I J C T A, 10(8), 2017, pp. 99-111 ISSN: 0974-5572

A User Story Based Approach to Measure System Complexity in Agile Software Development

Preeti Kaur*, Ritu Sibal* and Sangeeta Sabharwal*

ABSTRACT

Agile software development has gained widespread popularity and acceptance by the software industry. System complexity computation is essential for all kinds of estimations required in software development. User stories are work products that are used to enlist user requirements in agile methods. The objective of this research paper is to find a user story based method to compute the complexity of the software system to be developed. In this paper a conceptual model to represent user stories and dependencies between them is proposed. A complexity metric is proposed to calculate system complexity. This metric can be used by agile teams along with other custom metrics to estimate effort, sprint planning etc. Theoretical validation of the proposed metric is also done and the results show that this metric effectively measures the user story complexity and system complexity.

Index Terms: Agile Software Development, User Story, Software Complexity, Complexity Metric.

1. INTRODUCTION

System complexity has been traditionally measured using various metrics such as Macabe'scyclomatic complexity metric [21], metrics for object oriented software Chidamberand Kemerer [6], etc. An event-based approach to calculate the system complexity was proposed by Singh *et al.* [24]. System complexity metric was given by the authors and they considered event as the basic unit for deriving the metric. Complexity of the web based system is measured in the paper by Sabharwal *et al.* [20]. The authors proposed a metric using the concept of fault tree analysis. System complexity may also be calculated from analysis models like the class diagram and use case diagram [6, 14, and 19]. A measure of the system complexity provides the designer the freedom to modify the design to meet effort- cost constraints prior to product development and also gives information regarding the effort needed to understand and implement the requirements [4]. Complexity measures derived from analysis models can be used to rearrange and prioritise requirements for structurally simple designs. Class complexity measures can be used to estimate the overall structural complexity of the design.

Traditional software development methods have given way to agile software development methods as they are value driven and adaptive. The basic principles of Agile Manifesto include "highest priority to customer satisfaction, communication and collaboration, accepting and responding to changes, working software over comprehensive documentation" [3]. The main ideas followed in agile software development are less time spent on documentation, implementation and testing of the core requirements early in the development cycle. In agile software development, the direction of work is assessed throughout the development cycle. This continuous assessment is done with iterations called sprints. In agile software development highest priority is given to the customer satisfaction and lays focus on lightweight methods, deliveries at regular intervals and constant customer interaction and corroborating feedback [3].

* Netaji Subhas Institute of Technology, Delhi University, Delhi, *Emails: preetikaur1@rediffmail.com, ritusib@hotmail.com, ssab63@gmail.com*

In agile projects, user story is a natural language description of the user-centric function or goal that the end software product must have. However, technical details of how to achieve that goal are not described in the user story. We observed from the literature that even though user story is widely used artifact during agile software development there are few methods available to assess complexity of user story. Heuristics like INVEST framework given by Wake [27], general guidelines in quality framework for agile given by Heck & Zaidman [12] are some of the qualitative metric approaches that exist for assessing user story quality. User stories have been popularly used to estimate story points, which serve as a basis to estimate effort and cost to implement the story [26]. To the best of our knowledge, no attempt has been made so far to find system complexity using user stories.

We have observed from the literature that requirement dependencies also exist in agile projects [20]. According to Carlshamre*et al.* [5] only a fifth of the requirements do not have dependencies. Therefore, majority of requirements are interdependent. Hence, there is a need to explore and manage interdependencies amongst user stories. A system with higher number of interdependencies amongst user stories is more complex as compared to a system which has fewer interdependencies amongst user stories, since for such a system interdependencies amongst user stories will also affect implementation of user stories.

In this paper, we propose a systematic approach to calculate system complexity from user stories. To do so, we initially propose a conceptual model of a user story. The proposed conceptual model helps in understanding the structure of user stories and interrelationships amongst them. Two kinds of relationships between user stories have also been captured through the notion of *require* and *optional* dependency. The user story conceptual model is further used to propose a template for writing user stories in a formal syntax. Since user stories are written in natural language so it is difficult to extract information for quantitative assessment of user story complexity. Representation of a user story in a formal template helps in calculation of the complexity of a user story. Thereafter, a dependency matrix is derived based on dependencies between user stories. To find complexity of a user story, metric named User Story Complexity Metric (USCM) is deduced from dependency matrix. Finally, system complexity metric USCM (SYS) is calculated by summation of the values of USCM across all user stories of a system. To establish the validity of the proposed metrics, theoretical validation of the metrics using Braind's framework [30] is done.

The organisation of the paper is as follows. Related work in this area is described in the section 2. Section 3 of proposed work describes the research methodology followed by user story conceptual model, the proposed user story template, dependency matrix and User Story Complexity Metric. The theoretical validation of the proposed metric is discussed in the section 4. The section 5 on case study describes implementation of the proposed approach on two different case studies and presents the discussion about them. The last section 6, presents the conclusion and future work.

