# Towards an Assessment Tool for Controlling Functional Changes in Scrum Process

Asma Sellami[1], Mariem Haoues[2], Nour Borchani[1], and Nadia Bouassida[1]

[1] Mir@cl Laboratory, University of Sfax, ISIMS, BP 242. 3021. Sfax-Tunisia.
asma.sellami@isims.usf.tn, borchani.nour@gmail.com,
nadia.bouassida@isims.usf.tn
[2] Mir@cl Laboratory, University of Sfax, FSEGS, BP 1088. 3018. Sfax-Tunisia
mariem.haoues@isims.usf.tn

**Abstract.** Unlike the old development process models, agile provides flexible methods adapted to requirements change. Although, agile methods such as Scrum are widely used today, an important number of agile projects end up in failure. This is due mainly to inaccurate estimates of time and lack of a structured change control process. However, tracking and controlling changes is important for a better project control. The main objective of this paper is to propose a tool for change controlling in scrum process. For this purpose, we use the COSMIC Functional Size Measurement method for a precise quantification and rapid evaluation of a change request. Reporting, tracking, and controlling the change status per project at different levels of details will certainly help in future projects management.

**Keywords:** Functional change, User requirements, Functional size measurement, User story description, COSMIC-ISO 19761, Scrum, User stories, Agile.

## 1 Introduction

Software projects are undoubtedly hard to manage. In fact, software products are more complicated compared to other products due to their invisibility, complexity, conformity, and changeability [11]. Hence, managers must ensure the balance and trade-offs among the scope, schedule, and budget while ultimately satisfying the customers needs.

Among the different Software Life-Cycle (SLC) models, managers select the appropriate one depending on the project nature and the team skills. At the beginning of the SLC, user requirements are often unclear, ambiguous, and not entirely defined. Hence, the risk of changing user requirements during the software development is highly predicted. However, the cost of a requirements change at an early phase of the SLC is relatively low compared to the cost of changing requirements at a later phase [11]. Hence, flexible models more adapted to the user requirements change become a necessity.

Unlike the oldest SLC models, agile methods such as scrum embrace requirements changes. In fact, they allow active collaboration between development teams and customers (*i.e.,* product owners). The Product Owner (PO) reports the requirements in an unstructured way when talking to the development team [18]. Although scrum is gaining popularity in comparison with other agile methods, 61% of the scrum projects fail [13]. This is due mainly to the lack of comprehensive documentations in scrum [12],

the limited use of standardized measures, and the poorly change control. To avoid the project failure, a well-defined change control process is required at any step of the scrum process, even within an ongoing sprint. In addition, implementing such process will improve the flexibility of scrum.

User requirements are classified into three categories: Functional User Requirements (FUR), Non-Functional Requirements (NFR), and Project Requirements and Constraints (PRC) [3]. FUR express *"what the software shall do in terms of tasks and services"* [3]. NFR include *"any requirement for a hardware/software system or for a software product, including how it should be developed and maintained, and how it should perform in operation"* [3]. PRC describe *"how a software system project should be managed and resourced or constraints that affect its performance"*. Depending on the changed requirement, user requirements changes are classified into: (i) functional changes, and (ii) technical changes [1]. In fact, a functional change affects the FUR. While, a technical change may affect the NFR or the PRC.

This paper proposes a tool for FC control in scrum process. This tool is based mainly on sizing FC using the COSMIC FSM method. The remaining of this paper is organized as follows: Section 2 presents firstly an overview of the scrum process, Functional Size Measurement (FSM) and COSMIC FSM method. Secondly, it discusses some related works. In section 3, we describe the change control process. In section 4, we illustrate our process through the case study "E-Commerce" and present our tool. Section 5 discusses the limitations and concerns of the change control process. Finally, section 6 concludes the presented work and outlines some of its possible extensions.

## 2 Background

### 2.1 Overview of the Scrum Process

Scrum process allows a better communication between the development team and the PO. For a successful scrum project, the development team must learn how to manage themselves efficiently. In addition, the PO must be actively involved in every single phase of the software development. Scrum appears to work better with small projects that require five to nine persons in a development team including designers, developers and testers. Nevertheless, some companies adapt scrum for large-scale projects [10].

