SOFTENG 2018

The Fourth International Conference on Advances and Trends in Software Engineering

ISBN: 978-1-61208-632-3

April 22 - 26, 2018

Athens, Greece

SOFTENG 2018 Editors

Luigi Lavazza, Università dell'Insubria - Varese, Italy
Mira Kajko-Mattsson, KTH, School of Electrical Engineering and Computer Science, Sweden

SOFTENG 2018

Forward

The Fourth International Conference on Advances and Trends in Software Engineering (SOFTENG 2018), held between April 22, 2018 and April 26, 2018 in Athens, Greece, focused on challenging aspects for software development and deployment, across the whole life-cycle.

Software engineering exhibits challenging dimensions in the light of new applications, devices and services. Mobility, user-centric development, smart-devices, e-services, ambient environments, e-health and wearable/implantable devices pose specific challenges for specifying software requirements and developing reliable and safe software. Specific software interfaces, agile organization and software dependability require particular approaches for software security, maintainability, and sustainability.

We take here the opportunity to warmly thank all the members of the SOFTENG 2018 technical program committee, as well as all the reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated their time and effort to contribute to SOFTENG 2018. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

We also gratefully thank the members of the SOFTENG 2018 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope that SOFTENG 2018 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the field of software engineering. We also hope that Athens, Greece, provided a pleasant environment during the conference and everyone saved some time to enjoy the historic charm of the city.

SOFTENG 2018 Chairs

SOFTENG Steering Committee
Mira Kajko-Mattsson, Royal Institute of Technology, Sweden
Miroslaw Staron, University of Gothenburg, Sweden
Yoshihisa Udagawa, Tokyo Polytechnic University, Japan
Ulrike Hammerschall, University of Applied Sciences Munich, Germany

SOFTENG Industry/Research Advisory Committee
Philipp Helle, Airbus Group Innovations - Hamburg, Germany
Sigrid Eldh, Ericsson AB, Sweden
Tomas Schweigert, SQS Software Quality Systems AG, Germany
Michael Perscheid, Innovation Center Network, SAP, Germany
Janne Järvinen, F-Secure Corporation, Finland
Paolo Maresca, VERISIGN, Switzerland
Doo-Hwan Bae, Software Process Improvement Center - KAIST, South Korea

SOFTENG 2018
Committee

Table of Contents

System Requirements Prioritization Framework

Michel Högberg, Mira Kajko-Mattsson, Paulina Persson, Anne Håkansson
School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology
Stockholm, Sweden
e-mail: {michelh, mekm2, pauper, annehak}@kth.se

***Abstract*—Prioritization of system requirements is pivotal for coping with limited project resources. A well-structured and adequate prioritization method ensures that the most critical requirements get addressed first. Unfortunately, today, there are very few methods that are dedicated to requirements prioritization. This paper suggests a framework for prioritizing system requirements. The framework is called System Requirements Prioritization Framework (SRPF). It consists of eight components each representing a specific angle of the prioritization effort. Its components are (1) Input, (2) Stakeholders, (3) Prioritization Criteria, (4) Prioritization Methods, (5) Environment, (6) Resources, (7) Priority Scales, and (8) Urgency Levels. Our goal is to create a framework aiding companies in making structured and objective prioritization decisions. The theory on the framework's constituents and structure got educed in four consecutive exploration steps within the industry. The framework then got evaluated within the industry. Altogether, seventeen companies have been involved in the framework's exploration and five companies have been involved in the framework's evaluation. The evaluation results show that the framework is highly relevant and useful to the organizations studied.**

Keywords-Software project; development; prioritization method; decision making; customer benefit; corporate value.

I. INTRODUCTION

Projects have limited resources in terms of staff, time, and budget. Hence, it is not always possible to implement all the requirements in the current release or in the next coming releases [1]. Priorities must be made both by the stakeholders stating the requirements and the stakeholders attending to the requirements. Unfortunately, today, requirements are not always prioritized in an effective manner or they are not prioritized at all [2].

There are many reasons for the ineffectiveness of the requirements prioritizations. Stakeholders that state requirements believe that all their requirements are equally important. Hence, they are not always willing to prioritize them [3]. Stakeholders who attend to the requirements, on the other hand, do not always have adequate support for making priorities. Many try their best by using whatever tools they have. Many, however, still conduct prioritization in an ad hoc manner, often based on the will of some strongly opinionated individuals [4]. Or, as Stephen Covey claims, many companies prioritize what is on their schedule, and they do not schedule their priorities [5].

Lack of prioritization support may lead to many problems, such as (1) disagreements with respect to assigning priority [6][7], (2) too strong a subjectivity when prioritizing [8][9], (3) decisions conducted in uncertain conditions [10]–[12], (4) difficulty to reprioritize due to newly reported acute projects [6], (5) compliance among the prioritized requirements [7][9], (6) difficulties to implement all the requirements in the backlog, and many other problems. At its worst, the resources available will get quickly consumed on implementing less urgent requirements thus leaving scarce resources to the implementation of more urgent, business value adding requirements.