2. RELATED WORK

Several attempts have been made by researchers to formally represent the user stories. A widely used user story format is given by Cohn [8], where the *who* it is for and *what* it is for layout is given. Wautelet *et al.* [29] studied various templates found in literature in order to reach unification in the concept's syntax, an agreement in their semantics as well as methodological elements increasing inherent scalability of the projects. Lucassen *et al.* [17] studied several available formats for writing user stories and formulated a quality user story framework having a list of quality features a user story should aim for. Also based on the framework they proposed a conceptual model of a user story. They described user story through four parts i.e. role, means, ends and format and the way to decompose each of these. They also proposed a tool called AQUSA (Automatic Quality User Story Artisan) tool based on the quality framework which shows the

defects and deviations from good practices in writing user stories. It is observed that although user stories are designed to be independent but still have dependencies. In literature Martakis and Daneva [20] are among the few authors who explored user story dependencies and conducted an empirical study. A hybrid approach to architecture between agile and plan-driven methods was given by them to represent and tackle dependencies.

3. PROPOSED WORK

In this section we discuss our research method, the proposed conceptual model of user story and the template to formally represent a user story and method to compute the User Story Complexity Metric (USCM).

3.1. Research Methodology

The way a user story is written is dependent on the writing style and preferences of the system analyst. Several templates are being used in an ad-hoc manner for agile software development. No uniform template is followed for user story writing. Also, very few formal or semi-formal templates are found in literature [17]. Many variants of user story are prevalent in real use and literature [29]. Previously, user story used to be a short, unstructured description of the requirements similar to use cases. Nowadays, user stories are written using a short format having the *who*it is for and *what* it is for layout [8].

Our aim was to study all possible user story templates whether formally described or being used in an ad-hoc manner. To achieve this goal, scientific publications, websites and blogs were extensively searched through internet, studied and analyzed. Based on that study, we proposed a user story conceptual model and the user story template. The dependency information is extracted from the conceptual model that forms the basis for proposing the complexity metric. The research methodology also includes theoretical evaluation of the proposed approach. The research process is shown in fig. 1.