The scrum process, illustrated in Fig. 1, starts with a high-level definition of the project scope. Scrum uses the product backlog as a list of stories created by the PO based on their initial requirements. These stories may increase or decrease in size based on decisions made throughout the software development. The list of stories is prioritized by the PO to be used as an iterative input for different sprints (Iterations) [15]. Thus, the active involvement of the PO is mandatory to explain, elucidate the next iteration that should be implemented and evaluate/test the work done.

For a single sprint, four types of meetings should be held: sprint planning meeting, daily stand up meeting, sprint retrospective meeting, and sprint review meeting. The stories to be implemented in a sprint are captured during the planning meeting. They are selected from the product backlog according to their priorities and placed in a sprint backlog. In practice, usually, only the first two or three sprints are identified

and planned. Daily stand up meetings are held during the sprint to discuss: what has been done, what are you going to do, and what are the issues [6]. Each sprint is ended by a sprint retrospective meeting, during which the team reviews the sprint and decides which change will be made, and how they can improve their work in the next sprint.

As we mentioned previously, scrum uses the user stories to represent the user requirements at different levels of details. A User Story (US) is a requirement written in a specific way illustrating the type of user, feature or functionality that the user want to do in order to realize some benefit [6]. Below the US description adapted in practice.

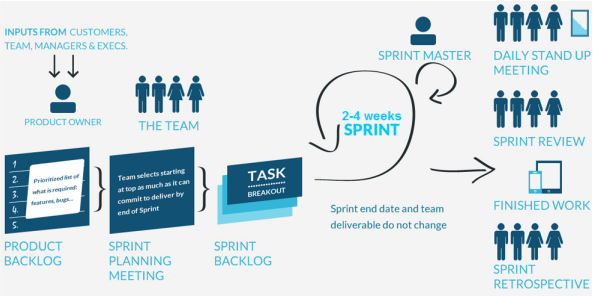


**Fig. 1.** Scrum software development process [2]

This description identifies **Who** will do the US or find it valuable <user type>, **What** it can be used for <feature or functionality>, and **Why** it is valuable or important <value or expected benefit>.

**As a** <user type>
**I want to** <feature or functionality>
**so that** <value or expected benefit>

Typically, development team uses the User Story Point (USP) to determine the effort required for the accomplishment of a US compared to other user stories in the same product backlog. Although its popularity, USP is not a good estimation technique. It has been widely criticized (*cf.,* [5], [9], [7], etc.). In fact, USP is only meaningful for a specific development team and project. Thus, it is necessary to use a standardized method that allows the measurement of the product functional size. In addition, the study in [7] proved that using COSMIC in agile projects gives a better results in estimating the effort needed to accomplish a US. For these reasons, we selected COSMIC ISO 19761.

### 2.2 Functional Size Measurement and COSMIC FSM Method: ISO 19761

Software size measurement throughout the SLC is used mainly for estimating the software development effort/cost and in driving decisions on the development project activities. The FSM methods measure the software size from the FUR. Functional Point Analysis (FPA) is the first FSM method proposed by Allan Albrecht in 1979. FPA

is supported by the "International Function Point Users Group" and ISO since 2003 (IFPUG-ISO 20926:2009). Thereafter, researchers proposed several methods to improve the original FPA method such as Nesma, MK II, FiSMA, and finally COSMIC.

COSMIC considers that a FUR involves a number of functional processes. Each Functional Process (FP) is detailed by a set of sub-processes of two types: data movement and data manipulation. A data movement moves a data group from/to a functional user (respectively Entry and eXit data movement) or from/to a persistent storage (respectively Read and Write data movement). Software size is measured by counting one CFP (COSMIC Function Point) for each data movement. The size of each FP is measured separately. The sizes of all functional processes are added to provide the software size.

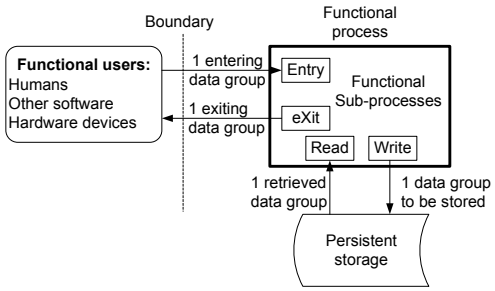


**Fig. 2.** COSMIC data movements [8]

COSMIC is the only FSM method that measure the size of a change to software. It defines a FC as *"any combination of additions of data movements or of modifications or deletions of existing data movements"* [8]. To measure the Functional Size of a FC, referred to as FS(FC), COSMIC attribute one CFP for each changed data movement regardless of the change type (addition, deletion, or modification). The FS(FC) is given by the aggregation of the sizes of all the added, deleted and modified data movements. The functional size of the software after a FC is given as the *sum* of all added data movements *minus* the functional size of all removed data movements [8].

