DESIGNING STORY CARD IN EXTREME PROGRAMMING USING MACHINE LEARNING TECHNIQUE

Azman Yasin[1], Shamsan Gaber[2], Mazni Omar[3], Haslina Mohd[4], Fauziah Baharom[5], Marina Md Din[6]

[1,2,3,4,5]Universiti Utara Malaysia, {yazman|mazni|haslina|fauziah}@uum.edu.my;
[6]Universiti Tenaga Nasional, marina@uniten.edu.my
[2]shamsagel@gmail.com

ABSTRACT:

Story card is one of the software development artifacts that can be used to gather requirements in extreme programming (XP). It can assists developers to translate and develop the system based on activities and rules stated in the story card. However, conventional XP story card framework or template is not well defined and only supports requirements in two or three sentences. It also does not states any information rather than system functionality. This may lead to conflicts, missing, and ambiguous requirements. In order to overcome this problem, Machine Learning is one of the techniques that can be used to extract the content from the list of requirements and produce the story cards based on the priority and rules of requirements. Thus, this study aims to to propose a new technique of designing story cards based on user requirements. The finding from the study is a conceptual model of designing story cards using machine learning technique. Future research will investigate how the technique adapt with the iterative changes of the requirements.

Keywords: agile, story card, parsing, text classification, text simplification, Machine Learning, Extreme Programming.

I. INTRODUCTION

Agile development methodology is one of the software development methodology approaches. This methodology implies on iterative and incremental development. It is a lightweight methodology utilizing specific techniques to deliver a product on time, under budget and meet customer satisfaction (Kavitha & Sunitha, 2011). Agile development methodology focus on delivering the system faster and at the same time able to cope with the customer changing needs and expectation (Tuck, France and Rumpe, 2003). The principles of agile methodologies are individual and interactions over process and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan (Kavitha & Sunitha, 2011). Agile development is not just as simple by gathering all specification in the beginning phase of development, the developer need to justify only possible and suitable requirement for implementation in the system from time to time until in the end customer will satisfied with the system. There are several popular examples of agile methodologies one of them is extreme programming (XP) (Beck, 2000).

II. EXTREME PROGRAMMING REVIEW

In XP, the requirement gathering technique is based on user stories where the customer will narrate the requirement for system development, thus incorporate the customer need directly in the system functionality. Since the requirement specification is in the iterative process, it is easier for the developer to react with the changes and adding new requirement to enhance the quality of the system according to what the customer want (Leffingwell, 2009).

The user story is seldom fully gaining all information to develop the functionality. This denotes that a framework is necessary to elaborate further on the requirement specifications changes by the customer. This is called elaboration phase or a stage at which the requirement/Agile Story gets further detail. At times, people may misinterpret that user stories are similar to Use Case. Use Case need not be comprehensive, but should have specification of conditions defining interaction with the product. In an

Agile environment a user Story is a written tool used in the requirements gathering process to describe the specification of a software feature and portrays a system's behavior in understandable way to both developers and users. Typically, a user story is brief as it consists of only one or two sentences (Naumovich, 2007). The user stories focus on user-centered rather than a functional breakdown structure. They provide a lightweight and effective approach to managing requirements for a system (Kavitha & Sunitha, 2011).

A user story is an informal statement of the requirement instead of a large requirements document. The story communicates to the design team WHAT is needed and does not specifically address anything about HOW to implement what is needed because the HOW part is strictly in the domain of the IT development team. The real intention of a user story is to provide the team with the ability respond quickly to user wants and needs (Leffingwell, 2009) It creates less overhead in the face of rapidly changing real world requirements or discovery of new requirements based upon the work in progress. It is not specifically a description of a feature in a program, but the underlying real world problem that the software component is designed to solve for the user business.

In addition, it also contains relevant inspirations to be explored with greater depth for later development. During the development process, several conversations between the customers and the development team will be conducted. These conversations are used to picture additional information for requirement documentation. This documentation will be attached to the card corresponding to suitable acceptance test criteria. Focusing on verbal communication is important to performed automated tests to communicate requirements. For acknowledgment, the customer may schedule it in any iteration they wish.

The development plan for XP begins when customer writes and elaborates user stories on story cards. Story cards are one of the important aspects in XP. It depicts the functionality of proposed system or software will be helpful to client. Three aspects included in user stories are (Cohn, 2000, 2003):

1) A written description of the story used for planning and as a reminder
2) Conversation about the story that serves to flush out the details of the story
3) Tests that convey and document details and that can be used to determine when a story is complete

These story cards are assessed by the developer to build a time box of iteration for development process based on the customer priory's requirement. Developers translate and develop the system according the story cards. The activities involved in this phase programming concurrently with test driven development. In the end of the phase, the customer will assess the functionality through acceptance test.

