A Measures-Driven Decision Support System for Managing Requirement Change in Scrum: An Empirical Evaluation

Hela Hakim[1], Asma Sellami[2] and Hanêne Ben-Abdallah[3]
[1]*Mir@cl Laboratory, University of Sfax, FSEGS, BP 1088. 3018, Sfax, Tunisia*
[2]*Mir@cl Laboratory, University of Sfax, ISIMS, BP 242. 3021, Sfax, Tunisia*
[3]*Higher Colleges of Technology, Dubai, U.A.E.*

Keywords: Managing Change, Decision Support System DSS, Functional Change FC, Structural Change SC, Functional Size, Structural Size, COSMIC FSM Method, Structural Size Measurement SSM Method, Scrum.

Abstract: In Scrum-based projects, precise assessments of requirement changes are crucial for effective management. A Decision Support System (DSS) can streamline managing these changes, improve collaboration, and enhance decision-making. This paper proposes a Measure-Driven Decision Support System (MD-DSS) for managing requirement changes at both functional and structural levels, using COSMIC FSM (ISO 19761) and an extended Structural Size Measurement method. The MD-DSS benefits all Scrum stakeholders, including Product Owners, Scrum Masters, Development Teams, and managers. Its performance was evaluated quantitatively and qualitatively across 15 software development projects.

1 INTRODUCTION

Managing software projects is challenging due to their complexity and frequent changes (Fairley, 2009). Effective change management is essential for balancing budget, timeline, and scope. Early requirements are often unclear, leading to frequent and cheaper changes in early stages (Bano et al., 2012). Agile methods like Scrum are favored for their adaptability (Dikert et al., 2016), but they face a 61% failure rate due to poor documentation and change control (Gilb, 2018).

Requirements are categorized into Functional User Requirements (FUR), Non-Functional Requirements (NFR), and Project Requirements and Constraints (PRC) (Abran, 2015). Change requests can be functional or technical (ISO, 2007). Scrum teams usually rely on expert judgment for changes, but this is not always effective (Abran, 2015).

This paper proposes a Measures-Driven Decision Support System (MD-DSS) to manage requirement changes in Scrum, improving on Hakim et al.'s (2020) work. The MD-DSS helps measure, prioritize, and evaluate changes at functional and structural levels.

The rest of this paper is organized as follows. Section 2 provides an overview of the COSMIC FSM method, the Structural Size Measurement SSM method and the SCRUM process. Section 3 discusses the related work. Section 4 provides our proposed measures-driven Decision Support System used for managing requirements changes at functional and structural levels of requirement change. Section 5 discusses the evaluation of the Decision Support System. Finally, section 6 summarizes the presented work and outlines some of its possible extensions.

2 BACKGROUND

This section describes an overview of the COSMIC FSM method, the SSM method and finally the scrum process.

2.1 COSMIC FSM Method

The Common Software Measurement International Consortium (COSMIC) represents an internationally recognized Functional Size Measurement (FSM) method. It is intentionally crafted to remain neutral towards any particular implementation choices embedded within the operational artifacts of the software under assessment. The COSMIC sizing process for evaluating the functional requirements magnitude of software encompasses three distinct phases: the measurement strategy phase, the mapping phase, and the measurement phase (as per COSMIC

Hakim, H., Sellami, A. and Ben-Abdallah, H.
A Measures-Driven Decision Support System for Managing Requirement Change in Scrum: An Empirical Evaluation.
DOI: 10.5220/0012807700003753

In *Proceedings of the 19th International Conference on Software Technologies (ICSOFT 2024)*, pages 161-168
ISBN: 978-989-758-706-1; ISSN: 2184-2833

v5.0, 2021). This approach hinges on the quantification of functional processes (FP), each comprising a collection of functional sub-processes that can be categorized into either data movement or data manipulation. Specifically, these movements are classified into four types: Entry (E), Exit (X), Read (R), and Write (W).

A data group is a set of attributes that describes one object of interest. The COSMIC measurement unit is one data movement of one data group indicated as one CFP (COSMIC Function Point). The size of a functional process is determined by the sum of the data movements it includes.