### 2.3 Related Work

Given the importance of change management in the agile context, a number of research studies have addressed the requirements changes in scrum process. For instance, Lloyd *et al.,* addressed the problem of requirements changes during the development process in distributed agile development [14]. They proposed a supporting tool to help managing requirements changes in distributed agile development. On the other hand, Stålhane *et al.,* proposed to analyze the impact of technical changes [17]. In particular, this study focused on the safety requirements. Thus, two main questions have been addressed: (i) will the requirement and design affect the safety? and (ii) will the update affect the safety? Regarding the use of functionality measures in agile project, Commeyne *et al.,*

proved that the use of ISO standards to measure the size of agile projects is mandatory [7]. This study demonstrated the reliability of COSMIC in estimating the size and therefore the effort required to accomplish the defined requirements.

**Table 1.** Summary of the proposals focused on requirements change in scrum process

| Study | Focus | Type | Findings |
|---|---|---|---|
| Lloyd *et al.,* [14] | Requirements change management in distributed agile development | Experimental | A supporting tool |
| Commeyne *et al.,* [7] | Evaluation of teams' productivity using COSMIC | Experimental | COSMIC is more reliable in estimating models with much smaller variances |
| Stålhane *et al.,* [17] | Impact of technical changes in safety requirements | Exploratory | A supporting tool that ensures the validity of safety |

Table 1 summarizes the main proposals that focused on the requirements changes in scrum process. We noticed that some studies focused on functional changes (*cf.,* [14]) while other studies focused on technical changes (*cf.,* [17]). However, changes in these papers have been always considered as new requirements. In addition, none of the previous studies used a change control process. Thus, no changes evaluation is provided. However, it is important to evaluate requirements' changes and provide useful information for the right audience. This will certainly help during the software maintenance as well as for new software development. Moreover, change evaluation is usually based on experts' judgment. Whereas, experts' judgment evaluation is less transparent compared to any other techniques and depends mainly on the experts' skills.

## 3 Change Control Process

### 3.1 Detailed User Story Description

In scrum, there is no standard US representation. Thus, different templates have been proposed mainly to describe what the users will need the software for. In addition, user stories are used at a high level of details [9]. This will impact the size measurement. In order to guarantee the quality of measurement, we propose a detailed description of US that represents all the information to apply COSMIC (see Fig. 3). Where:

- <UserType> is the user of the US referred to as the functional user in COSMIC.
- <Action> and <Object> are used to replace the concept "feature or functionality" in the US description provided in section 2.1. In fact, the "feature or functionality" is a combination of an action and an object that the action will be applied on.
- <value or expected benefit>: It is used to characterize the successful ending of US.
- <NFR> describes the non-functional requirements.
- <Attachments> any attachment that help defining the user story such as GUI.

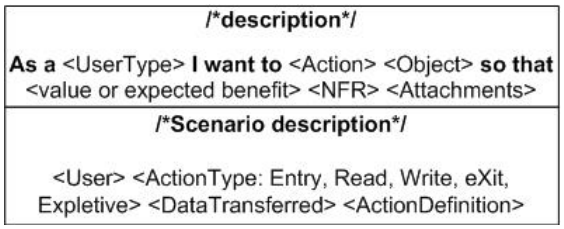


**Fig. 3.** Detailed user story description format

The <value or expected benefit>, <NFR>, and <Attachments> are optional.

In general, a US could have finer or coarser granularity than functional processes. That is a US could be a fraction of a FP or a set of functional processes. In our study, we consider that each US is associated to a FP. Thus, two user stories could not have the same [<UserType>, <Action>, <Object>, and <value or expected benefit>]. The US description in Fig. 3 provides more details in comparison with the old one. But, it does not represent the functional sub-process. Hence, moving to the scenario description is required to apply COSMIC. At this level, we distinguish the following concepts:

- <User>: External actor could be a human actor (*e.g.,* moderator, customer, etc.) or an external system in a direct relation with the software to be measured.
- <ActionType>: the action that will be applied on a data group is restricted to a number of verbs (*e.g.,* select, read, etc). These verbs are classified into four corpses: entry actions, read actions, write actions, and exit actions (see Appendix). These actions represent the sub-process in each US. Thus, a sub-process can be a data movement or a data manipulation. This depends on whether or not it transfer data.
- <DataTransfered>: represents the data that have been transferred in each sub-process. <DataTransfered> in COSMIC corresponds to the "data group" concept.
- <ActionDefinition>: gives a summary of the user story purpose.

