Improving Scrum User Stories and Product Backlog Using Work System Snapshots

Full Paper

Narasimha Bolloju
LNM Institute of Information Technology
narsi.bolloju@lnmiit.ac.in

Steven Alter
University of San Francisco
alter@usfca.edu

Anupriya Gupta
LNM Institute of Information Technology
y14uc044@lnmiit.ac.in

Simran Gupta
LNM Institute of Information Technology
y14uc297@lnmiit.ac.in

Sanyam Jain
LNM Institute of Information Technology
y14uc246@lnmiit.ac.in

Abstract

Lack of domain knowledge is often considered a reason for improper elicitation and specification of requirements of a software system. The work system method helps analysts understand the business situation to be supported by the software system. This research investigates the effects of preparing a work system snapshot, a key artifact of the work system method, on the quality of initial requirements specifications represented within the Scrum methodology. Those specifications take the form of a product backlog, a set of user stories to be addressed). The findings from a controlled experiment conducted with 165 students in a software engineering course indicate that the preparation of work system snapshot results in a significant reduction in invalid user stories and increase in valid user stories in the product backlog.

Keywords

Scrum, work system method, product backlog, user stories

Toward Better Initial Specifications in Agile Development

Elicitation and specification of requirements representing the functionality to be incorporated into an information system to be developed is often a challenging task. Frequently, important system functionalities are omitted or misstated (e.g., Rubinstein, 2007). In addition, functionality that should not be present in the developed system sometimes is included. Identifying a good set of system requirements calls for understanding the larger system that that the information system serves (Ackoff, 1971, 1973; Checkland, 1999; Churchman, 1970). When using the work system method (Alter, 2006, 2013), that system is the smallest work system that exhibits the problems or opportunities that launched the information system project. A work system is a system in which human participants and/or machines perform work (processes and activities) using information, technology, and other resources to produce specific product/services for specific internal and/or external customers. Understanding of the work system that is to be supported by an information system is a good starting point for specifying the functionality provided by that information system.

In recent years, Scrum has become a widely used agile methodology for information system development. VersionOne (2016) reports that nearly 70% of 3880 respondents from a broad range of software development community practice Scrum or Scrum/XP hybrid. A set of user stories (simplified requirements identifying a user, what the user wants, and why) constitutes the basic building blocks for describing the requirements or functionality of the information system to be developed. This methodology operates through a series of sprints, each of which produces operational software. A prioritized set of user stories is designated as the "product backlog" at the beginning of the first sprint, and the product backlog is updated after each sprint as new requirements are identified. Although the product backlog is revised at the beginning of each sprint, the product backlog created prior to the first sprint is a good indicator of the scope of the software to be developed.

Based on earlier research on employing work systems method as a front-end for object oriented systems analysis and design (Alter and Bolloju, 2016), we believe that creating a work systems snapshot - an important artifact of the work system method - prior to identifying user stories in the product backlog at the beginning of the first sprint can help in understanding of work system and hence can result in identification of a better-focused product backlog. This research study investigates the possible effect(s) of creating a work system snapshot on the quality of the initial product backlog produced in an agile project.

The remainder of this paper is organized as follows. First it provides background about requirements identification and modeling, the work system method (WSM), and Scrum methodology. Next it presents the research question and associated hypotheses, plus the details of the research method. Next, the paper presents findings from a controlled experiment conducted in a software engineering course. The experiment is designed to study the impact of creating work system snapshots prior to specification of the initial product backlog that will be used in the first Scrum sprint. The last section discusses the possible implications for research and practice, and limitations of this study.

Background

Requirements identification and modeling

The importance and difficulty of determining the scope or required functionality of an information system have been discussed many times. Nuseibeh and Easterbrook (2000, p44) point out that "requirements modeling and analysis cannot be performed adequately in isolation from the organizational and social context in which any new [information] system will have to operate." Inherent difficulties in requirements determination are related to the nature of expected information system functionality, decisions about the information system's boundary, environmental conditions, and understandability and usability of resulting artifacts (Cheng and Atlee, 2007).