According to COSMIC (v5.0, 2021), a functional change is described as "any amalgamation of additions, modifications, or deletions of existing data movements." In a functional process, the magnitude of a functional change is determined by the total of its added, deleted, and modified data movements. Subsequently, the functional size of the software following the change is calculated as the cumulative size of all added data movements minus the size of all removed data movements.

2.2 The Structural Size Measurement SSM Method

An extension of the COSMIC Functional Size Measurement (FSM) method, the Structural Size Measurement (SSM) method serves as a vital tool for addressing the need for more detailed measurements to quantify data manipulation within software products. Mirroring the approach of COSMIC, the SSM measurement process comprises three distinct phases: the Measurement Strategy Phase, Mapping Phase, and Measurement Phase. Proposed by Sellami (Sellami et al., 2015), the SSM method is tailored for UML sequence diagrams, developed in accordance with the measurement process advocated by Abran (Abran, 2010). The Structural Size Measurement method is applied to the combined fragments of a sequence diagram to gauge its Structural Size (SS). This SS, also referred to as control structural size, encompasses the structural size of both Conditional Control Structures (CCS) and Iterative Control Structures (ICS), as depicted through constructs like alt, opt, and loop. The SS of a sequence diagram is defined at a granular level, specifically reflecting the size of its control structures' flow graph.The use of SS requires the identification of two types of data manipulations depending on the structure type: CCS, ICS.

Each data manipulation is equivalent to one CSM (Control Structure Manipulation) unit. The sequence structural size is computed by adding all data manipulations identified for every flow graph.

The SSM The Structural Size Measurement (SSM) method defines a Structural change as "any combination of additions, modifications, or deletions of existing data manipulation" (Hakim et al., 2017). Within a functional process, incorporating its structural aspect, the magnitude of a Structural change is determined by the total of its added, deleted, and modified data manipulations. Consequently, the software's structural size following the change is calculated as the cumulative size of all added data manipulations minus the size of all removed data manipulations.

2.3 Overview of the Scrum Process

The Scrum process is a collaborative method for managing software projects, focusing on adaptability, transparency, and iterative progress. Key roles include the Product Owner, Scrum Master, and Development Team. Scrum uses artifacts like the Product Backlog and Sprint Backlog to organize work. Structured events, including Sprint Planning, daily Scrum meetings, Sprint Reviews, and Retrospectives, help guide the process. These principles enable teams to deliver value incrementally and respond to changes effectively.

3 RELATED WORK

Agile development effectively handles requirements changes(Abran, 2015), especially in Scrum. Key studies have addressed this:

Drury-Grogan and O'Dwyer (Drury-Grogan et al., 2013) identified factors influencing decision-making in Scrum, such as sprint duration, experience, and resource availability. Decisions often rely on subjective expert judgment, which lacks transparency. To improve objectivity, COSMIC FSM and SSM methods are recommended for accurate change evaluation.

Commeyne et al. (Commeyne et al, 2016) validated using ISO standards to measure agile project size, demonstrating COSMIC's reliability. Lloyd et al. (Lloyd et al., 2017) proposed a tool for managing changes in distributed agile development, and Stålhane et al. (Stålhane et al., 2014) analyzed the impact of technical change requests on safety requirements. Sellami et al. (Sellami et al., 2018)) developed a COSMIC-based tool to evaluate functional changes in Scrum. Hakim et al. (Hakim et al., 2020) proposed a detailed Requirements Change

Evaluation Process considering functional and structural levels for better decision-making.

While Scrum teams typically avoid mid-iteration changes to prevent defects, some necessary changes should be prioritized. This paper suggests a Decision Support System for managing requirement changes in Scrum, evaluated through 15 software development projects

4 MEASURES-DRIVEN DECISION SUPPORT SYSTEM FOR MANAGING REQUIREMENTS CHANGES

This section presents the steps describing our proposed MD-DSS for managing change in SCRUM. The MD-DSS proposed in the herein work is composed of three main parts.