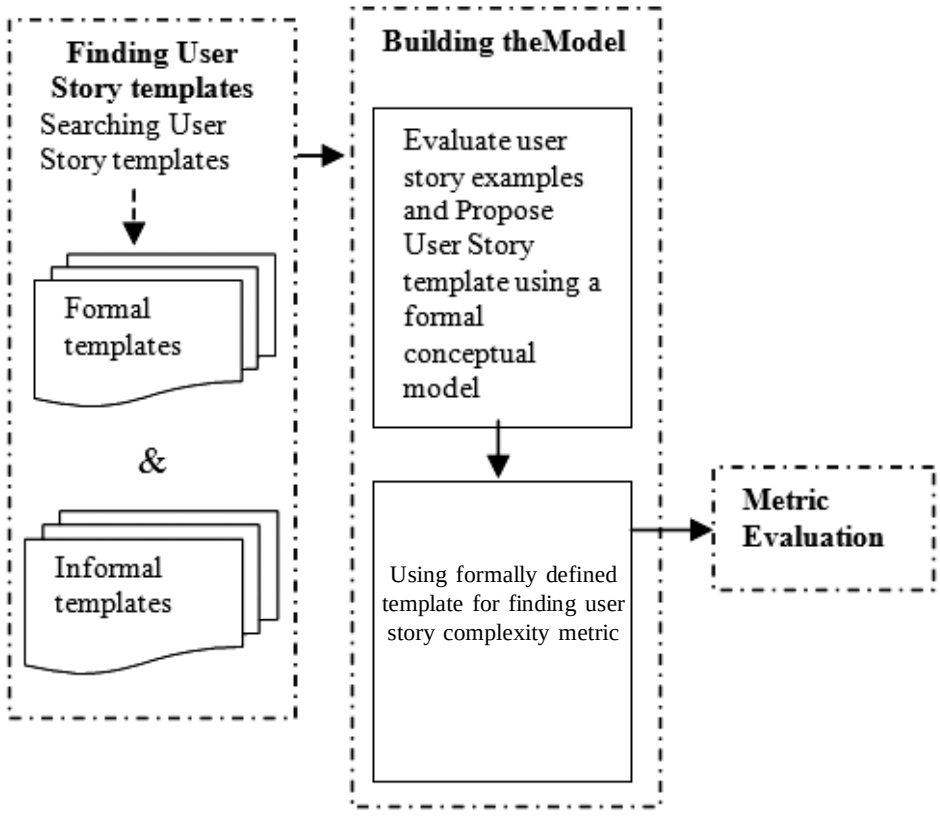

Figure 1: Research Methodology

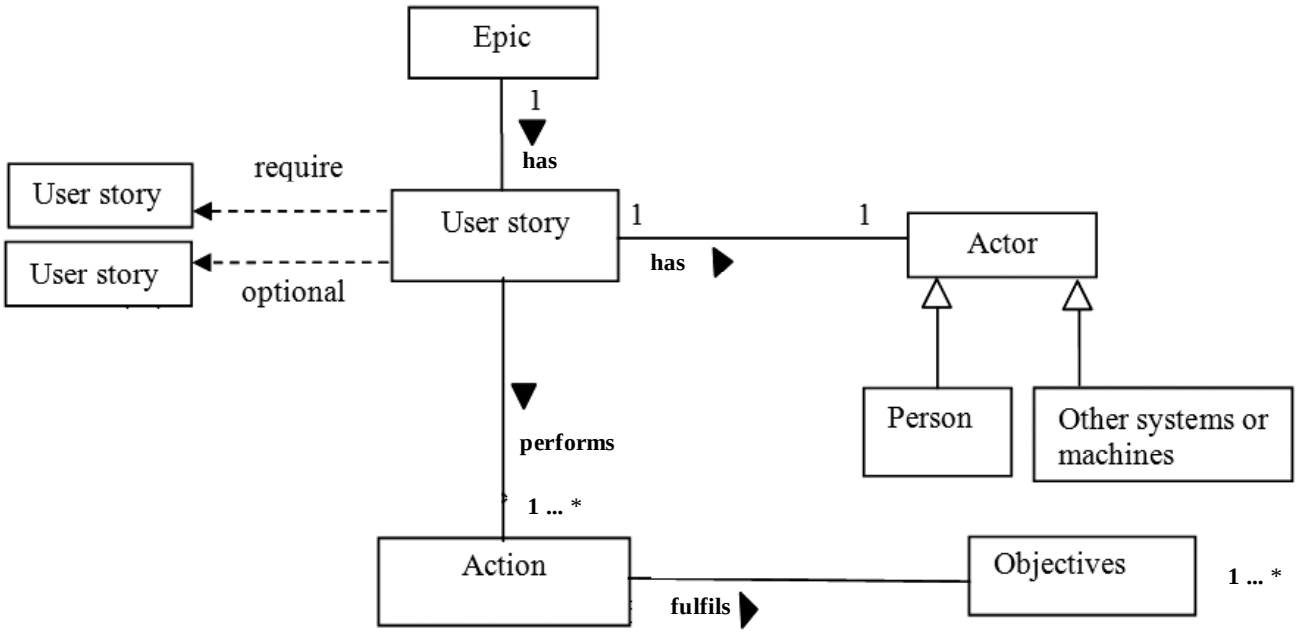

Figure 2: User Story Conceptual Model

3.2. User Story Conceptual model

The proposed User Story Conceptual Model is shown in fig. 2. This conceptual model will help in better understanding of all the components and attributes of user story and further help in determining the interdependencies among user stories.

The concepts of the user story conceptual model are as follows:

Epic: Epic represents a large user story that can be broken into small user stories. Epic has one or more user stories

User story: It represents the user defined functionality.

Actor: An actor is the person who will interact with the system. Actor in the user story could be a person or other system or other machine instead of a person. An actor is represented by a name.

Action: It is the actual functionality performed by the system in the particular user story and fulfilling a particular objective for the stakeholders. In this work we specify the objective in some user stories optionally for simplicity.

Further we propose two types of dependencies that can exist among user stories

Require dependency: A user story US1 will require another user story US2 to complete execution, before US1 could start as shown in fig 3. This dependency indicates an ordinal relationship in which one user story is dependent upon the completion of another user story.

We can explain *require* dependency with the help of an example. For the user story issue book in the Library Management System, a member record and book record must exist already so that book could be issued to a member of the library. Hence, user story for creating member record and creating book record should be executed before user story for issue book.

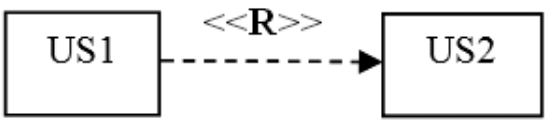

Figure 3: *Require* dependency

Optional dependency: Optional dependency exists between two user stories US1 and US2 if US1 may or may not require

US2 to complete execution before US1 can start as shown in fig 4. For example, in the Library Management System the user story return book is optionally dependent on the user story pay fine because it is not necessary that all the books will be returned after due date.