Stirna and Kirikova (2008) suggest combining enterprise modeling and agile software development which contributes to enhanced transparency across the organizational development and software development processes. In proposing a goal-oriented analysis technique, Mylopoulos et al (2001) highlight the need for exploring alternatives for achieving organizational and technical objectives prior to elaborating those requirements during subsequent phases. Since the information system's scope defines the functionality to be provided, under-scoping or over-scoping an information system can lead to architectures with inappropriate functionality (Buschmann, 2009). Loucopoulus and Kavakli (1995) advocate explicit modeling of the enterprise to gain insights into the purpose of the system to be developed by incorporating different viewpoints of various actors affected by the system.

Scrum Methodology

The growing popularity of agile methods in project management had led many organizations to adopt this approach to software development. The iterative and incremental nature of agile makes it adaptable by explicitly supporting an ability to respond to changes. That makes it easier to incorporate new technology when required. The entire project is broken down into sprints of short duration. In contrast, traditional software development sometimes was organized around large projects that took months or years before producing results.

Scrum is the most popular and widely used agile methodology. Its relative simplicity and ease of use differentiate it from other methods. It is well structured and is designed to keep project participants motivated throughout the project. It also involves users or user representatives directly. As noted earlier, Scrum operates through a series of sprints, each of which produces operational software. General direction for the sprints is provided by a "product backlog", a prioritized list of user stories to be addressed by the new information system. The prioritization provides guidance about which user stories are to be addressed first.

Ideally, the product backlog should be specified in a way that guides an efficient and effective project, i.e. a project that includes a series of deliverables that each can be completed in a short time and that in combination lead to the desired operational improvements. Atomic user stories are integral to producing a product backlog that provides successful guidance for the project. For example, user stories should contain an appropriate level of detail, and none of the user stories in the product backlog should take months to complete. In some cases, a larger user story can be split into smaller user stories so that it will be possible to complete sprints quickly and to adjust the product backlog accordingly (Cohn, 2008). When that cannot be done, the user story can be augmented with additional information.

Work Systems Method

The work system approach for understanding and analyzing work systems from a business perspective (Alter, 2006; 2013) starts from a different viewpoint and addresses some of the issues that user stories and product backlogs do not address well. User stories are about ways in which users will use an information system. The Scrum product backlog is a prioritized list of user stories. From a business viewpoint, production of software is only part of the story. The main priority from a business viewpoint is to improve the operation of a work system in which the new software will be used. That calls for linkage between what the work system needs and how the Scrum will generate software that supports work system needs. Table 1 is an example of a work system snapshot, one of WSM's primary tools for understanding and analyzing systems in organizations.

The work system framework, a central idea in WSM outlines nine elements of a basic understanding of a work system: customers, product/services, processes and activities, participants, information, technologies, environment, infrastructure, and strategies. (Alter, 2006, 2013). Someone who is trying to understand a work system that might be supported by an information system needs at least a basic understanding of who are the work system participants, what are the processes and activities, what product/services are produced by the work system, who are the work system's customers, and so on.

The basic idea of a work system snapshot (see example in Table 1) is to use six central elements of the work system framework to summarize the scope and operation of the work system that that is to be supported by the new information system. This particular work system snapshot summarizes aspects of the case study (see Appendix) that was used in the experiment described below. It was produced by one of the undergraduate students who served as coders for the experiment, which studied the impact of producing a work system snapshot before producing a Scrum product backlog. Each student subject in the treatment group produced a work system snapshot before trying to identify user stories and prioritize those user stories as a Scrum product backlog.

Producing a work system snapshot calls for describing the work system rather than the information system that will support the work system. A work system snapshot does not start with a distinction between the work system and the information system that supports it. Instead, it assumes that the work system simply uses certain software and hardware that listed under technologies. The boundaries of the information system are determined later in the analysis, after the problems or opportunities at hand are clarified and after the scope of the relevant work system is discussed.

Customers	Product/Services
• Editor • Author • Adviser	• Review of article • Modification of articles • Payment options • Subscribing new articles

Major Activities and Processes
• The author submits the articles on website and can view his subscriptions. • The adviser gives detailed reviews and comments for the articles. • The editor gives a final review of the article before publishing. • The assistant keeps track of the new subscription requests and submissions and manages them for further reviews. • The publisher keeps track of the payment and payment methods for the new subscriptions. • The subscriber either pay by cheque or credit card.