(1)-Change Request Classification

(2)-Requirement Change impact analyses at functional and structural level

(3)-Prioritizing Change and making-Decisions

In Scrum, change requests are made by the product owner or development team and must be articulated as a user story (USc). An impact analysis is then conducted. If the change is in an ongoing sprint, details like sprint size and start date are noted, and the functional and structural sizes of the changed components and all incomplete user stories in the sprint are measured. For implemented sprints, only the sizes of the changed components and user stories are measured. These measurements help evaluate the change and guide decisions to accept, deny, or defer the request.

4.1 Change Requirement Request Classification

In the context of software development or project management, "Change Requirement Request classification" refers to the systematic categorization of change requests based on various criteria such as their nature, impact, urgency, and priority. This process helps project teams and stakeholders better understand and manage change requests by organizing them into meaningful groups and facilitating decision-making and prioritization.

In this research work, we are interested to classify change request based on the nature of Change. We distinct between the functional change request and the structural change request.

4.1.1 Functional Change Request

A functional change request in the context of Scrum refers to a request for modifying or adding new functionality to the product being developed. Based on the refined US templates proposed by (Hakim et al., 2020) that support the COSMIC ISO measurement.

4.1.2 Structural Change Request

A Structural change request in the context of Scrum refers to a request for modifying or adding new structural aspects to the product being developed. The Structural aspect take into account the structure of the option scenario, alternative scenario (alternative scenario 1, 2) and the iterative scenario of a feature or functionality. Based on the refined US templates proposed by (Hakim et al., 2020) that support the SSM measurement.

4.2 Requirement Change Impact Analyses at Functional and Structural Level

4.2.1 Measuring of Requirement Change Request at Functional and Structural Level

Once requirements or changes are classified and described in the user story (US) format, software size measurements can be applied using measurement formulas based on this refined format. These formulas facilitate determining the functional and structural sizes of requirements. Requirements size derived from the product backlog differs from that derived from the increment product due to changes during the Scrum process. New functionalities may emerge, while others may be modified or deleted. The COSMIC functional size measurement method and the structural size method are used to determine the functional and structural sizes, respectively. The size of the product backlog is the sum of the sizes of all sprints it includes. (see Equation 1 and Equation 2).

$$SS(P) = \sum_{i=1}^{n} SS(S_i) \quad (1)$$

$$FS(P) = \sum_{i=1}^{n} [\![FS(S_i)]\!] \quad (2)$$

where -FS(P)is the functional size of the product backlog or the increment product.-SS(P)is the structural size of the product backlog or the increment product.-FS(Si) is the functional size of sprinti.-SS(Si) is the structural size of sprinti.-n is the number of sprints initially identified in the case when sizing the product backlog or the number of implemented sprints in the case when sizing the increment product.

The functional size, respectively, the structural size of a sprint is the sum of all the functional sizes, respectively, the structural sizes of all the user stories (US) it includes (see Equation3 and Equation4).

$$FS(S_i) = \sum_{j=1}^{m} FS(US_{ij}) \quad (3)$$

$$SS(S_i) = \sum_{j=1}^{m} SS(US_{ij}) \quad (4)$$

where-FS(Si) is the functional size of sprinti($1 \le i \le n$). -SS(Si)is the structural size of sprinti($1 \le i \le n$).-FS(USij)is the functional size of the USj in Si.-SS(USij)is the structural size of the USj in Si.
-m is the number of user stories in sprint Si.Note that FS(USij)is the sum of all the functional sizes of its actions(see Equation 5).The SS(USij) is the sum of all the Structural sizes of its alternatives (conditional and iterative) (see Equation 6)

$$FS(US_{ij}) = \sum_{k=1}^{p} FS(Act_{ijk}) \quad (5)$$

$$SS(US_{ij}) = \sum_{k=1}^{r} SS(Alt_{ijk}) \quad (6)$$

where:FS(US_{ij}) is the functional size of the US_jin S_i. -SS(US_{ij}) is the structural size of the US_jin S_i.-FS(Act_{ijk})is the functional size of action- Act_{ijk} in US_{ij}($1 \le i \le n$and$1 \le j \le m$).-SS(Alt_{ijk}) is the structural size of alternative Alt_{ijk}in US_{ij}($1 \le i \le n$and$1 \le j \le m$).-*p* is the number of actions in user story*j*.-r is the number of Alternatives in user story *j*.