### 3.2 Prioritizing User Stories

Algorithm 1 is used to help prioritizing user stories in the product/sprint backlog. Hence, it can be used when selecting user stories from the product backlog and when re-organizing user stories in an on-going sprint after a change.

In scrum, user stories are prioritized as requested by the PO [4]. However, the PO may not have enough knowledge about the implementation details. Hence, ordering user stories based only on priority is not sufficient. In fact, the developer's view is also important in the user stories prioritization. Taking into account the developer's perspective is important to maximize the business value released at the end of every sprint. Therefore, we propose to balance the US for PO and development team perspectives according to mainly three parameters: importance, priority, and functional size. The priority of user stories is defined by the PO (*i.e.,* P1 is more prior than P2, P2 is more prior than P3, etc.). The importance of a US can be Essential or Desirable. User stories in the same cluster of classes (the same data base, service, etc.) have the same importance. The functional size is measured using COSMIC.

**Algorithm 1:** Prioritizing user stories

**Aim** : Prioritizing user stories
**Require:** **P(US)** the priority of user story (US).
**Imp(US)** the importance of user story (US).
**FS(US)** the functional size of user story (US).
**Ensure** : User stories organized by taking into account their priorities and importances first of all and then their functional sizes.

```
begin
    if Imp(USi) = Imp(USj) & P(USi) != P(USj) then
        Select the more prior user story ;
    else if Imp(USi) != Imp(USj) & P(USi) != P(USj) || P(USi) = P(USj) then
        Select the most important (Essential) user story ;
    else if Imp(USi) = Imp(USj) & P(USi) = P(USj) then
        Select the user story with minimum size ;
    else if Imp(USi) = Imp(USj) & P(USi) = P(USj) & FS(USi) = FS(USj) then
        Select the user story that requires less demand on resources (time or budget) ;
```

On the other hand, developers identify the status of a US that can be used to control the development progress. Thus, the status of a US[3], as shown in Fig. 4, can be:

- **New** is the status of a US in the product backlog.
- **To do** is the status of a US assigned to an on-going sprint.
- **In Progress** is the status of a US currently being implemented.
- **To Verify** is the status of a US ready for testing.
- **Done** is the status of a US tested with success in the customer environment.

As it is described in [16], we kept only the "Done" and "In Progress" status.


**Fig. 4.** User stories status in scrum process

### 3.3 Measuring Software Functional Size from User Stories Description

This section proposes a set of formulas that can be used to measure the software size based on the US description. It must be noted that the FS of the product backlog can be

[3] https://www.dreamstime.com/stock-illustration-scrum-task-kanban-board-sticky-notes-whiteboard-post-agile-software-development-hanging-tasks-team-image91765825

different from the FS of the increment product. In fact, changes always happen. Hence, new functionality may appear, while others may be modified or deleted. The FS of the product backlog is given by measuring the sizes of all sprints initially identified. While, the FS of the increment product depends on the sizes of all the implemented sprints.

The functional size of the product backlog or the increment product is equal to the sum of the functional sizes of *all* the sprints it includes (see Equation 1).

$$FS(P) = \sum_{i=1}^{n} FS(S_i) \tag{1}$$

Where:

- FS(P) is the functional size of the product backlog or the increment product.
- FS($S_i$) is the functional size of sprint $i$.
- $n$ is the number of sprints initially identified in the case of product backlog size measurement or the number of implemented sprints in the case of increment product size measurement.

The functional size of a sprint is equal to the sum of the functional sizes of *all* the user stories it includes (see Equation 2).

$$FS(S_i) = \sum_{j=1}^{m} FS(US_{ij}) \tag{2}$$

Where:

- FS($S_i$) is the functional size of sprint $i$ ($1 \leq \mathrm{i} \leq n$).
- FS($US_{ij}$) is the functional size of the user story $j$ in $S_i$.
- $m$ is the number of user stories in sprint $S_i$.

