New innovation on software implementation methodology for the 21st century, --What Software science can bring to Natural Language Processing--

Issam A. Hamid,
Iwate Prefectural University, Iwate, Japan
issam@soft.iwate-pu.ac.jp
And
Fumio Negoro,
The Institute of Computer Based Software Methodology and Technology, Tokyo, Japan
f-negoro@lyee.co.jp

Abstract

In this paper we are discussing a motives of a big research project. As a new challenge for having software methodology suitable for the new century, needed for its flexibility, extensibility, easy to implement and rich to reflect user intention in all means. These objectives were not easy in the conventional software development due to lack in a suitable theory that can reflect the user intention into a program code. In this paper we are going to give the motivation behind a big academic joint research project to reach those objective. Also, we are going to presents a new software paradigm by involving on a survey on new software development suitable for such purposes, especially for natural language processing.

1. Introduction

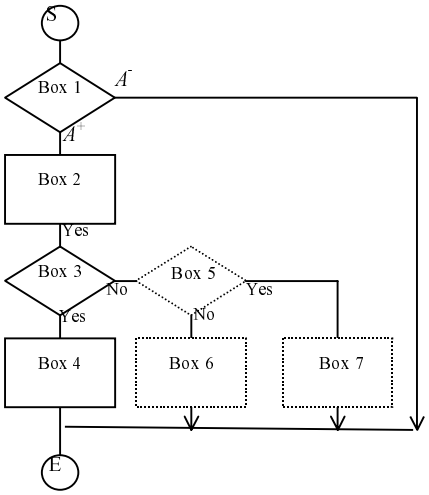
Software science in practice deals with the development of large and often complex information processing systems. This includes the techniques for the description of requirements and systems in their development phases. Its main goals are a good quality of the engineering process and its results, as well as high productivity. Like any other engineering discipline, software engineering needs strong foundational principles that reduce the ambiguity in requirement realization and other issues raised through the design process. Engineering discipline use a bunch of foundations as a basis for a deeper understanding, a body of rules, procedures, and engineering process. Hence, requirement must concern itself with an understanding of beliefs of stakeholders *(epistemology),* the question of what is observable in the world *(phenomenology),* and the question of what can be agreed on as objectively true *(ontology).* Such issues become important whenever one wishes to talk about validating requirements, especially where stakeholders may have divergent goals.

The development of large software systems can hardly be carried out with modeling and description techniques that are based on programming language only. As a consequence, so-called modeling languages have become popular in software and system engineering. Typical examples are design language like SDL (for formal representation of requirement specification), state chart (for formal description and verification of these requirement), as well as the object oriented modeling methods, like OMT, objectory, and others. Recently, the so-called unified modeling language UML, has been released. All these approaches have the goal to provide modeling methods in terms of diagrammatic description techniques for the analysis and design phase. In these phases we describe the aspects of the application domain the requirements as well as the architecture of a software system.

All these approaches suffer however; under the fact that they are too syntax oriented and lack a proper and simple semantic foundation. A precise description and common understanding of the semantics as well as of development relations between the various diagrams for the description of software systems provided by those methods is missing.

On example of these is the natural language processing, NLP. The requirement expressed in NL can be incomplete and lack a lot of precision hidden in the words expression. This leads to have a difficulty to extract rules to develop a representation semantics from those expression, and in turn lead to difficulty to implement those requirement into a useful implementation, (i.e., software program). Individual representation may be ambiguous, allowing multiple and perhaps unintended interpretations. The mapping between different bunches of requirements, representing an intent, in the hierarchy, is partially specified, if at all, making it impossible to accurately trace the linkage of implementation decisions. The analysis is mostly limited to syntactic checks. It is not possible to check the semantic properties of requirement specification, such as safe, and fairness of its connections, or to check the relative correctness of relations between correlated requirement given in a specific hierarchal structure.

Here we would like to represent a framework of a refinement for requirement for correctness, in terms of requirement representation in natural language, and see that requirement resulted as specification. So they are complete and correct in their semantics. This paper is based on presentation in the same conference under title “Intent Operationalisation for Source Code Generation” by Dr. Fumio Negoro. Additionally in this paper we try to analyze Lyee framework as a general solution for the



	Instructions in the box
Box 1	Which does a subset of the elements (Logical Atoms) belong to, A^+ or A^-?
Box 2	Objectify one of the elements of the subset in A^- corresponding to the subset in A^+ so as to make that element a substance of the existence.
Box 3	Has the objectification been done?
Box 4	Objectify the rest of the elements of the subset in A^- so as to make them attributes of the existence.
Box 5	(Explanations on dotted boxes are omitted.)
Box 6	
Box 7	