4.2.2 Evaluating of Requirement Change Request at Functional and Structural Level

Table 1: Evaluating a FC request when USc status = undone/done (Hakim et al, 2020).

Low	Moderate	High
FS(FC)=1CFP	2CFP≤FS(FC)≤FS(USundone/USdone)	FS(FC)>FS(USundone/USdone)

Table 2: Evaluating a SC request when USc status = undone/done. (Hakim et al, 2020).

Low	Moderate	High
SS(SC)=1CSM	2CSM≤SS(SC)≤SS(USundone/USdone)	SS(SC)>SS(USundone/USdone)

Tables 1 and 2 present respectively the Evaluating a aFC request when USc status = undone/done respectively the Evaluating of a SC request when USc status = undone/done

4.3 Prioritizing Change and Making-Decision

Ensuring alignment between user stories (US) and product owner expectations is crucial for software project success. While Scrum prioritizes user stories based on the product owner's preferences, this may overlook implementation details. Incorporating developer insights into prioritization is vital for optimizing business value. A holistic approach integrates perspectives from both product owners and development teams, considering importance, priority, and functional and structural sizes. User story priority, determined by the product owner, uses higher numerical values for greater priority. User story importance, categorized as Essential or Desirable, is assessed by development teams. Functional size is measured using COSMIC FSM, and structural size through a structured method.

4.3.1 Prioritizing Change

Algorithm 1 presents how to prioritize the requirements changes using both the COSMIC functional and structural size measurement methods. Each Functional change is evaluated using the COSMIC FSM method, while each Structural change is evaluated using SSM method.

Four basic values (Priority, Importance, CFP, and CSM) are used for running 'prioritizing user stories' algorithm, and therefore implementing the decision-making.

Aim: Prioritizing user stories taking into account the following inputs: P(US), I(US), FS(US),and SS(US)
Inputs: P(US): The Priority of a User Story (US); I(US): The Importance of a US;FS(US): The Functional Size of a US;SS(US): The Structural Size of a US.
Outputs: User stories are organized by taking into account their priorities, importance, and their functional and structural sizes (Hakim et al, 2020).

If *P(USi) != P(USj)* **then**
Select the more prior user story (US);

Else if *P(USi) == P(USj) & I(USi) != I(USj)* **then** Select the most important (Essential) US ;

Else if *P(USi) == P(USj) & I(USi) == I(USj) & FS(USi) != FS(USj)* **then**
Select the user story with minimum functional size;

Else if *P(USi) == P(USj) & I(USi) == I(USj) & FS(USi) == FS(USj) & SS(USi) != SS(USj)* **then**
Select the user story with minimum Structural size;

Else
Select the user story that requires less demand on resources (time or budget);
End

Algorithm 1: Prioritizing user stories.

4.3.2 Decision-Making in Ongoing/ an Implemented Sprint

The evaluation of software size across various levels of detail is crucial not only for estimating effort/cost but also for facilitating decision-making, including budgetary allocations and portfolio management (Abran, 2010). In this section, we introduce Algorithms 2 , outlining a series of steps tailored for decision-makers (e.g., product owner, scrum master, development team) to guide their choices regarding Functional Change (FC) requests and Structural Change (SC) requests. The decision-making process entails the following actions: Accept the FC request and SC request, Deny the FC request and SC request, Defer the FC request and SC request

Deciding on a RC in an Ongoing Sprint

Algorithm 2*:* Deciding on a RC based FC and SC in an ongoing sprint.

Aim*:* Deciding on a FC and SC in an ongoing sprint Require: FS(FC), SS(SC), FS(USundone), SS(USundone),FS(USc), and SS(USc).

