A Comprehensive Ontology-Driven Software Development Architecture: An Approach to Developing Romantic Software Products

Nehemiah Mavetera, North West University, South Africa; E-mail: maveteran@uniwest.ac.za

ABSTRACT

Information systems have been criticized for their lack of flexibility and content richness. The problem has been traced back to the developmental stages of these systems. Current ISD approaches are mechanistic, that is, they lack a way of capturing the humanistic element that is inherent in a socio-technical environment such as information systems. To address this anomaly, ontologies can be introduced at the developmental stages to capture the romanticism inherent in these systems, to mediate during the design and development of the software products for these systems and to facilitate easy sharing of information among different information systems. This paper discusses an architecture that positions ontologies at the center of a software development case tool. The ontology drives and coordinates, the requirements analysis, design, and coding of software products that are domain specific. Case based reasoning tools, Bayesian Networks, WordNet, domain specific ontology, conceptual graphs and formal logic are the tools that are incorporated into this software development architecture.

Keywords: Ontology, software development, Architecture, Romantic software products.

1. INTRODUCTION

"There is a reason why computers have not yet become fervent natural language speakers. (It's not a matter of processing power and never will be): we simply are not programming them correctly." ***(El Baze, 2005)***

Current Information systems exhibit a mechanistic character that has curtailed their usability. These systems are very efficient at structuring data to enable and facilitate its interpretation. Mechanistic systems are based on the concept of explicit programming (Agentis International, n.d). Explicit programming produces software products that do not capture semantic and context rich data, a characteristic that is needed in all modern day systems.

The American National Standards Institute (ANSI) proposed a conceptual schema for knowledge encoding in the 1970s (Sowa, 2000). While the schema can coordinate efficiently between the applications, user interface and database of a system, it relied on syntactic coding, and is not evolvable. Other software development paradigms such as the structured approach (Pressman, 2005), object oriented approach (Pressman, 2005, Dennis et al, 2002); software product lines approach (Carnegie Mellon Software Institute, n.d.) software kernels approach (Information Technology University, Denmark, n.d) have been introduced to try and improve the adaptability, evolvability, reusability of software products as well as increase the semantic richness of the resultant information systems. As their eighth basic principle of system development methodologies, Whitten et al (2004), tell systems developers to design their system for growth and change. Pressman(2004), in discussing the nine software myth, raises issues such as evolvability, quality measurement, throughput levels, reusability of software products, management of scope creep during development, and software products documentation as some of the most misunderstood aspects that need to handled carefully during a software development process.

This paper discusses how the ontological approach can incorporate the aspects into information system development. The rest of the paper is as follows. **Section two** discusses the software development problem, **section three** looks at the transition from mechanistic to romantic software products, **section four** briefly discusses the role of ontology in information systems and finally **section five** discusses an ontology driven software development architecture. The conclusion closes the discussion but also summarizes the way forward in the ontology research process in ISD.

2. THE SOFTWARE DEVELOPMENT PROBLEM

The problem in the resultant software products and whose characteristics subsequently emerge in the final information system have been tracked down to the developmental stages of the software product. The software development process is a part of a system development process that includes a set of activities, methods, best practices, deliverables and automated tools that developers use to develop and maintain information systems. Basden (2001), in his article "Christianity Philosophy and Information Systems" decried the continual lack of return of investment from information systems investments. He noted that there is something deeply wrong in the way the "artifact" is developed. This artifact is the software product. He further searches for the problem and the solution. "What is wrong? "and "what do we do about it?" His solution set coupled with other researchers suggestions are herein included as a way to improving the usability of the software product.

2.1 Software Development Issues

Issues considered during the software development process play a vital role in shaping and determining the qualities of the software product. Basden (2001) noted four areas of concern that can be addressed to improve the quality of the product. The areas look at fashioning of technical artefacts for use, development of technology from which we fashion an artefact, the use of the artefact and users' and developers' overall perspective on the use of technology (herein we add the social context of information systems and their situated ness). In addition, other issues require:

- Developers to focus on designing reusable components
- Developers to focus more on the innovative elements of a software product design
- That the innovative elements of a software product represent the domain related additions that make the difference between domain packages.

Managing these issues leads to a gradual change from mechanistic to romantic systems.

