active
      </RequirementsAnalysisRequestStatus>
      <RequirementsAnalysisType>
            Normal
      </RequirementsAnalysisType>
      <RequirementsAnalysisSystemDocument>
            WorkFlow 5 Data Sheet
      </RequirementsAnalysisSystemDocument>
      <RequirementsAnalysisIDNumber>
            1
      </RequirementsAnalysisIDNumber>
      <RequirementsAnalysisClassification>
            Functional
      </RequirementsAnalysisClassification>
      <RequirementsAnalysisCost>
            3334.95
      </RequirementsAnalysisCost>
      <ElicitedRequirementIDNumber>
            1
      </ElicitedRequirementIDNumber>
   </SpecifiedRequirement ID="SpecifiedRequirement_1">
</WorkFlow>
< snip >
```

- Requirements Verification Use Case Execution- This Financial Aid WorkFlow project was straightforward and simple. Verification scripts were not necessary or applicable for this case study. Therefore, no requirement characteristics were verified within the ontology.
- IEEE Standards Use Case Execution-This use case involved populating the data items which stored IEEE Standard 830-1998 recommended data. The researchers were the primary parties responsible for this task. Pre-defined items defined in the Workflow ontology were populated and followed IEEE Standard 830-1998 guidelines. This use case inclusion in the requirements engineering process is a major difference from the traditional requirements engineering process. Its inclusion assures that IEEE recommended data is part of requirements definition.
- Requirements Validation Use Case Execution- The validated requirement information was used to extract requirement reports by ValidationStakeholderID which made validation of requirements more efficient.
- Produce SRS Use Case Execution- Execution of the Produce SRS Use Case populated the DocumentedRequirements class. This class contained all of the requirements that would be part of the software requirements specification document. The scripts documentedRequirements.xsl and srs.xsl were two scripts executed that produced the Workflow Software Requirements Document.

7 DISCUSSION

Our new methodology was applied successfully in this case study. The application of this methodology was straightforward and used the computer and software tools to manage software requirements. The ontology in this methodology was very easy to tailor to meet needs of stakeholders and requirements engineers. The IEEE Software Engineering Body of Knowledge (SWEBOK) standard data recommendations were also automatically implemented in the ontology design. The methodology was flexible and easy to follow and produced an IEEE Standard 830-1998 quality software requirements specification document.

This research utilized ontology engineering, IEEE Standard 830-1998 recommendations, and the SWEBOK guide to gain the benefits that each component presented. An ontology offered a standard vocabulary of domain terms and knowledge reuse. The IEEE Standard 830-1998 recommendation yielded consistent software requirements documentation.

REFERENCES

[1] Calero C., Ruiz F. and Piattini M., *Ontologies for Software Engineering and Software Technology*, Heidelberg, Germany, Springer, 2006.

[2] Cocchiarella N., "Formal Ontology," *Handbook of Metaphysics and Ontology*, H. Burkhardt and B. Smith, eds., Philosophia Verlag, Munich, Germany, 1991.

[3] CMSWiki, "Ontology," *CMSWiki Homepage: Ontology*, CMSWiki, 2006, http://www.cmswiki.com/tiki-index.php?page=Ontology (current 13 April 2013).

[4] Elliott R., "*Software Requirements Elicitation, Verification, and Documentation: An Ontology Based Approach*", doctoral dissertation, Mississippi State University, MS State, Ms, December 2012.

[5] Elliott, R. and Allen, E., "A Methodology for Creating An IEEE Standard 830-1998 Software Requirements Specification Document", *The 24th Annual Consortium for Computing Sciences in Colleges: South Central*, LSU-Shreveport, La, April 2013(poster).

[6] Georgiades M. and Andreou A., "Automatic Generation of a Software Requirements Specification (SRS) Document," *Proceedings: The 10th International Conference on Intelligent Systems Design and Applications (ISDA)*, Cairo, Egypt, IEEE, pp. 1095-1100 December 2010.

[7] Gruber T., "What Is An Ontology?" *The Protégé Ontology Editor and Knowledge Acquisition System*, Standford Press, 1992, http://www-ksl.stanford.edu/kst/what-is-an-ontology.html (current 13 April 2013).

[8] IEEE Computer Society, *Guide to Software Engineering Body of Knowledge (SWEBOK)*, Standard ISO/IEC TR 19759-2005, International Organization for Standardization, Geneva, Switzerland, February 2005.

[9] Jaspan C., Keeling M., Maccherone L., Zenarosa G.L., and Shaw M., "Software Mythbusters Explore Formal Methods," *IEEE Software*, vol. 26, no. 6, November 2009, pp. 60-63.

[10] Linkova Z., Nedbal R., and Rimna M., *Building Ontologies for GIS*, technical report 932, Institute of Computer Science Academy of Sciences of the Czech Republic, Czech Republic, March 2005, http://www3.cs.cas.cz/ics/reports/v938-05.pdf (current 13 April 2013).

[11] Noy N. and Tu S., "Developing Medical Informatics Ontologies with Protégé," *Proceedings: The 27th Annual Symposium American Medical Informatics Association*, Washington, DC, 2003, AMIA, Powerpoint Presentation.