Figure 1. Structure of the principle of objectification

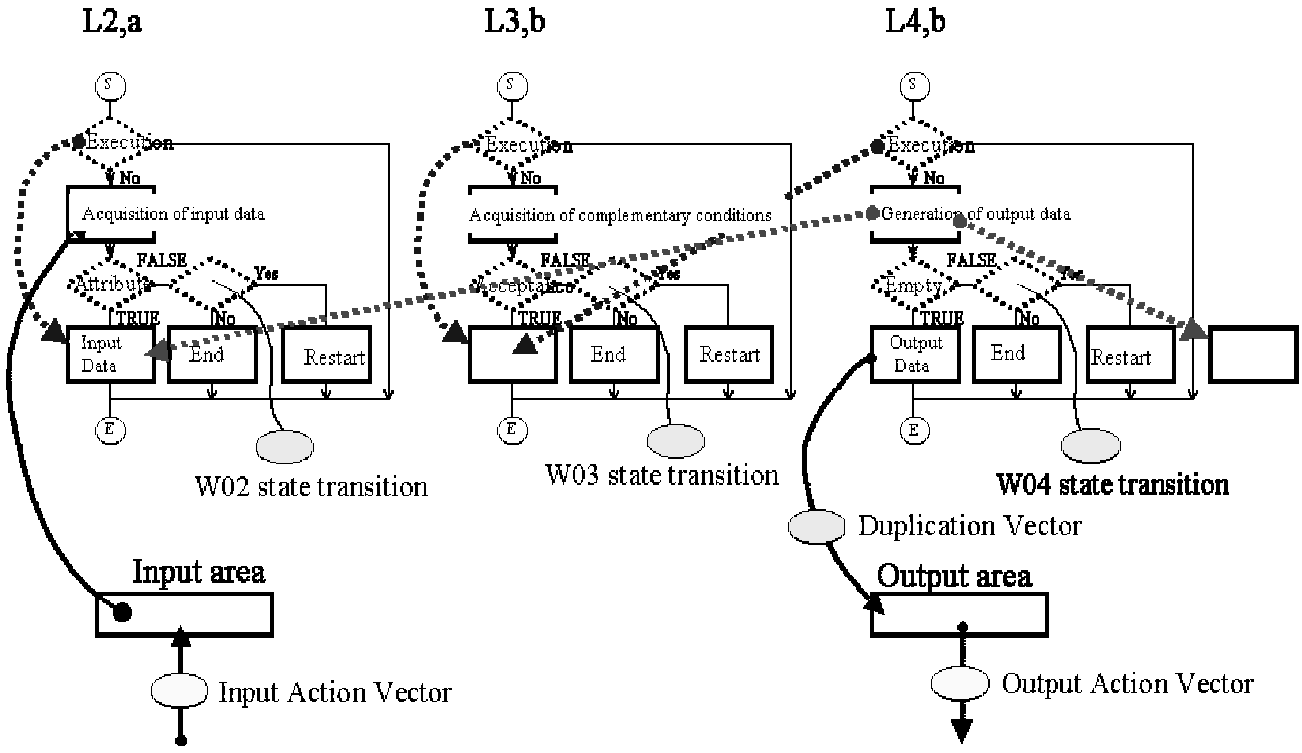


Fig. 2 Verbing or action-ing the words

objective of big project aim to approach new generation of software science.

2. User requirement and it natural language representation

An *intent* (*i.e.,* requirement expressed in natural language) can be replaced by a set of propositions that can be defined by Predicate Structure. This structure works are realization representation in abstract sense of user requirement (user specified intent). The users specified their intent using natural language representation, and consequently, through Lyee universal model, so all the words from such presentation are extracted, and in turn each words is been transferred into predicate structure. The predicate structure represents the realization of these words a common unified presentation for all words used in NL context reflecting that intent, independent of their meaning or actual semantics in relation to the intent or context they have been used for or in.

The predicate structure is presenting universal presentation of the state of a word as a self-entity, in a space without relation or relative meaning. So in other words, a word represented by the predicate structure at the initial stage is just reflecting the static state of that word before link it into a sentence to become part of the context in the intent, or the requirement.