3. DEVELOPING ROMANTIC SOFTWARE PRODUCTS

To bridge between the mechanistic development methods and the required romantic methods, we are going to use the ontology artifact. Romantic systems possess a certain degree of humanistic behavior. They are open, non-deterministic and

do not subscribe to mechanistic ideas of representation, formalization, program, order, reason, stability and control like machines. These systems borrow their definition from romanticism (Basden, 2001; John, n.d.; Gregor, n.d; Loflin, n.d) which imitates belief systems that depend on "irrationalism and feelings".

Ontology has enjoyed many definitions in the literature. The section below gives a brief of the common accepted meanings to ontology. This is the definition widely used in information systems development.

3.1 Ontology of a System

Ontology can be viewed as an engineering artifact. In this part, ontology consists of a specific vocabulary used to describe a certain reality (Guarino, 1998). The vocabulary used is accompanied by a set of explicit assumptions, which give people the intended meaning of such a vocabulary.

Studer et al (1998) add that ontology is an explicit formal specification of a shared conceptualization. Formalization looks at machine readability (syntactics) of the ontology. Explicit specification incorporates the clear identification of concepts, properties, relations, functions, constraints and axioms (semantics) within a universe of discourse. If a thing is clear to a subject, that thing should make sense to the said subject(Mavatera, 2004b). The addition of the phrase 'shared conceptualization' denotes ontologies as abstract models of phenomena in the world with implicit knowledge in them. Furthermore, there is some sense of mutual understanding of the concept among people in the same contextual environment (pragmatics).

Neches *et al* (1991) defined ontology as 'the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary'. Swartout *et al* (1997) describe ontologies as a hierarchically structured set of terms for describing a domain. Gruber (1993) defines ontology as 'a specification of a representational vocabulary for a shared domain of discourse…' He goes on to say that ontology is an 'explicit specification of a conceptualization'. In a more literal way, Ontology consists of a set of concepts and their relationships, forming a conceptual structure that underlies the interpretation of any system model. In short, ontology of a system can be taken as a set of representational terms in the universe of discourse. The ontology is used for 'sharing and reuse of formally represented knowledge'.

4. ONTOLOGY IN INFORMATION SYSTEMS

The purpose of using ontology is to develop software products that capture semantics and social context of information systems through the development of databases of domain ontologies and application packages that capture semantics, context and the situated ness of organizational information systems. It is:

'*…about awareness, connection and meaning, impact versus activity and knowledge versus data'. in-PharmaTechnologist.com (2005)*

The ontology replaces the conceptual schema at the center of an integrated information system as previously stipulated by ANSI (see section 1 above). In this research, we take advantage of ontology characteristics such as easy to use, different formal expressiveness with reasoning support, integrated form generation to acquire instances, ability to build test cases and use the cases to check consistency, ability to be manipulated and reason at run time, ability to drive control logic of a program, ability to be tuned so as to automate the software testing process as well as ability to allow user involvement at any stage of the development to position it at the center of the development of romantic software products.

In short putting ontology at the center of the software development tool allows the resultant software products to be adaptable, evolvable and be context aware.

5. ONTOLOGY DRIVEN SOFTWARE DEVELOPMENT ARCHITECTURE

Figure 1 below shows the architecture of the software development tool that we refer as the OntoSoft case tool. The OntoSoft tool has three major components, the knowledge base repository, the designer engine and the reasoner.