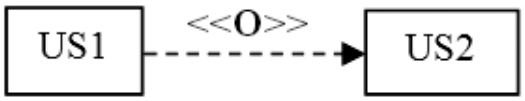

Figure 4: *Optional* dependency

3.3. User Story Template

User story template is the base for representing any user story in a formal manner. The components of user story template are same as the concepts of the user story conceptual model. In addition, we assign to every user story a User story id. User story id is a unique identification attached to each user story and it is called as identifier. Unique identification of each user story is ensured by the developer as this identification can be used later for mapping requirements or for determining the dependencies.

The user story template is given as follows:

US = < identifier, actor<p/m/os>, dependency<<<R>>/ <<O>>>

where

p: p is person,

m: m is machine,

os: os is other system

<<R>>: is require dependency,

<<O>>: is optional dependency

3.4. Dependency Matrix

In this section we show how to document user stories using user story template so as to extract information like dependencies between the user stories. These dependencies would be further used to calculate user story complexity and subsequently complexity of the system being developed. We represent the dependencies between the user stories in the form of N X N dependency matrix. Dependency matrix is a square matrix showing link between the system elements which are user stories here. The user stories are listed across the rows and columns of the dependency matrix. A value 0, 1 or 2 marked at each row column intersection in the matrix depending on the kind of relation that exists between user stories.

- If no dependency exists between the two user stories then a value 0 is assigned to matrix entry.
- If an *optional* dependency exists between two user stories, then a value 1 is assigned to the matrix entry.
- If a *require* dependency exists between two user stories, then a value 2 is assigned to matrix entry.

By assigning the dependency values as mentioned above we generate the dependency matrix for all user stories in the system to be developed. We have assigned value 2 to *require* dependency as there is a strong dependency relation among user stories. We have assigned value 1 to *optional* dependency as there is conditional dependency that exists between any two user stories. A user story may depend on other user story for its execution but not always.

3.5. User Story Complexity Metric

In this section we outline the procedure to compute the complexity of a user story using User Story Complexity Metric (USCM). This metric represents the relative complexity of individual user stories based on the dependency between user stories. The dependency matrix for the entire set of user stories is used to derive the complexity i.e. USCM for individual user story(s). These values are used to further calculate the system complexity i.e. USCM (SYS). USCM of a user story is based on the dependency matrix using the formula given below.

$$USCM_i = \frac{\sum_{j=1}^{N} D_{i,j}}{N} \tag{1}$$

Where

$USCM_i$ is the complexity of the i^{th} user story

$D_{i,j}$ is the dependency value of i^{th} user story with the j^{th} user story

N is the total number of user stories

From the computation of USCM for every user story, system complexity is computed using the formula given below.

$$USCM(SYS) = \frac{\sum_{i=1}^{N} USCM_i}{} \tag{2}$$

Where

USCM (SYS) is the system complexity

$USCM_i$ is complexity of a user story

N is the total number of user stories

4. THEORETICAL VALIDATION OF USER STORY COMPLEXITY METRIC

The goal of theoretical validation is to show that the proposed metric measure is acceptable. It also helps in assessing whether the metric actually measures what it purports. Framework given by Braind *et al.*[30] provides a theoretical validation criteria based on the mathematical propertiesthat formalize measurement concepts like size, length, complexity, cohesion, and coupling. We use Braind's framework to theoretically validate the proposed USCM and is discuss in this section.

4.1. Theoretical Validation using Braind's Framework

Braind *et al.* [30] proposed a formal mathematical measurement of properties provided by features such as size, complexity, cohesion and coupling and provides theoretical basis for software measurement. Braind's framework has the following concepts:

System and Modules: A system S is defined as a pair <E, R>, where E is the set of elements of S, and R is a binary relation on E($R \subseteq E \times E$) which represents relation between S's elements [30].

Considering system S as <E, R>, a subsystem $m = <E_m, R_m>$ is a module of S if and only if $E_m \subseteq E$, $R_m \subseteq (E_m \times E_m)$ and $R_m \subseteq R$ [30]. The elements of the module are connected to the elements of the rest of the system by relationships. Requirements of a system to be built are represented through user stories so, we can say that user story represents the system to be built and it qualifies the definition of 'System'.

Elements of the 'system' are user stories.

Relations are the dependency relationships between the user stories.

User Story Complexity Metric proposed in this paper is theoretically validated using Braind's framework of five properties as shown in table 1 and is given below:

Table 1
Braind's Framework Properties [30]

Property No.	*Property Description*
#1	Non Negativity
#2	Null Value
#3	Symmetry
#4	Module Monotonicity
#5	Disjoint Module Additivity

Property #1 ***Non Negativity****:* The complexity of the system S = <E, R>, represented by the User Stories is nonnegative [30] and is greater than zero i.e.

$$\text{Complexity}(S) \geq 0$$

In the proposed work, complexity is calculated by assigning weights to the dependency relationships which are always positive values, therefore the complexity is always non negative.