Story cards are written by the customer in XP to lucid their business needs since they know their business need very well compared to developer. However, normally customers only have a general picture of the requirement. Thus, conventional XP story card framework or template is not well defined and only supports requirements in two to three sentences. It does not state any information rather than system functionality. This will lead to conflicts, missing, and ambiguous requirements. According to Cohn story cards must be testable, definable, and valuable to the customer, small and independent to overcome hardly make a decision or predict wrong priority of the requirements or story cards (Cohn, 2000, 2003).

III. PROBLEM BACKGROUND

Generally, software developers intend to extract story cards from user requirements that have been collected from the user. The requirements will be extract and categorize manually. Therefore, it is a need to propose a new technique to customize the requirements so that it able to produce story cards from user requirements instantly.

Therefore, the main objective of this research is to propose a new technique of designing story cards based on user requirements. The proposed technique plays an important role in producing and categorizing story cards from the user requirements.

IV. RESEARCH CONCEPTUAL MODEL

The proposed conceptual model composed of Documents on User Requirements, Key-Words Indexing Term generating from the User Requirements Documentation. Then a Text Parser is used to make a comparison between a key-word and the requirement, then a Key-Point of the Story Card will be generated using Machine Learning Technique.

Validation Parser is used to validate the scenario constructed by student in the Story Card based on the related Key-Point. The scenario will be validated by the Validation Parser, if the Scenario is not valid then the student has to re-construct the scenario until it is valid. Figure 1 shows the research conceptual model of this study.

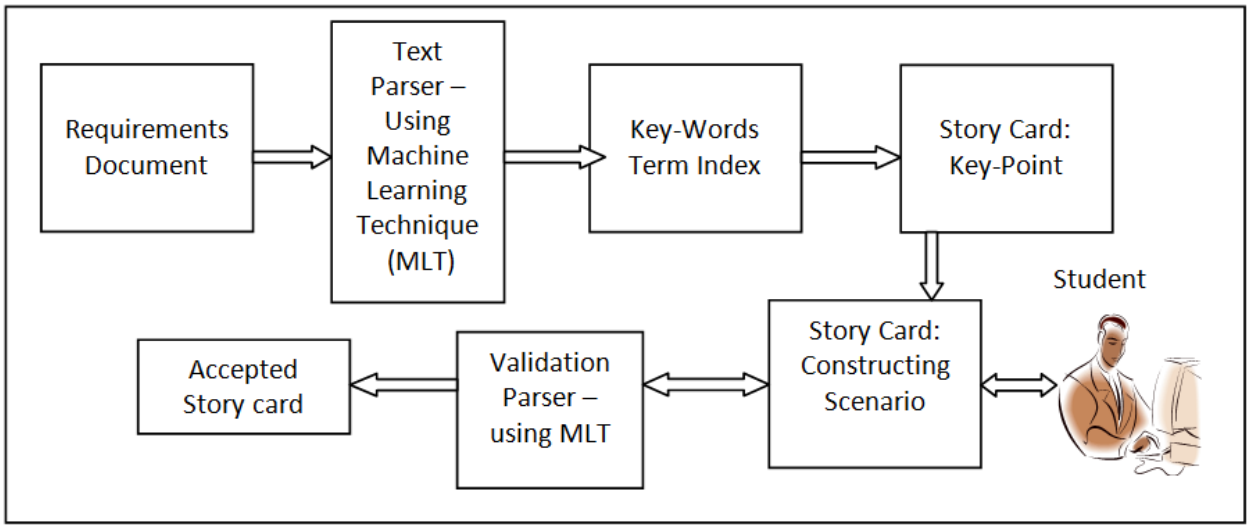

Figure 1. Research Conceptual Model

V. TEXT SIMPLICATION ARCHITECTURE

This section will explain in details about the creation of "Text Simplification" and the validation "Text Classification" of the story card. The Text simplification can be defined as any process that reduce the syntactic complexity of a text while attempting to preserve its meaning and information content (refer to Figure 2). The aim of text simplification in this research is to extract simple sentences from a text that can be used as a story card.

The tasks of the Text simplification can be divided into three stages *analysis, transformation and regeneration*. The architecture uses one module from each of these stages. The text analyzed the analysis module and then passed to the transformation module (Siddharthan, 2004). The transformation module applies rules for syntactic simplification and calls the regeneration module to address issues of text cohesion. When there is no further simplification, the transformation stage will generate the outputs of the simplified text.

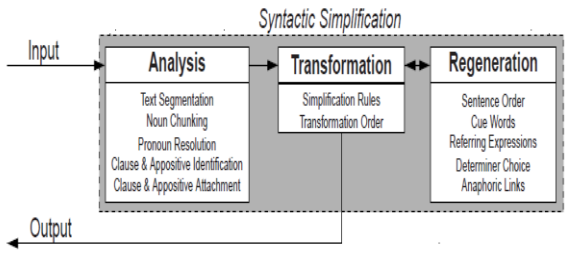

Figure 2.Text Simplification

VI. ANALYSIS AND RESULT

There are several modules should be achieved to complete the analysis stages which discusses in the following section.