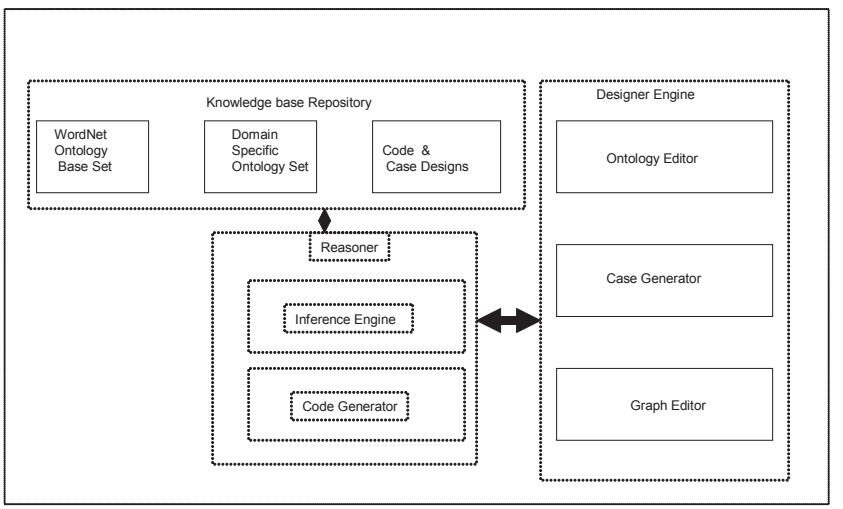
5.1 The Knowledge Base Repository

Unlike the Rebuilder case tool discussed in Gomez,(2004), the OntoSoft knowledge base repository consist of three parts, the WordNet ontology base, the domain specific ontology base(not found in Rebuilder or any other case tool developed so far) and a code and case base set.

The WordNet ontology base is taken and maintained "as is" . WordNet is a type of terminological ontology(Sowa, n.d). It is a lexicon and consists of information about " syntax, spelling, pronunciation and usage of words" In short, it is a natural language knowledge base. It is not updated so as to maintain linguistic consistency in terms of international grammar and general meanings to terms.

The domain specific ontology set is specific to an application software domain and is allowed to change according to the different conceptualizations and ontological commitments(Guarino, 1998) to a certain domain. This is the knowledge base that users' and developers can fine tune to suit their application domains. Finally, the case and code designs base store new and old designs that are relevant to a specific application domain.

Figure 1. Ontology software development case tool architecture



5.2 The Designer Engine

It consists of an ontology editor, case generator and graph editor. The three are used to develop domain specific ontology on the run, cases that capture the differing designs in the application domain and graphs that are used to map related concepts in a domain through their respective conceptual relations.

The designer engine uses the same principles as Rebuilder in terms of case indexing with the only difference being the graph editor. Rebuilder uses UML as case editor. UML as a case editor does not link the cases to the meaning of the cases which are stored in the domain ontology. It is purely syntactic. OntoSoft uses conceptual graphs that are a graphic notation for logic based on existential graphs. The conceptual graphs are augmented with features from linguistics and the semantic networks of artificial intelligence. Conceptual graphs can be used to map to and from natural languages. As a presentation language, they are used for displaying logic in a more human readable form. The conceptual graphs will be linked to the domain specific ontology to beef the knowledge content of the cases and designs.

5.3 The Reasoner

The reasoner consists of the inference engine and a code generator. The inference engine is like the communication engine between the designer engine and the knowledge base repository. It accepts user queries, retrieves old cases, and links new cases to WordNet and domain specific ontologies, links code to the cases and conceptual graphs. In fact, it is the brains behind the OntoSoft case tool.

The code generator automatically develops code specific to a retrieved or adapted case. The reasoner uses Bayesian Networks (BN) techniques to index cases and case based reasoning (CBR) principles which are well covered in Gomez(2004). CBR is based on the reuse of experience. It captures every reasoning instance as an episode that is registered and stored in a case. As each case captures a specific situation and is context related, then the syntactic, semantic and pragmatic aspect of the situation is also captured(Mavetera, 200b). The reader is directed to this article for further explanation on case based reasoning.

6. CONCLUSIONS

This paper discussed problems that currently bedevil our information systems to a proposed development architecture that can solve most of these problems. Of importance is the focus of the paper on the "artifact" during software products development. Unless the artifact is made adaptive, evolvable during the design stage, software developers must not expect the resultant information system to be adaptive and evolvable. The paper positions ontology at the center of romantic software products development process. These products will be reusable, process able, and in addition, they can be adapted and evolved to come up with entirely new software products. The software myths (Pressman, 2005), Software Product Lines(Carnegie Mellon Software Institute, n.d), software kernels (Dittrich & Sestoft, 2005) are all issues that can be solved by positioning ontology at the center of the software development process.

The next stage of the research is to engage industry partners, initially to investigate the software development practices that are in existence, the approaches, the methods, techniques, and the tools they use to come up with a product that give information systems their social situated ness. These findings are a very good tool which can be used to validate and motivate the industry use of the OntoSoft case tool framework. The OntoSoft case tool will also be tested using a prototype.