If FS(FC)>FS(USundone) &&SS(SC)>SS(USundone)**then**
Defer the FC to the next sprint;
Defer the SS to the next sprint;
Delete (USc)i from the ongoing sprint;
Add (USc)f to the next sprint;
Else if FS(FC) <FS(USundone)&&SS(SC) < SS(USundone) **then**
If FS(FC)>FS(USc)i && SS(SC)>SS(USc)$_i$ **then**
Defer the FC to the next sprint;
Defer the SC to the next sprint;
Delete (USc)i from the current sprint;
Add (USc)f to the next sprint;
Else if FS(FC)<FS(USc)&& SS(SC)<SS(USc) **then**
If FS(USc)f>FS(USc)i && SS(USc)f>SS(USc)$_i$ **then**
If Remainingtime(USc)f <requiredtime&&teamprogress= early **then**
Accept the FC;
Accept the SC;
Delete(USc)i from the current sprint;
Add(USc)f to the current sprint;
Else
Defer the FC;
Defer the SC;
Delete (USc)i
Add (USc)f to the next sprint;
Else if FS(USc)f<FS(USc)i && SS(USc) f<SS(USc)i **then**
Accept the FC; Accept the SC;
Delete(USc)i from the current sprint;
Add(USc)f to the current sprint;
Else if FS(FC) == 1 CFP &&SS(SC) == 1 CSM **Then**
Accept the FC; Accept the SC;
Delete (USc)I from the current sprint;
Add (USc)f to the current sprint;
End

Algorithm 2: Deciding on a RC.

5 EVALUATION

Table 3: A comparison between MD-DSS evaluation and experts' evaluations.

Change Evaluation / Software projects	MD-DSS	Expert1	Expert2
1	Low	Low	Moderate
2	Low	Moderate	Low
3	Moderate	Moderate	Moderate
4	Low	Moderate	Low
5	High	High	High
6	Moderate	Moderate	Moderate
7	Moderate	Moderate	Low
8	Moderate	Moderate	Moderate
9	Moderate	Low	Moderate
10	Moderate	Low	Moderate
11	High	High	High
12	Low	Low	Low
13	High	High	High
14	Low	Low	Moderate
15	Moderate	Moderate	Low

This section evaluates the MD-DSS by comparing its change evaluation with that of experts. This information helps stakeholders decide on the prioritization and acceptance, deferral, or denial of functional and structural changes. Feedback was collected from two experts on our MD-DSS, and 15 SCRUM-based final projects were used for evaluation. These diverse projects include mobile apps, web applications, business applications, and real-time software. We used the same database as in our previous work (Sellami et al., 2018).

5.1 Experts Evaluation

The MD-DSS was evaluated through an empirical verification based on a comparison between the results derived from an automated tool and that derived from experts (See Table 3).

These results are based on functional and structural size measurements, change size evaluations, and prioritization algorithms, compared with assessments from two experienced Scrum experts. The experts, with over 10 years in the industry, evaluated the importance of requirement change requests for 15 projects. They were provided with pre- and post-change functional and structural sizes, change descriptions, and change sizes. Experts classified each request as low, moderate, or high. In 81% of the cases, their classifications matched those of the MD-DSS. High-priority changes were accurately identified by both methods. However, for smaller changes, experts sometimes differed from the automated evaluation, often classifying them higher based on comparisons with other projects.

5.2 Comparative Evaluation

5.2.1 Case Studies and Results

The measurement results are given in Table 4. For each project we measure its functional size before and after the change noted by FSi(sw) and FSf(sw), respectively. We also measure its structural size before and after the change noted by SSi(sw) and SSf(sw). Then we measure the functional size of the change request manually and automatically using our MD-DSS noted by FS(FC)m, SS(SC)m and FS(FC)aut, SS(SC)aut, respectively. Based mainly on the functional size of the functional change and the structural size of the structural change, we determine the functional change status, the structural change status, both of them, manually and automatically noted by *FC status m* and *FC status aut* and *SC status m* and *SC status aut*, respectively (See Table 4).

Table 4: Experimentation result.