The functional size of a user story is equal to the sum of the functional sizes of its actions (see Equation 3).

$$FS(US_{ij}) = \sum_{k=1}^{p} FS(Act_{ijk}) \tag{3}$$

Where:

- FS($US_{ij}$) is the functional size of the user story $j$ in $S_i$.
- FS($Act_{ijk}$) is the functional size of action $Act_{ijk}$ in $US_{ij}$ ($1 \leq \mathrm{i} \leq n$ and $1 \leq \mathrm{j} \leq m$).
- $p$ is the number of actions in user story $j$.

### 3.4 Steps for the Change Control Process

To avoid any trouble in the project progress, following a control change process is important. Fig. 5 illustrates our change control process. It starts when a change request is submitted. A change request is documented in order to give a detailed description of the change. A well change documentation must include the following items:

– **Statement of the need:** identify clearly the changed item that must be analyzed.
– **Reasons for the change:** describe the causes for changing user requirements.
– **Conditions of Success:** the requester define what he expects from the change.

The documented change is submitted to the project team. The change revision with the team is required especially for complex change. In our previous work [16], we proposed an approach for evaluating FC request. We also proposed a set of algorithms to decide about a change request (accept, defer, or deny). These decisions are made based on the FS(FC) and the development progress. If a change is deferred or denied, the development team must communicate the decision to the change requester. The communication is needed in order to discuss the customer expectation, and the change values and cost. The development team must explain to the PO how long the change is going to take, and its impact on the project progress. After making the decision, it is required to update the documents to track all changes that are to be implemented. The change request is then implemented. It is send later for revision and testing.

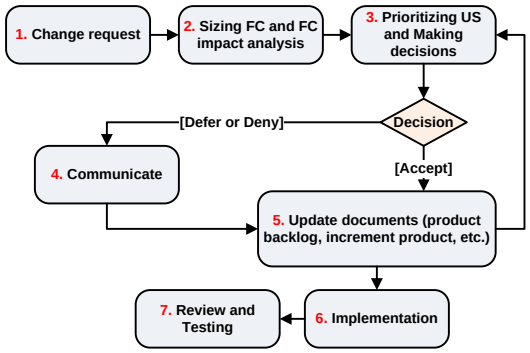


**Fig. 5.** Change control process for product team

## 4 Case Study and Tool

### 4.1 Case Study

The "E-commerce" is a web site developed by a team of engineering students. The web site allows a customer to buy computer equipment on-line. It includes initially ten user stories. After discussing their priority and measuring their sizes, these user stories are initially classified into three sprint: S1(US5, US6, and US7), S2(US4, US8, US9, and US10), and S3(US1, US2, and US3). The initial measurement results are given in Table 2. By applying our formulas in section 3.3, the total size of "E-commerce" is equal to:

FS("E-commerce") = FS(S1) + FS(S2) + FS(S3), where
FS(S1) = FS(US5) + FS(US6) + FS(US7) = 4 CFP + 6 CFP + 6 CFP = 16 CFP

FS(S2) = FS(US4) + FS(US8) + FS(US9) + FS(US10)
= 3 CFP + 3 CFP + 3 CFP + 6 CFP = 15 CFP
FS(S3) = FS(US1) + FS(US2) + FS(US3)
= 3 CFP + 3 CFP + 3 CFP = 9 CFP
Hence, the FS("E-commerce") is equal to 16 CFP + 15 CFP + 9 CFP = 40 CFP.

**Table 2.** Detailed Product Backlog

| **User Stories** | **Description** | **FS(US)** | **Status** | **Priority** | **Importance** |
|---|---|---|---|---|---|
| US 1 | **As a** visitor **I want to** consult the list of articles | 3 CFP | New | P2 | Essential |
| US 2 | **As a** visitor **I want to** consult the details of an article | 3 CFP | New | P2 | Desirable |
| US 3 | **As a** visitor **I want to** search for an article by its name | 3 CFP | New | P2 | Desirable |
| US 4 | **As a** client **I want to** command an article | 3 CFP | New | P3 | Essential |
| US 5 | **As an** administrator **I want to** add a new article | 4 CFP | New | P1 | Essential |
| US 6 | **As an** administrator **I want to** update the details of an article | 6 CFP | New | P1 | Desirable |
| US 7 | **As an** administrator **I want to** delete an article | 6 CFP | New | P1 | Desirable |
| US 8 | **As an** administrator **I want to** consult the list of commands | 3 CFP | New | P3 | Essential |
| US 9 | **As an** administrator **I want to** consult the details of a command | 3 CFP | New | P3 | Desirable |
| US 10 | **As an** administrator **I want to** update the information of a command | 6 CFP | New | P3 | Desirable |