7. REFERENCES

AGENTIS. n.d. *Simplifying the complexity of Application Development. Developing and deploying J2EE solutions with Agentis Adaptive Enterprise TM Solution Suite,* Agentis International, Inc.

BASDEN, A. 2001. Christian Philosophy and Information Systems. Presented to institute for Christian studies, Toronto. [Online]. Available: http://www.isi.salford.ac.uk/dooy/papers/cpis.html. [Cited on 31 August 2006].

CARNEGIE MELON SOFTWARE ENGINEERING INSTITUTE, nd. Software Product Lines. [Online]. Available: http://www.sei.cmu.edu/productlines/ , [Cited 14 August 2006].

DITTRICH, Y & SESTOFT, P. 2005. Designing Evolvable Software products: Coordinating the evolution of different layers in kernel Based Software Products. [Online]. Available: http://www.itu.dk/research/sdg/doku.php . Cited October 2006].

EL BAZE, N. 2005. Cracking Software Development Complexity: In order to reap the benefits of progress in processor technology, we must fundamentally rethink how to write software, *Line 56-The Business Executive daily*. [Online], Available: http://www.line56.com/articles/, [Cited 30 March 2005].

GOMEZ, P., 2004. Software Design Retrival using Bayesian networks and WordNet

GREGOR, S. n.d. The Struggle torwards an understanding of theory in information systems.School of Business and Information Management, The Australian National University. [Online]. Available:http://epress.anu.edu.au/info_systems/ part-ch01.pdf. [Cited 31 August 2006].

GRUBER, T. A.1993. Transition Approachto portable ontology specifications. *Knowledge Acquisition*. Vol.5.1993. p.199-220.

GUARINO, N., 1998. Formal Ontology and Information Systems. *In Proceedings of FOIS'98,* Trento, Italy, 6-8 June, 1998. Amsterdam, IOS Press, p.3-15.

IN-PHARMA TECHNOLOGIST.COM, 2005, Improved data dealing drives drug discovery. [Online], Available: http://www.in-pharmatechnologist.com/news/, [Cited, 30 March, 2005].

JOHN, R.R. 2003. When Information Come of Age: Technologies of Knowledge in the Age of Reason and Revolution, 1700-1850. In William and Mary Quarterly, Vol. LX, No. 2, Reviews of Books.

LOFLIN, L. n.d. Romanticism Notes, [Online]. Available:http://www.sullivan-county.com/nf0/nov_2000/romanticism.htm. [Cited on 4 September 2006].

MAVETERA, N. 2004a. The Use of Ontologies in Designing Agents for E-Markets: From a Mechanist to a Romantic Approach: *In Proceedings of the International Business Information Management Conference (IBIM '04),* July, 2004, Amman, ISBN: 0-9753393-1-1

MAVETERA, N. 2004b. The Philosophy of Ontologies: A new Information Systems Development Paradigm. *In proceedings of the International Science and Technology Conference (ISTC'04),* Vanderbailpark, SA

NECHES, R.; FIKES, R; FININ, T; GRUBER, T; PATIL, R; SENATOR, T; SWARTOUT, W.R.1991. Enabling Technology for Knowledge sharing. *AI Magazine*. Winter 1991. p.36-56.

PRESSMAN,R.S 2005, Software Engineering : A Practictitioner's Approach, 6th Ed, MacGraw–Hill, International Edition.

SOWA, F. J. 2000. Ontology, Metadata and Semiotics. Conceptual structures: Logical, Linguistic and computational issues., Springer-Verlag, Berlin, 2000, p 55-81, Ganter & Mineau, Eds. (*Lecture notes in AI #1867)*

STUDER, R., BENJAMINS, R. & FENSEL, D. 1998. Knowledge Engineering: Principles and Methods. *Data and Knowledge Engineering*. Vol.25. p.161-197

SWARTOUT, B., PATIL, R., KNIGHT, K. & RUSS, T. 1997. Toward Distributed Use of Large Scale Ontologies. *Ontological Engineering.* AAAI-97, Spring Symposium Series. 1997. p138-148.

TURBAN, E., MCLEAN, E. & WETHERBE, J. 2004. *Information Technology for Management: Transforming organisation in the Digital Economy, 4th ed*. John Wiley & Sons, Inc.

WHITTEN, J.L & BENTLEY, L.D. 2004. Systems Analysis and Design methods. Irwin/McGraw- Hill