Property #2 ***Null Value*****:** The complexity of the system, S=<E, R> is null if R is empty [30] i.e.

$$\text{Complexity}(S) = 0$$

In the proposed work, complexity is calculated by assigning weights to the dependency relationships. The complexity of the system will be zero if no relationships exist among the User Stories.

Property #3 ***Symmetry*****:** The complexity of the system, S=<E, R> is not dependent on the convention followed to show the relationships between its elements [30] i.e.

$$\text{Complexity}(S) = \text{Complexity}(S^{-1})$$

In the proposed work, the complexity measure will remain same as it does not depend on the convention followed to represent the relationships and also it is not sensitive to direction of relationship.

Property #4 ***Module Monotonicity*****:** The complexity of the system S=<E, R> is greater than equal to the sum of the complexities of its two sub systems with no common relationships [30].

$$\text{Complexity}(S) \geq [\text{Complexity}(S_1) + \text{Complexity}(S_2)]$$

We consider a set of user stories from two sub systems of HR Management System shown in table 2. If we make sub systems of the HR Management System, it consists of HR manager and its user stories, Functional manager and its user stories. Both the subsystems have no common relations. USCM value of HR manager sub system is 0.16 and USCM value of Functional manager is 0.32. USCM value for the complete system is 0.48. Sum of complexity of the two sub systems is equal to the complexity of the HR Management System. Hence, the property #4 of module monotonicity is true for the proposed metric.

Table 2
User Stories for HR Management System

US1:	As a HR manager, I want to browse my existing quizzes so that I can recall what quizzes I have in place.
US2:	As a HR manager, I want to match an open position's skill with quiz topics so that I can create a relevant quiz.
US3:	As a HR manager, I want to add questions to the quiz so that I can create a relevant quiz.
US 4:	As a Functional manager I can access results of the quizzes so that I can get best candidates.
US 5:	As a Functional manager I can conduct the interview with the candidates so that I can select candidates for the open positions.
US 6:	As a Functional manager I can send profile of selected candidates for background checking.

Property #5 *Disjoint Module Additivity*: The complexity of the system S=<E, R> is equal to the sum of the complexities of its two disjoint sub systems [30].

$$\text{Complexity}(S) = [\text{Complexity}(S_1) + \text{Complexity}(S_2)]$$

This property #5 also holds true for the proposed metric USCM. Consider the example of HR Management System, sum of the complexities of the two sub systems is equal to the complexity of the HR Management System. Hence, the property of disjoint module additivity is true for the proposed metric.

The proposed USCM satisfies all properties given by Braind *et al.* [30] and proves that the metric measures the concept of complexity based on dependencies among the user stories. Hence, we can say that the proposed complexity metric USCM is theoretically sound and construct validity of the metric also exists.

5. CASE STUDY

Two case studies are taken to show the calculation of complexity metric. Library Management System and Online Food Ordering System have been selected to represent the practical application of our proposed work. The proposed technique is implemented for both the case studies and comparative analysis is also performed.

5.1. Case Study1: Library Management System

Library Management System deals with the automation of working of library. We consider a total of thirty-five user stories for the Library Management System shown in table 3. The user stories of the Library Management System are written using the User Story Template as shown in table 4. From the user story dependencies shown in table 4, a dependency matrix (shown in appendix 1) is generated and used to calculate the complexity metric.

User story US1 using the proposed template can be represented as follows

US = <identifier, actor, dependency >

US1: As a Librarian, I can issue a book to the member.

US1 = <1, L, <<R>> {5, 8}>

Identifier is a number here i.e. 1, 2 etc.

Actor is Librarian here i.e. L. Dependency is defined here as *Require* dependency or *Optional* Dependency i.e. <<R>> or <<O>> respectively.

Similarly all the user stories are represented in the above mentioned template and are shown in table 4.

Table 3
Use Stories for Library Management System

US1:	As a Librarian, I can issue a book to the member.
US2:	As a Librarian, I can view different categories of books available in the library.
US3:	As a Librarian, I can perform return book operation from the member.
US4:	As a Librarian, I can view list of books available in each category.
US5:	As a Librarian, I can add books and their information to the database so that I can update it.
US6:	As a Librarian, I can edit book information.
US7:	As a Librarian, I can access all the accounts of the members.
US8:	As a Librarian, I can create member account so that new members could be added.
US9:	As a Librarian, I can delete the member account.