Participants	Information	Technologies
• Editor • Adviser • Assistant • Publisher • Author • Subscriber	• Submitted articles • Subscriptions • Reviewed articles • Payment of subscriptions	• DBMS for website • Word for editing • Email server classification • Payment gateway

Table 1. Sample Work System Snapshot Prepared by a Student

Research Question and Hypotheses

Since work system snapshots offer a simple mechanism for summarizing a business situation and thereby facilitating deeper analysis of the views of different stakeholders, we anticipated that preparing a work system snapshot prior to the specification of functional requirements of a system to be developed might result in better specifications. In this study, we focus on the impact of incorporating a work system snapshot into the process of specifying user stories and the Scrum product backlog before or during the first sprint of an agile development project. The research question for this study addresses a practical issue that could lead to better results in agile development projects:

> *Does the creation of work system snapshots prior to specifying the user stories lead to the production of better initial product backlogs (prioritized lists of user stories) for a given problem statement?*

We use the conceptual model quality framework suggested by Lindland et al (1994) to assess the quality of the product backlogs (i.e., prioritized user stories). This quality framework describes model quality using syntactic, semantic and pragmatic categories. Bolloju and Leung (2006) applied this framework for assessing the quality of three popular UML artifacts. Since the product backlog is typically represented as a table of user stories, the syntactic and pragmatic quality aspects of this framework were not relevant for this research. Consequently, our research focuses only on the semantic category, which emphasizes validity and completeness. The validity aspect is related to whether the user stories are relevant to the problem domain (i.e., functionality that is not required is not captured). The completeness aspect is related to extent to whether required user stories are identified (i.e., required functionality is captured).

Hypotheses

The hypotheses for this research are based on the following ideas:

- A modified version of an existing exercise from a textbook implies a number of user stories that can be identified and prioritized as a product backlog for a hypothetical Scrum project.
- Student answers can be evaluated in terms of completeness and validity.

- Each user story can be evaluated for validity, i.e. for the extent to which it is relevant to the problem domain.
- A group of qualified observers can identify an "expected product backlog" that can serve as the basis for evaluating completeness of student answers to the same exercise.

We identify two null hypotheses pertaining to completeness and validity of product backlog represented by user stories. The first null hypothesis (H1) focuses on completeness as evaluated based on the extent to which user stories in a specific product backlog belong to the user stories in the expected product backlog:

H1 - The completeness of a specified product backlog with respect to the expected product backlog is not affected by the process of creating a work system snapshot prior to the specification of the product backlog.

The validity component of semantic quality is investigated using two sub-hypotheses.

H2a - The proportion of valid user stories specified in product backlog is not affected by the process of creating a work system snapshot prior to the specification of product backlog.

H2b - The proportion of invalid user stories specified in product backlog is not affected by the process of creating a work system snapshot prior to the specification of product backlog.

Method

Experimental Design

Task

In this experiment, each subject was expected to prepare a list of user stories, i.e., the product backlog, with prioritization for the CompJournal case, a hypothetical case about a publisher of computer journals (see Appendix A). Each subject in the treatment group was required to prepare a work system snapshot prior to preparing the product backlog.

The experimental task was performed as a classroom test (which accounted for 3% of the final grade) in a software engineering course. The subjects from both control and treatment groups - seated in different classrooms - were given equal time (50 minutes) to complete the assigned task and to respond to a short questionnaire. The time provided for this task was estimated to be sufficient for this task based on an earlier use of this case study, an adaptation of a case from Yourdon and Argilla (1996). One of the authors had used the case at a different university for a modeling exercise involving different requirements.

Subjects and groups

This experiment was carried out as a part of an undergraduate course on software engineering at a private technological institution in Jaipur, India. 165 third year students who were computer science and engineering (CSE) and computer and communications engineering (CCE) majors participated in this experiment. Students were randomly assigned to the control group (n=84) and treatment group (n=81). All of these students had completed a similar set of courses on programming, data structures and algorithms (except the CSE students who had completed a database course as well). Thus, the academic background and prior knowledge of these full-time students (age 19-21 years and very little work experience) was quite similar. As part of the course the students had previously prepared a detailed product backlog as a class exercise, and had also created product backlogs for a semester-long undergraduate project prior to this experiment.