Sft	FSi(sw)	FSf(sw)	SSi (s w)	SSf (s w)	FCdescription	FS (FC)	FS (FC)aut	SS (SC)m	SS (SC)aut	FC status m	FCstatus aut	SCstatu sm	SCstatu s aut
1	47	50	10	12	Add US“ Contact administrator”	3	3	1	1	**M	**M	*L	*L
2	70	76	30	31	Add US“ communicate with other client”	6	6	2	2	**M	**M	*L	*L
3	80	88	30	32	Add US “Create User account”	8	8	2	2	**M	**M	*L	*L
4	43	46	15	17	Add US“ Create	3	3	1	1	**M	**M	*L	*L
5	40	56	10	15	Add three US	16	16	5	5	***H	***H	***H	***H
6	50	57	15	16	Add user story “Create user	7	7	1	1	**M	**M	*L	*L
7	22	17	2	3	Delete US “add employee”	5	5	1	1	**M	**M	*L	*L
8	27	31	8	9	Add US“ add new	4	4	1	1	**M	**M	*L	*L
9	47	51	10	12	Add US“create a	4	4	1	1	**M	**M	*L	*L
10	28	31	5	7	Add US“ publish a welcome	3	3	1	1	**M	**M	*L	*L
11	75	85	15	17	Add US “module	10	10	2	2	***H	***H	***H	***H
12	197	197	30	35	Modifying the US “Logon” users will logged on using an ID	3	1	1	1	*L	**M	*L	*L
13	105	97	20	30	Changes between 2versionsV1.0a	92	92	20	20	***H	***H	***H	***H
14	24	30	4	5	Add US “logon”	6	6	1	1	**M	**M	*L	*L
15	79	83	10	12	Add US “Registration”	4	4	1	1	**M	**M	*L	*L

L :*Low **M :M oderate *** H :High**

5.2.2 Evaluation Metrics

By analyzing all the results listed in Table 4, we noted that the MD-DSS gives exactly the same results (software functional size and status identification) for business applications, web applications and real time application. However, for the mobile apps (e.g., Restaurant management system) our MD-DSS could not measure correctly the functional size of the functional change respectively the structural size of the structural change as well as the functional change status respectively the structural change status. In fact, this deviation can be related to the update or reading information from the data storage device. It depends on whether the data are stored in an internal or external data storage devices. We compared the manual results to the automatic results generated by our tool by using the precision (see Eq. 2) and the recall (see Eq. 3) metrics. Thus, our tool achieved a precision and a recall equal to 93%.

$$\text{Precision} = \text{T P}/\ \text{T P} + \text{FP} \quad (2)$$

$$\text{Recall} = \text{T P}/\ \text{T P} + \text{FN} \quad (3)$$

Where: – TP: number of functional changes respectively the structural changes status correctly identified by our tool. – FP: number of functional changes respectively the structural changes status incorrectly identified by our tool. – FN: False negatives are the number of functional changes' status incorrectly not identified.

6 CONCLUSION

This research explores the importance of a decision support system based on functional and structural measures for managing change requests in the SCRUM process. The system evaluates requirement changes by quantifying them as user stories, aiding in prioritization and decision-making for product owners, Scrum masters, development teams, and managers. It was tested on 15 software development projects with expert input, comparing automated and manual methods. Future improvements will include incorporating factors such as risk, functionality use, complexity, urgency, change type, requestor, affected product parts, and dependencies, using AI for enhanced decision-making.

REFERENCES

Abran, A. (2010). *Software Metrics and Software Metrology*. IEEE Computer Society.

Abran, A. (2015). *Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers*. Wiley-IEEE Computer Society Pr, 1st edition.

Abdalhamid, S. and Mishra, A., 2017. Adopting of agile methods in software development organizations: systematic mapping. *TEM Journal*, *6*(4), p.817

Al Salemi, A. M. and Yeoh, E. T. (2015). A survey on product backlog change management and require- ment traceability in agile (Scrum). In *the 9th Malay- sian Software Engineering Conference (MySEC)*, pa- ges 189–194.

Ambler, S. W. (2014). *User Stories: An Agile Introduction*. Bano, M., Imtiaz, S., Ikram, N., Niazi, M., and Usman, M. (2012).