US10: As a Librarian, I can edit member account.
US11: As a Librarian, I can calculate fine for a member and update his account.
US12: As a Librarian, I can register supplier.
US13: As a Librarian, I can renew supplier registration.
US14: As a Librarian, I can place order for books.
US15: As a Librarian, I can place order for journals and magazines.
US16: As a Librarian, I can compile order for books.
US17: As a Librarian, I can compile order for journal and magazines.
US18: As a Librarian, I can make payment to supplier.
US19: As a Librarian, I can generate member report.
US20: As a Librarian, I can generate supplier report.
US21: As a Librarian, I can renew membership of library members.
US22: As a Librarian, I can edit supplier information.
US23: As a Librarian, I can generate order report on yearly basis.
US24: As a Librarian, I can generate lost book report.
US25: As a Librarian, I can generate lost journal and magazine report.
US26: As a Librarian, I can view list of journals and magazines.
US27: As a Member, I can view different categories of books available in the library.
US28: As a Member, I can view list of books available in each category.
US29: As a Member, I can search for a particular book.
US30: As a Member, I can get a book issued.
US31: As a Member, I can view books issued to me.
US32: As a Member, I can put a request for a new book to be purchase in the library.
US33: As a Member, I can return a book to the library
US34: As a Member, I can view history of books issued to me.
US35: As a Member, I can own a membership account having membership number.

Table 4
User Stories of the Library Management System in the template format

US1: As a Librarian, I can issue a book to the member.US1= <1, L, {<<R>> : 5, 8}>
US2: As a Librarian, I can view different categories of books available in the library.US2= <2, L, {nil}>
US3: As a Librarian, I can perform return book operation from the member.US3= <3, L, {<<R>>>:1, 5, 8, <<O>>:11}>
US4: As a Librarian, I can view list of books available in each category.US4= <4, L, {<<R>>:5}>
US5: As a Librarian, I can add books and their information to the database so that I can update it.US5= <5, L, {nil}>
US6: As a Librarian, I can edit book information.US6= <6, L, {<<R>>: 5}>
US7: As a Librarian, I can access all the accounts of the members.US7= <7, L, {<<R>>: 8}>
US8: As a Librarian, I can create member account so that new members could be added.US8= <8, L, {nil}>
US9: As a Librarian, I can delete the member account.US9= <9, L, {<<R>>: 8}>
US10: As a Librarian, I can edit member account.US10= <10, L, {<<R>>: 8}>
US11: As a Librarian, I can calculate fine for a member and update his account.US11= <11, L, {<<R>>: 8}>
US12: As a Librarian, I can register supplier.US12= <12, L, {nil}>
US13: As a Librarian, I can renew supplier registration.US13= <13, L, {<<R>>: 12}>
US14: As a Librarian, I can place order for books.US14= <14, L, {nil}}>
US15: As a Librarian, I can place order for journals and magazines.US15= <15, L, {nil}>
US16: As a Librarian, I can compile order for books.US16= <16, L, {<<R>>: 14}>
US17: As a Librarian, I can compile order for journal and magazines.US17= <17, L, {<<R>>: 15}>

US18: As a Librarian, I can make payment to supplier.US18= <18, L, {<<R>>: 14, 15}>
US19: As a Librarian, I can generate member report.US19= <19, L, {<<R>>: 8}>
US20: As a Librarian, I can generate supplier report.US20= <20, L, {<<R>>: 12,13}>
US21: As a Librarian, I can renew membership of library members.US21= <21, L, {<<R>>: 8}>
US22: As a Librarian, I can edit supplier information.US22= <22, L, {<<R>>: 12}>
US23: As a Librarian, I can generate order report on yearly basis.US23= <23, L, {<<R>>: 16,17}>
US24: As a Librarian, I can generate lost book report.US24= <24, L, {<<R>>: 5, 7}>
US25: As a Librarian, I can generate lost journal and magazine report.US25= <25, L, {<<R>>: 27}>
US26: As a Librarian, I can view list of journals and magazines.US26= <26, L, {<<R>>: 27}>
US27: As a Librarian, I can add journal and magazine information to database.US27= <27, L, {nil}>
US28: As a Member, I can view different categories of books available in the library.US28= <28, M, {<<R>>: 8, 27}>
US29: As a Member, I can view list of books available in each category.US29= <29, M, {<<R>>: 5}>
US30: As a Member, I can search for a particular book.US30= <30, M, {<<R>>: 5, 35}>
US31: As a Member, I can get a book issued.US31= <31, M, {<<R>>: 35}>
US32: As a Member, I can view books issued to me.US32= <32, M, {<<R>>: 31}>
US33: As a Member, I can put a request for a new book to be purchase in the library.US33= <33, M, {<<R>>: 35}>
US34: As a Member, I can return a book to the libraryUS34= <34, M, {<<R>>: 31, 35, <<O>>: 11}>
US35: As a Member, I can own a membership account having membership number.US35= <35, M, {<<R>>: 8}>

Dependency Matrix for Library Management System is given in appendix 1. USCM value for each user story is computed using equation 1. We have observed from the dependency matrix given in appendix 1 that the individual complexity of the user stories varies between ranges 0 to 0.2. This shows that there are some user stories which are completely independent of other user stories and do not depend on any other user story. Using the formula given in equation 2, the USCM (SYS) metric value of the Library Management System is calculated and is 2.06.