Procedure

The control group and treatment group were seated in two separate classrooms on the same day. Each subject in the control group classroom received a copy of the case study (shown in Appendix A) and a two-page template for recording the product backlog. The teaching assistant (TA) for the course supervised this activity. The treatment group received the same case study, the same two-page template for the product backlog, and an additional one-page template for producing a work system snapshot.

The treatment group had been introduced to the WSM by the instructor. During that previous session, students working in small teams practiced writing up work system snapshots for a different case that had been used in previous class sessions. During the experiment, the students in the treatment group prepared a work system snapshot and a corresponding product backlog for the same case study given to the control group. The students in the treatment group had 15 minutes for preparing the work system snapshot and then had 30 minutes for preparing product backlog.

The control group and treatment group had the same amount of time (total 50 minutes with last five minutes for completing a questionnaire) for producing the required specifications. During the entire activity, the research team ensured that no exchange of information of any kind among the subjects within the groups and between the groups.

Expected Product Backlog and Matching Criteria

Both validity and completeness were evaluated by comparing student answers with an expected product backlog (i.e., the set of required user stories) identified after a thorough discussion of product backlogs prepared independently by three undergraduate students who are co-authors of this paper. Reconciliation and consolidation of the three product backlogs resulted the expected product backlog (see Appendix B).

As illustrated in Figure 1, a well-defined criterion was used for matching each product backlog prepared by the subjects against the expected product backlog. A user story belonging to a given product backlog is considered fully matched with one of expected user stories when the actor, the functionality ("I want to") and purpose ("so that") are nearly identical. A subject's user story is considered partially matched whenever part of that user story matches a user story from the expected product backlog. Partially matched user stories are converted to a fully matched stories when a subsequent user story complements or adds to the previously matched user story. As part of this matching exercise, synonymous words (e.g., like "rejecting an article" and "filtering the articles") are considered equivalent. Thus, this criterion provided for matching multiple finely specified user stories with a single expected user story.

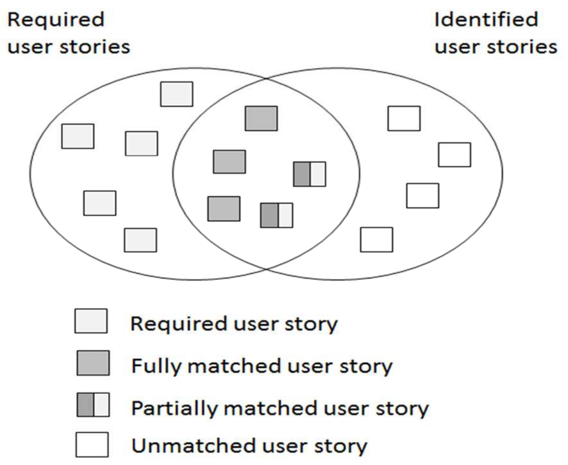

Figure 1 – Completeness and validity measures

Coding Procedure

For the first round of coding, product backlog specifications prepared by five subjects from the control group and five from the treatment groups were randomly selected for independent coding by the three student co-authors who were not aware of the research hypotheses until the entire coding was completed. The interrater reliability Fleiss' Kappa - calculated using ReCal3 (2017) - for this round of coding was 0.384 with an average pairwise agreement of 72.5%. As this extent of agreement could only be interpreted as fair, the student co-authors engaged in a second round of calibration by coding another set of 10 product backlog specifications after discussing the differences noticed in ten from the first round of coding. Since the resulting Fleiss Kappa (0.597 with an average pairwise agreement of 80.06%) achieved high or moderate agreement, all the remaining product backlog specifications were distributed equally among the three raters for subsequent coding.