Applying Algorithm 1, US5, US6, and US7 have been selected in sprint S1. Table 4 provides the detailed scenarios of these three user stories. As planned, the development team started by S1. The selected three user stories have been implemented in two weeks. They moved then to sprint S3. US1, US2, and US3 have been selected to be implemented in this sprint. As mentioned in Table 2, the priority of these user stories is equal to P2. Whereas, the importance of US1 is Essential and that of US2 and US3 is Desirable. Hence, according to Algorithm 1, the development team starts by US1.

By the end of implementing US1, the PO proposes to add US11, US12, and US13 (see Table 3). The priority for all the added user stories is P2 and their importance is Essential. The status of user stories initially in S3 are: US1 status = done and US2 status = US3 status = In Progress. The question here is whether to accept the FC and implement it in S3 or defer the FC to the next sprint S2. In this case, the FC affects an ongoing sprint and proposes the addition of three user stories. The total FS of undone user stories is equal to 6 CFP. While, the FS(FC) is equal to 10 CFP. We classify the user stories (initially in S3 and new ones) using Algorithm 1. For this purpose, we compare

the importance of the new user stories (US11, US12 and US13) with the user stories in S3 with status = in progress (US2 and US3). Hence, the user stories will be organized as follow: US12, US13, US11, US2, and US3. The user stories coming from the FC appear in the beginning of the list. Thus, by applying the Algorithm for FC decision making in an on-going sprint in [16], we make the following recommendations.

- Accept US11, US12, and US13.
- Defer US2 and US3 to the next sprint S2.

**Table 3.** Functional change details

| **US** | **User story description** | **FS(US)** | **Status** | **Priority** | **Importance** |
|---|---|---|---|---|---|
| US 11 | **As a** visitor **I want to** create an account | 4CFP | New | P2 | Essential |
| US 12 | **As a** customer **I want to** logon | 3 CFP | New | P2 | Essential |
| US 13 | **As an** administrator **I want to** logon | 3 CFP | New | P2 | Essential |

In practice, after all the changes proposed during its implementation, this product has been delivered after six sprints each one lasts for two weeks.

**Table 4.** Detailed scenarios of US5, US6 and US7 in S1

| **User Stories** | **User** | **Action** | **Data transferred** | **Status** | **Data Movement** | **CFP** |
|---|---|---|---|---|---|---|
| US 5 | Administrator | Enter | Article Data | To Do | Entry | 1 |
| | | Verify | Article Data | | Read | 1 |
| | | Add | Article Data | | Write | 1 |
| | Administrator | Receive | E /C message | | eXit | 1 |
| | | | | | | **FS(US5) = 4 CFP** |
| US 6 | Administrator | Select | Article ID | To Do | Entry | 1 |
| | | Read | Article ID | | Read | 1 |
| | Administrator | Receive | Article Data | | eXit | 1 |
| | Administrator | Enter | Article Data | | Entry | 1 |
| | | Save | Article Data | | Write | 1 |
| | Administrator | Receive | E /C message | | eXit | 1 |
| | | | | | | **FS(US5) = 6 CFP** |
| US 7 | Administrator | Select | Article ID | To Do | Entry | 1 |
| | | Search | Article ID | | Read | 1 |
| | Administrator | Receive | Article Data | | eXit | 1 |
| | Administrator | Select | Article ID | | Entry | 1 |
| | | Delete | Article ID | | Write | 1 |
| | Administrator | Receive | E/C message | | eXit | 1 |
| | | | | | | **FS(US5) = 6 CFP** |

### 4.2 Tool

Fig. 6 gives some snapshots of the proposed tool in this paper. Interface 1 is used to add a new US with respect to the US description proposed in section 3.1. All the added user stories are placed in the product backlog as provided in interface 2. This interface represents the product backlog, sprint backlog and some additional details concerning the project progress such as: done user stories (in number and CFP), implemented sprint, etc. As it is provided in this interface, three user stories are selected in S1. Interface 3 provides the details of an ongoing sprint such as: starts and finish dates, percentage of done user stories, etc. Interface 4 is used to update user story to respond to a change request. Finally, interface 5 is used to make recommendations after a FC request and a better change control.