By the nature of the predicate structure, and by the nature of the Lyee framework, the words bring to itself some change from its sabbatical nature (i.e., static existence being not linked to other word) to dynamic existence; being linked to other words to reflect a behavior representation given through the intent context (i.e., requirements). The linkage with other words, come through the nature of the predicate structure when the word have some certain status. By status we mean, that each word in a context is been represented by predicate structure which in turn transformed into a piece of program code running in a computer memory system. Each word has some status to be either in true state or false state according to the nature definition of the predicate structure in Fig. 1. The Lyee framework is to establish all the memory status to become true. So the iteration till the memory status becomes true, is as if we are verbing the word, (objectification or bringing action on words). We call this as objectification, which represents the action of bringing meaning to a set of words. Through such realization the semantic representation of the intention definition is been realized.
The realization of words as the way given in Fig.2 brings the vision of requirement realization in synchronous nature. This means that the words relationship as it is presented in the context are realized by the synchronization nature of the Lyee theory, through which the relationships between words is established. This means that the meaning between words, reflected as syntactical and semantical presentation have been reflected in total through the synchronization behavior of the three components of similar predicate structure, representing what we called the scenario function(Fig.2). This is in contrast to the conventional realization of requirement, which usually be done in asynchronous way. Because, the entire requirement represented in NL are been treated into independent parts for realization. In Object oriented systems, the requirement is been classified into classes and the customization of requirement on these classes brings some sort of asynchronous realization or engineering on these requirements. In general conventional systems the structural and behavioral aspects of the system are separated for sake of reducing the complexity, but this lead to combining them later, and so add some sort of engineering realization for such integration, and in turn this add extra engineering requirement aspect, may leads the software system to have extra cost. Some system software for realizing those requirements needs special language called as configuration languages, supported by configuration management systems [Isazadeh A., Requirements Eng (1999), Springer-Verlag, 4:19-37].
Also, conventional realization of requirement or user intention is incomplete, because some portion of the requirements, and one or a few possible interactions were picked, but neglect other possible events (like failure conditions) that could arise during the same interactions.
Also, the asynchronous nature of engineering requirement is unstructured, and in most cases informal approaches to handle them fail to show how to combine these requirements to give a complete requirement specification. Unfortunately, this is left to the analyst who tries to make the integration, in which the original intent may not be readily visible.
We think the realization of the NL semantics can be done by using the Lyee software structure, consisting of the three components W02→W04→W03, with iterations. The above sequence can be interpreted as defining the word's input, done by a vector specified as W02. Then the corresponding words' output or input vector with the logical or calculation formula to calculate that output, is done by the W04 vector. The W03 is doing the calculation or what is to be called as reasoning to make the objectification done by the user as correct semantics for their intention in regard to those defined words. The W03 as a predicate judges to keep the iteration going in or going out of that cycle to another word cycle, this is the same for a set of words which the iteration cycle judge the realization of the meaning of the set of words as complete by the W03 final iteration step.
We think that the realization of Lyee framework for NL processing is rather a new motivation for unambiguous realization for the semantics and understanding or NL expression, if these expressions have been transformed into Lyee structure.
Lyee framework does not need modeling techniques to verify the resulted program structure, represented as a set of vectors. This is because the iterations done by the scenario function, bring the validation and verification onto the requirement. This is better than conventional way which needs modeling of requirement and

additionally, because the choice of modeling technique affects the set of phenomena that can be modeled, and may even restrict what a requirements engineer is capable of observing. So avoiding this brings some sort of capability to validate the requirement without the needs of all those modeling techniques, which need additional engineering requirement, may differ than the user has given.

Groundwork for the process model realization of requirements in Lyee is unique and universal by its nature, for all type of applications. We think it is realizable through universal Turing machine model. This is due the clean incremental nature of the predicate structure used in Lyee framework. We think there is no specific requirement identification is needed, but for conventional requirement identification we need a process model, and the selection of methods and techniques for the various requirement activities. The term *process* here is used to denote an instance of a process model, which is an abstract description of how to conduct a collection of activities, describing the behavior of one or more agents and their management of resources through collection of words. Those words are represented in such model by three pallets named as W04 W02 and W03. A *method* provides a prescription represented through W04 for how to perform a collection of activities on a set of words, focusing on how a related set of techniques can be integrated, and providing guidance on their use through the integration of the scenario function in hand. Level goals (such as business goals) are refined into lower-level goals (such as technical goals that are eventually operationalised in a system). [Requirements Engineering: A Roadmap, *Bashar Nuseibeh & Steve Easterbrook,* ICS,2000, pp.35-46]

In Lyee this is recalled as engineering, and as well as the scenario function is doing all the above, we think the term engineering is avoided here. As we see that engineering is the involvement of engineer to act on the user requirement for processing them and analyzing them for accuracy and realization. We believe this may shift the actual requirement from the resulted ones.

3. Conclusion

This is paper shows an overview on using Lyee framework for requirement realization in comparison to conventional methods. We conclude that this lead us to new direction for new generation of NL requirement realization suitable for new generation of software science. This is the foundation of our main international research project for new software science suitable for NL processing.