Measures

The completeness measure is the percentage of user stories listed in a given product backlog that matched with the expected product backlog. This measure compares the number of fully matched user stories plus half the number of partially matched stories (in the intersection area in Figure 1) to the expected number of user stories (Appendix B). The validity measure compares the number of fully matched user stories plus half the number of partially matched stories to the total number of user stories identified by a subject. The invalidity measure compares the number of unmatched user stories to the total number of user stories identified by a subject. Thus, these three measures indicate the quality of a product backlog as compared to the expected (required) product backlog.

Results

Table 2 provides descriptive statistics for the control and treatment groups, and also the T-test results comparing the means. The treatment group produced significantly more valid user stories. The total number of user stories specified and the number of unmatched user stories are significantly lower for the treatment group. On the other hand, there is no significant difference in the numbers of fully matched and partially matched user stories.

- The null hypotheses H1 was accepted based on the lack of a significant difference in completeness between the treatment and control groups.
- The null hypotheses H2a was rejected based on the significantly higher proportion of valid user stories in the treatment group.
- The null hypotheses H2b was also rejected based on the significantly lower proportion of invalid user stories in the treatment group.

	Control Group (n=74)		**Treatment Group (n=71)**		**T-test Result**
	Mean	**Std Dev**	**Mean**	**Std Dev**	
# of user stories specified	17.59	4.49	13.14	4.02	Significant (p < 0.001)
# of unmatched user stories	12.57	4.63	8.59	3.53	Significant (p < 0.001)
# of partially matched user stories	1.86	1.22	1.72	1.12	Not significant (p = 0.45)
# of fully matched user stories	2.16	1.59	2.41	1.50	Not significant (p = 0.34)
Completeness (percentage)	23.80	11.77	25.14	11.09	Not significant (p = 0.48)
Invalidity (percentage)	70.37	14.74	64.22	14.04	Significant (p < 0.05)
Validity (percentage)	17.98	10.40	26.15	12.58	Significant (p < 0.001)

Table 2: Descriptive statistics and T-test results

Discussion

The results indicate that the preparation of work system snapshot prior to product backlog specification resulted in product backlogs with higher quality. The treatment group identified significantly more valid user stories and fewer invalid user stories. There was no significant difference in the completeness of product backlogs even though the treatment group that produced the work system snapshot had an additional task to perform during the same 50 minutes that the control group had. (Note that the measure of completeness was related to the number of user stories in the product backlog, not the number of valid user stories.) The control group flailed around more than the treatment group, and consequently the control group produced more user stories of lower quality. However, both the groups had high invalidity percentages probably due to their lack of business experience.

The significant difference between the control and treatment groups in the number of user stories belonging to the product backlog may be related to an important phenomenon. It may result from the way the work system snapshot helped subjects in the treatment group produce more effective user stories. As an example, the participants section in the work system snapshot seemed to be a reminder to the treatment group to include the subscriber as an important actor and subscription as an important functionality. Overall, we can conclude that the preparation of work system snapshot prior to product backlog specification resulted in higher quality product backlogs as the treatment group attained more business focus compared to the control group.

Although the results from this experimental study suggest potential benefits from applying ideas from WSM in agile development, it is important to identify limitations of the research. First, subjectivity in interpreting user stories prepared by the subjects is an obvious issue. As was discussed, we addressed that issue through careful calibration of the coding between the three coders, but still see subjectivity as a significant limitation. Another limitation is that most students had little business background and therefore may have had some difficulty interpreting the case study. A final issue is that the subjects in the control group were given entire 50 minutes to prepare the list of user stories. This may have indirectly encouraged the subjects to write more user stories and hence resulted in more invalid user stories.

In spite of the above limitations, the encouraging results from this experimental study can serve as the basis for future research. It would be especially valuable to repeat this experiment with experienced systems analysts. Results with experienced analysts could further demonstrate the benefits of using the work system snapshots, or, alternatively, could show that they already consider and incorporate the concepts that are captured in work system snapshots and related tools. Any replication of this research with experienced analysts should include use of questionnaires to learn more about their views of the potential value of using a work system snapshot at the beginning of an agile project. Finally, careful examination of the results from our 165 subjects might identify specific user stories that were understood less well. That might be a step toward categorizing user stories and identifying types of user stories that need to more focus, at least by novice analysts.