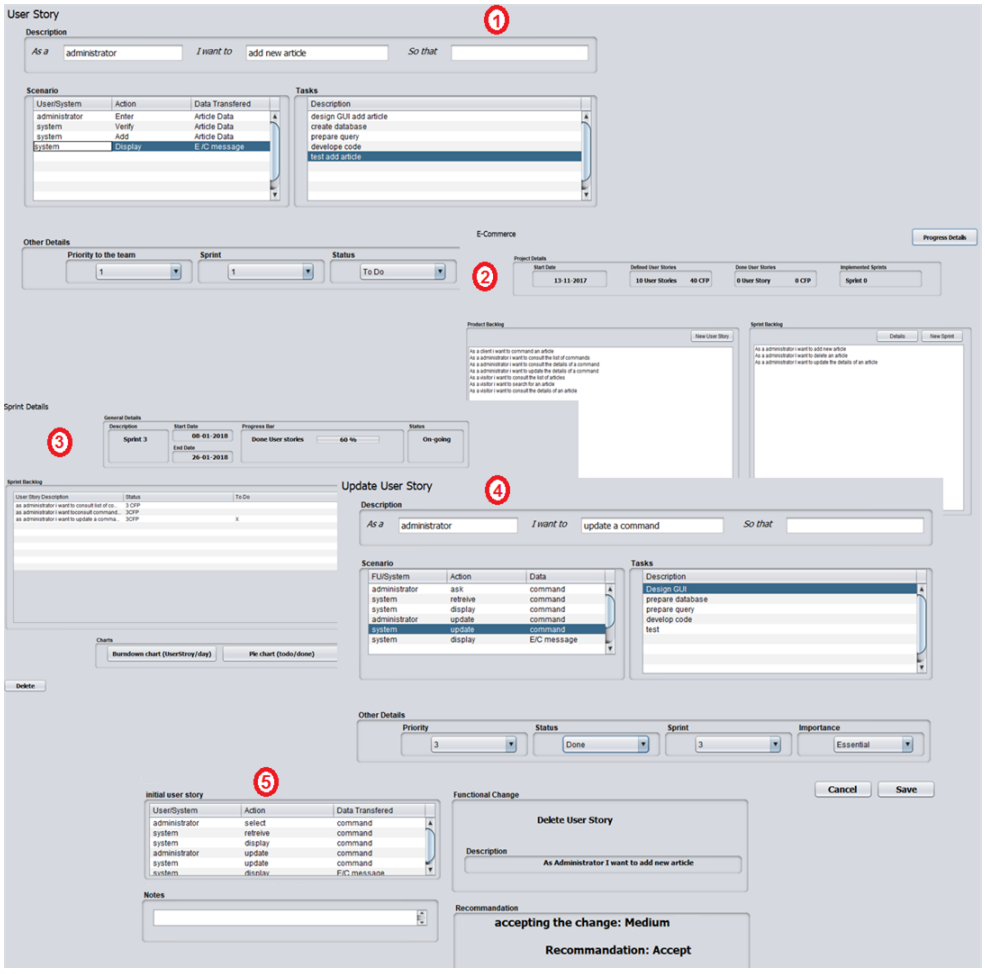


**Fig. 6.** Snapshots of tool with the case study "E-commerce"

## 5 Discussion and Limitations

User requirements are the basis of software project. In agile, they are described by user stories. The US description is a valuable document for the development team. Hence, if user stories are poorly described, the software development/maintenance will be critically vulnerable. In fact, well detailed user stories are not always available in practice since they require much more time. However, the more time spent at the beginning of the process, the little time will be required later. Consequently, a detailed description of a US is needed. So that any change can be carefully monitored and reported.

On the other hand, COSMIC is a flexible method that readily fits into the scrum process. We proved in this study that COSMIC can be applied throughout the process even within an ongoing sprint. Functional changes can be requested at any time and evaluated any where for better decisions-making. Thus, COSMIC provides greater flexibility to scrum process without disrupt its structure. In addition, the proposed process can be extended for the hidden requirement and high level detailed requirement by using approximate measurement approaches. In the case of a technical change (*i.e.,* a change is the NFR) that requires the implementation of additional FUR, the change is translated into functional change. Hence, it can be measured and evaluated using COSMIC.

The tool has been tested using one case study "E-Commerce". In order to ensure the efficiency of our change control process, testing the tool and algorithms in an industrial environment is required. Feedback from practitioners is necessary to improve the tool.