Causes of requirement change - a systematic literature review. In *EASE 2012*.

Berardi E., Buglione L., S. L. S. C. T. S. (2011). Guideline for the use of cosmic fsm to manage agile projects, v1.0.

Cohn, M. (2004). *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional.

Commeyne, C., Abran, A., and Djouab, R. (2016). *Effort Estimation with Story Points and COSMIC Function Points: An Industry Case Study*.

COSMIC (2017). *The COSMIC Functional Size Measurement Method, Version 4.0.2, Measurement Manual*.

COSMIC (2020). *The COSMIC Functional Size Measurement Method, Version 5.0, Announcement of Version 5.0 of the COSMIC Measurement Manual – March 31, 2020*

Drury-Grogan, M., O'Dwyer, O.: An investigation of the decision-making processin agile teams. *Int. J. Inf. Technol. Decis. Mak.* ***12****(6),* 1097–1120 (2013)

Desharnais, J. M., Kocaturk, B., and Abran, A. (2011). Using the cosmic method to evaluate the quality of the documentation of agile user stories. In *2011Joint Conf. of the 21st International Workshop on Software Measurement and the 6th International Conf. on Software Process and Product Measurement*, pages269–272.

Dikert, K., Paasivaara, M., and Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations. *Journal of Systems and Software*, 119(C):87–108.

Fairley, R.E.(2009).*Managing andLeadingSoftwareProjects*. Wiley-IEEE Computer SocietyPr.

Furtado, F., Zisman, A.: Trace++ (2016): a traceability approach to support transitioning to agile software engineering. In: *The 24th International Requirements Engineering Conference (RE), pp. 66–75.*

Gilb, T. (2018). Why agile product development systematically fails, and what to do about it!

Haoues, M., Sellami, A., and Ben-Abdallah, H. (2017). Functional change impact analysis in use cases: An approach based on COSMIC functional size measurement. *Science of Computer Programming, Special Issueon AdvancesinSoftwareMeasurement*,135:88– 104.

Hakim, H,.Sellami, A., and Ben-Abdallah, H. (2020). An in-Depth Requirements Change Evaluation Process using Functional and Structural Size Measures in the

Context of Agile Software Development. In ICSOFT (pp. 361-375).

Hamed, A.M.M and Abushama, H. Popular Agile Approaches in Software Development: Review and Analysis. Computing Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on (2013), pp. 160-166.

Download the official Scrum GuideLloyd, D., Moawad, R., and Kadry, M. (2017). A supporting tool for requirements change management in dis- tributed agile development. Future Computing and Informatics Journal, 2(1):1–9.

Schwaber, K. (2004). Agile Project Management with Scrum (Developer Best Practices). Microsoft Press; 1edition.

Sellami, A., Hakim, H., Abran, A., and Ben-Abdallah, H. (2015). A measurement method for sizing the struc- ture of UML sequence diagrams. Information & Software Technology, 59:222–232.

Sellami, A., Haoues, M., Borchani, N., & Bouassida, N. (2018, July). Guiding the Functional Change Decisions in Agile Project: An Empirical Evaluation. In International Conference on Software Technologies (pp. 327-348). Springer, Cham.

Sellami, A., Haoues, M., Borchani, N., & Bouassida, N. Orchestrating Functional Change Decisions in Scrum Process using COSMIC FSM Method. ICSOFT 2018: 516-527

Sellami, A., Haoues, M., Borchani, N., & Bouassida, N.Towards an Assessment Tool for Controlling Functional Changes in Scrum Process. IWSM-Mensura 2018: 34-47

Shalinka Jayatilleke, Richard Lai, A systematic review of requirements change management , Information and Software Technology 93 (2018) 163–185

Stålhane,T.,Hanssen,G.K.,Myklebust,T.,andHaugset,B. (2014). Agile change impact analysis of safety cri- tical software. In Bondavalli, A., Ceccarelli, A., and Ortmeier,F.,editors,ComputerSafety,Reliability,and Security, pages444–454.Verwijs, C. (2016).