5.2. Case Study 2: Online Food Ordering System

Online Food Ordering System deals with a system which can be used to order food online to a restaurant. The user stories for this case study are shown in table 5. Using the formula given in equation 2, the USCM (SYS) metric value of the Online Food Ordering System is calculated and is 1.76.

Table 5
Use Stories for Online Food Ordering System

US1: As a Customer, I should be able to place order for food online.US1= <1, C, {<<R>>:11}>
US2: As a Customer, I should get sms when status of the order changes.US2= <2, C, {<<R>>:1}>
US3: As a Customer, I should be able to make payment online. US3= <3, C, {<<R>>:1, 11}>
US4: As an Owner, I should be able to send sms to customer and notify if order is going to be late.US4= <4, O, {<<R>>:1, 11}>
US5: As a Customer, I should be able to place order for food online through phone.US5= <5, C, {nil}>
US6: As a Customer, I should be able to choose from the most popular menus of the week.US6= <6, C, {nil}>
US7: As a Customer, I should be able to see images of the food that can be ordered online.US7= <7, C, {nil}>
US8: As a Customer, I should be able to search reviews about the food.US8= <8, C, {<<R>>:9, 10}>
US9: As a Customer, I should be able to write reviews about the food.US9= <3, C, {<<R>>:1}>
US10: As an Owner, I should be able to publish reviews written by customers.US10= <10, Ow, {<<R>>:9}>
US11: As a Customer, I should be able to enter personal information for placing order.US11= <11, C, {nil}>
US12: As an Owner, I should be able to send special scheme information to the customers.US12= <12, Ow, {<<R>>:11, <<O>>:13}>
US13: As an Owner, I should be able to offer special schemes to the customerUS13= <13, Ow, {nil}>

USCM (SYS) metric value for case study 1 of Library Management System is calculated using the proposed approach and is 2.06. User story 3 has highest value of metric USCM, which indicates that it is dependent on more number of other user stories. Hence implementing user story 3 will require more effort. USCMvalue for user stories 2, 5, 8, 12, 14, 15 and 27 is zero. This shows that these user stories are not dependent on any other user story. Similarly, USCM (SYS) metric value for case study 2 of Online Food Ordering System is also calculated and is 1.76.

The results indicate that the Online Food Ordering System is having low value of USCM (SYS) as compared to Library Management System. Therefore, we can conclude that the Food Ordering System is less complex than the Library Management System.

6. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a template to represent user stories formally in agile paradigm. This contributes in improving the quality of representation of requirements being represented through the user story template. A formal representation of the user story through a conceptual model is also proposed. A complexity metric named User Story Complexity Metric (USCM) is proposed to find the complexity of a user story and in turn complexity of the system which could be used in calculating the story points in future. We have also applied our approach on two case studies to compute the metric in order to show the practical application of the proposed approach. We successfully validated the proposed metric USCM theoretically using Braind's framework [30] to establish the construct validity of the work.

In future we intend to propose a method to derive user story points from this metric which in turn will help the developer in estimating effort and cost.

REFERENCES

[1] P. Abrahammson, I, Fronza, R. Moser, J. Vlasenko, and W. Pedrycz, "Predicting Development Effort from User Stories", in the proceedings of International Symposium on Empirical Software Engineering and Measurement (pp. 400-403). 2011.

[2] Ananda Rao, and K. Reddy, "Detecting Bad Smells in Object Oriented Design Using Design Change Propagation Probability Matrix", in the proceedings of International Multi Conference of Engineers and Computer Scientists: Vol. 1. 2008.

[3] K. Beck, M. Beedl, A. van Bennekum, A Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas, "Manifesto for Agile Software Development", from http://www.agilemanifesto.org. 2007.

[4] D. Card, and W. Agresti, "Measuring Software Design Complexity". The Journal of Systems and Software. 185-197. 1988.

[5] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. Dag, "An Industrial Survey of Requirements Interdependencies in Software Product Release Planning", in the proceedings of Fifth International Symposium on Requirements Engineering (pp. 88-91). 2001.

[6] S.R. Chidamber, and C. F. Kemerer, "A Metric Suite for Object Oriented Design", IEEE Transactions on Software Engineering, 20(6), 476-493. 1994.

[7] E. Coelho, and A. Basu, "Effort Estimation in Agile Software Development using Story Points", International Journal of Applied Information Systems, 3(7), 7-10. 2012.

[8] M. Cohn, "Use Stories Applied for Agile Development", Addison-Wesley.2004.