Conclusion

The approach of focusing on the work system by using a work system snapshot before launching into a typical agile scrum could have substantial benefits. In the context of agile development projects, the work system approach helps in maintaining focus on the business goal of improving work system performance, rather than IT project goal of producing software. It does this by providing an intermediate artifact that can help in clarifying thoughts of systems analysts before producing required user stories for a given system. User stories and product backlogs are a step in that direction, but still focus much more on the IT goal of producing software rather than the business goal of improving work system performance. Regardless of whether it might prove useful to question the fundamental idea of user stories, it is possible that research about combining work system ideas with user story writing could lead to better modeling and better communication between business and IT professionals during systems analysis and design.

REFERENCES

Ackoff, R. L. 1971. "Towards a system of systems concepts." *Management Science* (*17*:11), pp. 661–671. doi:10.1287/mnsc.17.11.661

Ackoff, R. L. 1991. "Science in the systems age: Beyond IE, OR, and MS." in *Facets of systems science* (pp. 325-335). Springer US.

Alter, S. 2006. *The work system method: connecting people, processes, and IT for business results*. Work System Press.

Alter, S. 2013. "Work System Theory: Overview of Core Concepts, Extensions, and Challenges for the Future." *Journal of the Association for Information Systems* (14:2), pp. 72-121.

Alter, S., and Bolloju, N. 2016. "A Work System Front End for Object-Oriented Analysis and Design." *International Journal of Information Technologies and Systems Approach (IJITSA)* (*9*:1), pp. 1-18.

Babar, M. A. 2009. "An exploratory study of architectural practices and challenges in using agile software development approaches." in *Software Architecture, 2009 and European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference* (pp. 81-90). IEEE.

Bolloju, N., and Leung, F. S. 2006. "Assisting novice analysts in developing quality conceptual models with UML." *Communications of the ACM* (*49*:7), pp. 108-112.

Buschmann, F. 2009. "Learning from Failure, Part 1: Scoping and Requirements Woes." *IEEE Software* (26:6), pp. 68–69.

Checkland, P. 1999. *Systems thinking, systems practice: includes a 30-year retrospective*, Chichester, UK: John Wiley and Sons.

Cheng, B.H.C. and Atlee, J.M. 2007. "Research directions in requirements engineering." in *Proceedings of Future of Software Engineering,* pp. 285–303.

Churchman, C.W. 1970. Operations research as a profession." *Management Science* 17(2), pp. 37–53.

Cohn, M. 2008. "Writing the Product Backlog Just Enough and Just in Time." *Scrum Alliance Weekly Column, February, 12*.

Conboy, K., Coyle, S., Wang, X., and Pikkarainen, M. 2010. "People over process: key people challenges in agile development." *IEEE Software* (*99*:1), pp. 47-57.

Estler, H. C., Nordio, M., Furia, C. A., Meyer, B., and Schneider, J. 2014. "Agile vs. structured distributed software development: A case study." *Empirical Software Engineering* (*19*:5), pp. 1197-1224.

Lindland, O.I., G. Sindre, and A. Solvberg. 1994. "Understanding Quality in Conceptual Modeling." *IEEE Software* (11:2), pp. 42-49.

Loucopoulos, P., and Kavakli, E. 1995. "Enterprise modelling and the teleological approach to requirements engineering." *International Journal of Cooperative Information Systems* (4:01), pp. 45-79.

Mylopoulos, J., Chung, L., Liao, S., Wang, H. and Yu, E. 2001. "Exploring alternatives during requirements analysis." *IEEE Software* (18:1), pp. 92-96.

Nuseibeh, B., and Easterbrook, S. 2000. "Requirements engineering: a roadmap." in *Proceedings of the Conference on the Future of Software Engineering*, pp. 35–46.

Paetsch, F., Eberlein, A., and Maurer, F. 2003. "Requirements engineering and agile software development." in Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on (pp. 308-313). IEEE.

ReCal3, 2017. "ReCal3: Reliability for 3+ Coders – Dfreelon.org." Accessed February 28, 2017. http://dfreelon.org/utils/recalfront/recal3/.