## 6 Conclusion and Future Work

In practice, changes in user requirements are inevitable and present a main issue. Any deviation from user requirements may lead to project failure or induce an extra effort and much time through the software life cycle to satisfy a change request.

This paper proposed an assessment tool that support a FC control throughout the scrum process. Our tool is based on a detailed description of user stories and their sizing with COSMIC. In fact, the evaluation of a change request and the decision made responding to the change are based mainly on its functional size and its impact on the development progress. This avoid any misunderstanding due to the subjective evaluation of the change (done mostly by developers). Hence, the proposed tool adds objectivity to the scrum process. It can be used by decision-makers to meet customer's needs, identify problems in future projects, and estimating future software project effort.

For further work, we consider that approximate/rapid change evaluation is required especially for an urgent change request. In addition, we consider that it is required to use the proposed tool in real industrial environment.

## Appendix

**Entry corpus:** Assign, change, choose, click, create, edit, give, input, modify, provide, re-enter, select, submit, type, update.
**Exit corpus:** display, edit, list, output, post, present, print, return, send, Show update, view.

**Read corpus:** find, get, obtain, post, read, recognize retrieve, Validate, Verify.
**Write corpus:** add, archive, change, create, define, delete, edit, insert, record, register, remove, save, store, Update.

## References


1. ISO/IEC 14143-1: Information Technology - Software Measurement - Functional Size Measurement. Part 1: Definition of Concepts (2007)
2. Scrum software development process (2018), https://www.maxxor.com/
3. Abran A., Desharnais J.M., K.B.R.D.S.C.W.S.B.D.F.P.L.A.S.L.V.F.B.C.G.C.M.L.S.H., C., W.: Guideline on Non-Functional & Project Requirements: How to consider non-functional and project requirements in software project performance measurement, benchmarking and estimating (2015)
4. Ambler, S.W.: User Stories: An Agile Introduction (2014)
5. Berardi E., Buglione L., S.L.S.C.T.S.: Guideline for the use of cosmic fsm to manage agile projects, v1.0 (2011)
6. Cohn, M.: User Stories Applied: For Agile Software Development. Addison-Wesley Professional (2004)
7. Commeyne, C., Abran, A., Djouab, R.: Effort Estimation with Story Points and COSMIC Function Points: An Industry Case Study (2016)
8. COSMIC: The COSMIC Functional Size Measurement Method, Version 4.0.2, Measurement Manual (October 2017)
9. Desharnais, J.M., Kocaturk, B., Abran, A.: Using the cosmic method to evaluate the quality of the documentation of agile user stories. In: 2011 Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement. pp. 269–272 (Nov 2011)
10. Dikert, K., Paasivaara, M., Lassenius, C.: Challenges and success factors for large-scale agile transformations. Journal of Systems and Software **119**(C), 87–108 (Sep 2016)
11. Fairley, R.E.: Managing and Leading Software Projects. Wiley-IEEE Computer Society Pr (2009)
12. Furtado, F., Zisman, A.: Trace++: A traceability approach to support transitioning to agile software engineering. In: The 24th International Requirements Engineering Conference (RE). pp. 66–75 (Sept 2016)
13. Gilb, T.: Why agile product development systematically fails, and what to do about it! (2018), https://www.gilb.com/blog/why-agile-product-development-fails-and-what-to-do-about-it
14. Lloyd, D., Moawad, R., Kadry, M.: A supporting tool for requirements change management in distributed agile development. Future Computing and Informatics Journal **2**(1), 1–9 (2017)
15. Schwaber, K.: Agile Project Management with Scrum (Developer Best Practices). Microsoft Press; 1 edition (2004)
16. Sellami, A., Haoues, M., Borchani, N., Bouassida, N.: Orchestrating functional change decisions in scrum process using cosmic fsm method. In: the 13th International Conference on Software Technologies (in press). ICSOFT '18 (2018)
17. Stålhane, T., Hanssen, G.K., Myklebust, T., Haugset, B.: Agile change impact analysis of safety critical software. In: Bondavalli, A., Ceccarelli, A., Ortmeier, F. (eds.) Computer Safety, Reliability, and Security. pp. 444–454 (2014)
18. Taibi, D., Lenarduzzi, V., Ahmad, M.O., Liukkunen, K.: Comparing communication effort within the scrum, scrum with kanban, xp, and banana development processes. In: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering. EASE'17 (2017)