[9] J. Desharnais, B. Kocaturk, and A. Abran, "Using COSMIC Method to Evaluate the Quality of the Documentation of Agile User Stories", in the proceedings of 21st International Workshop on Software Measurement and 6th International Conference on Software Process and Product Measurement (pp. 269-272). 2011.

[10] A. Gomez, G. Rueda, and P. Alarcon, "A Systematic and Lightweight Method to Identify Dependencies Between User Stories", Lecture Notes in Business Information Processing: Vol. 48. (pp. 190-195). 2010.

[11] N. C. Haugen, "An Empirical Study of Using Planning Poker for User Story Estimation", in the proceedings of AGILE 2006 Conference. (pp. 23-34). 2006.

[12] P. Heck, and A. Zaidman, "Quality Framework for Agile Requirements: A Practitioner's Perspective", Report TUD-SERG 2014. 2014.

[13] V. Heikkila, C. Lassenius, D. Damian, and M. Paasivaara, "A Mapping Study on Requirements Engineering in Agile Software Development",in the proceedings of 41st Euromicro Conference on Software and Advanced Applications (pp. 199-207). 2015.

[14] B. Henderson-Sellers, D. Zowghi, T. Klemola, and S. Parasuram, "Sizing Use Cases: How to Create a Standard Metrical Approach",in proceedings of the 8th International Conference on Object-Oriented. Information Systems. (pp. 409-421).2002.

[15] S. Kang, O. Choi, andJ. Baik, "Model-based Dynamic Cost Estimation and Tracking Method for Agile Software Development",in the proceedings of 9th International Conference on Computer and Information Science. (pp. 743-748). 2010.

[16] E. Kupiainen, M. Mantyla, andJ. Itkonen, "Using Metrics in Agile and Lean Software Development- A Systematic Literature Review of Industrial Studies", Journal of Information and Software Technology, 62, 143-163. 2015.

[17] G. Lucassen, F. Dalpiaz, J. M. van der Werf, and S. Brinkkemper, "Forging High-Quality User Stories: Towards a Discipline for Agile Requirements",in the proceedings of International Conference on Requirements Engineering. (pp. 126-135). 2015.

[18] A. Lucia, A, andA. Qusef, "Requirements Engineering in Agile Software Development", Journal of Emerging Technologies in Web Intelligence, 2, 212-220. 2010.

[19] M. Marchesi, "OOA Metrics for the Unified Modeling Language", in proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering. (pp. 67-73). 1998.

[20] A. Martakis,andM. Daneva, "Handling Requirements Dependencies in Agile Projects: A Focus Group with Agile Software Development Practitioners", in the proceedings of International Conference on Requirements Engineering. (pp. 273-282). 2005.

[21] T. J. McCabe, "A Complexity Measure", IEEE Transactions on Software Engineering, 2, 308-320. 1976.

[22] D. North, Blog post, The Perils of Estimation. from http://dannorth.net/2009/07/01/the-perils-of-estimation. 2009.

[23] S. Sabharwal, R. Sibal, and C. Sharma, "Deriving Fault Trigger Metric for Web Based Systems", Journal of Web Engineering 15(1-2), 1-28.2016.

[24] S. Singh, S. Sabharwal, and J. P Gupta, "Events-An Alternative to Use Case as starting point in Object-Oriented Analysis",in the proceedings of 2nd International Conference on Emerging Trends in Engineering and Technology. (pp. 1004-1010). 2009.

[25] Y. Singh, S. Sabharwal, andM. Sood, "A Systematic Approach to Measure the Problem Complexity of Software Requirements Specifications of an Information System", Journal of Information and Management Sciences, 1(5), 69-90. 2004.

[26] S. Srinath, R. Ramakrishnan, andJ. Ram, "Generic Model for Semantics-Based Versioning in Projects",in IEEE Transactions on Systems, Man, and Cybernetics- Part A: Systems and Humans. 3, 108-123. 2000.

[27] B. Wake, "INVEST in Good Stories and SMART Tasks", http://xp123.com/articles/invest-in-good-stories-and-smart-tasks. 2003.

[28] X. Wang, L. Zhao, Y. Wang, andJ. Sun, "The Role of Requirements Engineering Practices in Agile Development: An Empirical Study",in the proceedings of Asia Pacific Requirements Engineering Symposium, Vol. 432. (pp. 195-209). 2014.

[29] Y. Wautelet, S. Heng, M. Kolp, andI. Mirbel, "Unifying and Extending User Story Models", in the proceedings of International Conference on Advanced Information Systems Engineering, Vol. 1. (pp. 267-280). 2014.

[30] L. Braind, S. Morasca, andV. Basili, "Property Based Software Engineering Measurement", IEEE Transactions on Software Engineering, Vol. 30. (pp. 120-140). 1999.