A. Resolving Third-Person Pronouns

This model used Anaphora resolution algorithms. The algorithm preprocesses the text by annotating each noun phrase within formation about agreement values and grammatical functions. It then considers each noun phrase from left to right, forming a new co-reference class for non-pronominal noun phrases and adding pronouns to existing co-reference classes. At sentence boundaries, the algorithm halves the salience of each co-reference class and replaces each pronoun by its noun.

B. Deciding Clause Boundaries

The aim of this model is to train a machine learning system to identify the beginnings and ends of functional clauses. This is similar to a chunking or sentence boundary detection problem, but in this case clauses may also be nested. A text must be segmented into clauses before the detailed functional annotation that describes the theory can be applied. Usually, a clause consists of a verb phrase and its non-clause arguments.

C. Deciding Relative Clause Attachment

Relative clause attachment is a problem which has traditionally been approached in a parsing framework. However, determining what a relative pronoun refers to is not a problem that can always be solved in a syntactic framework. In particular, parser like Stanford typed parser (Marie-Catherine de Marneffe and Manning, 2008) .can easily detect the relation between the relative clauses. The following example can show how the parser identifies the relative clause:

Sentence: Ali, who was the CEO of the company, played golf.

By using the Stanford typed parser

nsubj(CEO-6, Ali-1)
nsubj(played-11, Ali-1)
cop(CEO-6, was-4)
det(CEO-6, the-5)
rcmod(Ali-1, CEO-6)
det(company-9, a-8)
prep_of(CEO-6, company-9)
root(ROOT-0, played-11)
dobj(played-11, golf-12)

The tags which attached to the words represented the grammatical structure of the sentences. For example, nsubj is represented the Nominal Subject which is consider as the relative clause of the sub sentences from the main text (Marie-Catherine de Marneffe and Manning, 2008).

D. Detecting the Preposition Words

Stanford provides another parser which is able to detect the preposition words and its location in the text. This parser called basic parser. Following example shows how this parser works
Sentences: Ali, who was the CEO of the company, played golf.

nsubj(played-11, Ali-1)
nsubj(CEO-6, who-3)
cop(CEO-6, was-4)
det(CEO-6, the-5)
rcmod(Ali-1, CEO-6)
prep(CEO-6, of-7)
det(company-9, a-8)
pobj(of-7, company-9)
root(ROOT-0, played-11)
dobj(played-11, golf-12)

The tag prep is representing the preposition words in the text. From the basic parser we will only take the preposition tag continents (Marie-Catherine de Marneffe & Manning, 2008).

E. Constructing a story card

Simple sentences from the text can easily be reconstructed after retrieving the relative clauses from the parser.

If the first relative clause nsubj(CEO-6, Ali-1) is taken and traversing the words which are in the tag " CEO-6, Ali-1" with the output from the typed parser. The result will be as followed:

cop(CEO-6, was-4)
det(CEO-6, the-5)
rcmod(Ali-1, CEO-6)
det(company-9, a-8)
prep_of(CEO-6, company-9)

From the basic parser we take the prep tag only prep(CEO-6, of-7). The number which attached to the words is representing the location of the word in the text. Arranging the words between the brackets based on the numbers will generate the result as following:

Ali-1 was-4 the-5 CEO-6 of-7 a-8 company-9
Ali was the CEO of a company
Taking the other relative clause it will produce
Ali-1played-11golf-12
Ali played golf

The results show that the proposed model and the Machine Learning technique adapted in the model can be used to generate the story card with priority and rules of requirements.

VII. CONCLUSION

This paper discusses about agile methodology and Extreme Programming and their use in software developments. It also illustrates some basic knowledge about Machine Learning and its usage in this research especially in text classification. The outcome of the research is a conceptual model of designing story cards using machine learning technique and a proposed architecture for text simplification in English. This paper also discusses about third person pronoun, deciding clause boundaries, detecting relative clauses from Stanford parser and constructing the sentences based on the Stanford parser

REFERENCES

Siddharthan, A. (2004), "Syntactic simplifaction and text cohesion". University Of Cambridge Computer Laboratory , [E-Book] Available: netLibrary e-book.

Beck, K. (2000). "Extreme programming Explained Embraced Change". Addison Wesley

Cohn, M. (2000). "User Stories Applied". Boston, Pearson.

Cohn, M. (2003). "User Stories Applied for Agile Software Development". Addison Wesley.

Kavitha, C. R. and Sunitha, M. T.(2011). "Requirement Gathering for small Projects using Agile Methods". IJCA Special Issue on Computational Science - New Dimensions & Perspectives (3), pp. 122–128.

Leffingwell, D. (2009). "Scaling Software Agility: Best Practices for Large Enterprises", Agile Chicago, IL.

Marie-Catherine de Marneffe and Manning C. D. (2008). "Standford typed Dependencies manual". Stanford University, September 2008. [E-Book] Available: netLibrary e-book.

Naumovich N .(2007). "User Stories for Requirements Elicitation". 2007. Plano, TX.

Tuck, D. France, R. and Rumpe, B. (2003). "Assumptions Underlying Agile Software Developmen". Journal of Database Management.