Ramesh, B., Cao, L., and Baskerville, R. 2010. "Agile requirements engineering practices and challenges: an empirical study." *Information Systems Journal* (*20*:5), pp. 449-480.

Rubinstein, D. 2007. "Standish group report: There's less development chaos today." *Software Development Times* March 1.

Stirna, J., and Kirikova, M. 2008. "How to Support Agile Development Projects with Enterprise Modelling." in *Information Systems Engineering: From Data Analysis to Process Networks*. New York: IGI Publishing, 2008. pp. 159-185.

Yourdon, E., and Argila, C. A. 1996. *Case Studies in Object-Oriented Analysis and Design*. Prentice Hall PTR.

VersionOne. 2016. *The 10th annual state of agile survey*. Technical report, Version One.

Appendix A: CompJournals Case Study

(Adapted from Yourdon and Argila, 1996)

SmallBytes is a software journal published by the CompJournals company. This company is planning to publish four new journals in the field of computer science and technology. CompJournals has asked you to build a web-based software system for efficiently managing the subscriptions, reviews, and publications of various journals. The web-based system to be developed should be made accessible by the editors, editorial assistants, authors, reviewers and subscribers of different journals according to their role. Analyze the existing business operations (given below for SmallBytes journal) and then prepare a list of user stories with prioritization for the proposed web-based system for CompJournals that can be used for the current and future journals.

SmallBytes is published on a monthly basis. A typical monthly issue consists of 5-10 articles, each written by one or more authors in the software engineering field. Though the authors receive no payment for their articles, they do receive a year's free subscription as a token of appreciation for their efforts. If they already have a subscription, then a new subscription (free) is made with the start date as day after the last date of current subscription.

SmallBytes has an editorial board of advisors. Each advisor on the editorial board normally serves for a one-year or two-year term, for which they receive a complimentary subscription to the magazine. As with most magazines, issues are scheduled and planned months in advance.

The editor does a quick review of the submissions to ensure that they fit with the magazine before assigning three or four advisors for detailed reviews and comments. Upon receiving replies from the advisors by email, the assistant records the review comments into the system. From time to time, upon receiving instructions from the editor, the assistant updates the list of advisors and their subject areas.

An editor manages SmallBytes with the support of an assistant who is responsible for keeping track of subscriptions and submissions using an in-house computerized system. The assistant receives subscription orders (new and renewal) from subscribers, and new article submissions by authors via mail or email, and registers those orders and article submissions into the system.

SmallBytes is sold on a subscription basis to its subscribers. Most subscriptions are for a one-year period, but the publisher accepts subscriptions for periods longer, or shorter than a year by simply pro-rating the annual subscription price. Payments for new subscriptions are normally received by cheque. Some subscribers pay by credit card, using a prescribed form that includes a signature, as the bank insists. This means that credit card orders are typically sent by fax or mail.

Appendix B: CompJournals Case Study - Expected Product Backlog

	As a/an	**I want to**	**so that**
1	Author	Submit an article for a journal	my article can be reviewed for publication
2	Author	Re-submit an article based on the comments from the editor and advisors	my revised articles can be published
3	Editor	Assign advisors to an article according to their field of interest	they can review the articles and provide comments
4	Editor	Reject an article without any further reviews	irrelevant articles need not be assigned for review
5	Editor	schedule the accepted articles for a forthcoming issue	forthcoming issue can be prepared
6	Editor	Accept an article for publication based on advisers' recommendations	authors can be informed and the article can be prepared for publishing
7	Advisor	Submit comments on an article assigned to me for review	the editor can assess the overall quality of the article
8	Advisor	Maintain my profile including personal details, subject areas of my interest	I get articles related to my interest for review
9	Assistant	Get the reminders for the expiring tenures of the advisors	so that their tenures can be extended or they can be replaced with new advisors
10	Assistant	Check the completeness of the articles submitted	the editor can decide review by advisors is necessary or not
11	Assistant	Add new advisors and their fields of interest	advisors can get the articles on the updated topics also
12	Publisher	Add a journal information including subscription price	I can set the price according to the period of subscription
13	Subscriber	Subscribe to the one or more journals	I can receive the journal issues