Priorities are very powerful. Even if companies have good resources, they may quickly jeopardize their productivity, if they spend them on requirements that have little bearing on the financial business health or other form of revenue or benefit of the software company and/or its customers. Despite this, requirements prioritization has been, and still is, one of the most difficult tasks in today's strongly chaotic and unpredictable development environments. Prioritization is also one of the most neglected research topics. To the knowledge of the authors of this paper, there is scarce literature about requirements prioritization [13]–[15].

This paper suggests a framework for prioritizing system requirements. The framework is called *System Requirements Prioritization Framework* and is referred to as SRPF. It consists of eight components, each representing a specific angle of the prioritization process. These components are (1) *Input*, (2) *Stakeholders*, (3) *Prioritization Criteria*, (4) *Prioritization Methods*, (5) *Environment*, (6) *Resources,* (7) *Priority Scales, and* (8) *Urgency Levels*. Our goal is to create an effective support aiding companies in making structured and objective prioritization decisions.

SRPF is a framework composed of a basic structure of the constituents required for making requirements priorities. As a framework, it is open for various kinds of adaptations and additions to the companies' own development milieus. Its mission is to support companies in their objective prioritization work within system development.

Altogether, seventeen companies were involved in this study. For confidentiality reasons, we do not disclose their names. Instead, we use fictitious names, whenever necessary.

The remainder of this paper is as follows. Section II describes the research method taken in this study. Section III presents the results of the exploration phase. Section IV describes the framework whereas Section V reports on the results of the framework evaluation. Finally, Section VI makes final remarks and suggestions for future research.

II. Research method

Our research method was a typical qualitative and inductive study. It was carried out in four phases. These are (1) *Exploration phase*, (2) *Design phase*, (3) *Evaluation phase*, and (4) *Fine-Tuning* phase.

Since limited research has been done within the area, we educed as much knowledge as possible about the current prioritization practice within the literature and industry. This phase was quite long and extensive. The results of the *Exploration* phase gave us enough feedback for designing the preliminary version of SRPF in the *Design* phase. The preliminary version was then evaluated in the *Evaluation* phase within the industry. Here, we used six evaluation criteria for assessing the relevance and usefulness of the SRPF. Finally, using the results of the *Evaluation* phase, we fine-tuned our framework and created its new improved version. Below, due to space restrictions, we only describe the *Exploration Phase* and the evaluation criteria.

A. Exploration Phase

We started our study with a thorough investigation of the domain of requirements prioritization. Here, we first made an extensive literature study using the following keywords: *prioritization, system requirements, decision making, customer benefit,* and *corporate value*. Unfortunately, this study resulted in very few sources on which we could base our research. Therefore, we continued to educe knowledge about the prioritization domain by studying the industrial practice. We did our exploration in four consecutive steps via interviews and surveys using the exploration questionnaires as presented in Figure 1.

First, we conducted a case study within *Company 1* using *Questionnaires 1* and *2*. Using *Questionnaire 1*, we interviewed two system development managers. Here, we focused on finding out (1) what the company's prioritization model looked like, (2) what criteria were considered in the prioritization work and how they were weighed, (3) whether the company used any predefined priority scales [16][17], (4) what the communication process looked like, and finally, (5) how they defined corporate value.

Using *Questionnaire 2*, we interviewed three business area managers, one technical manager, and the CEO. Here, we inquired about (1) what information was used when communicating on project prioritizations, (2) whether the business and system managers used any predefined priority scales, (3) whether any supporting tools were used, (4) what criteria were considered when prioritizing, (5) whether and how the company paid heed to the strategic goals, and finally, (6) how the company defined the expected value or benefit of attending to the prioritized system requirements.

To further broaden our insights into the prioritization work, we interviewed another company, *Company 2* using *Questionnaire 3*. By studying the questions, our reader may see that in addition to some questions that had already been asked in *Questionnaires 1* and *2*, we inquired about the project prioritization models and processes, their designs and uses, and the contexts of prioritization. Finally, we asked our interviewees to point out which of the criteria were the most important ones when doing prioritizations.

To assure that we have understood the requirements prioritization domain, we conducted a survey on the web using *Questionnaire 4*. Fifteen respondents were involved in this survey. Here, we first found out whether our respondents had the right competence for answering our questions. We also investigated what their companies and development departments looked like. Regarding prioritization work, we focused on finding out (1) what the respondents' work model looked like, (2) how they made priorities, (3) whether they used any criteria and methods for determining priorities, (4) whether any business strategic priorities were followed, and finally, (5) we inquired about the roles and responsibilities.

B. Evaluation Criteria

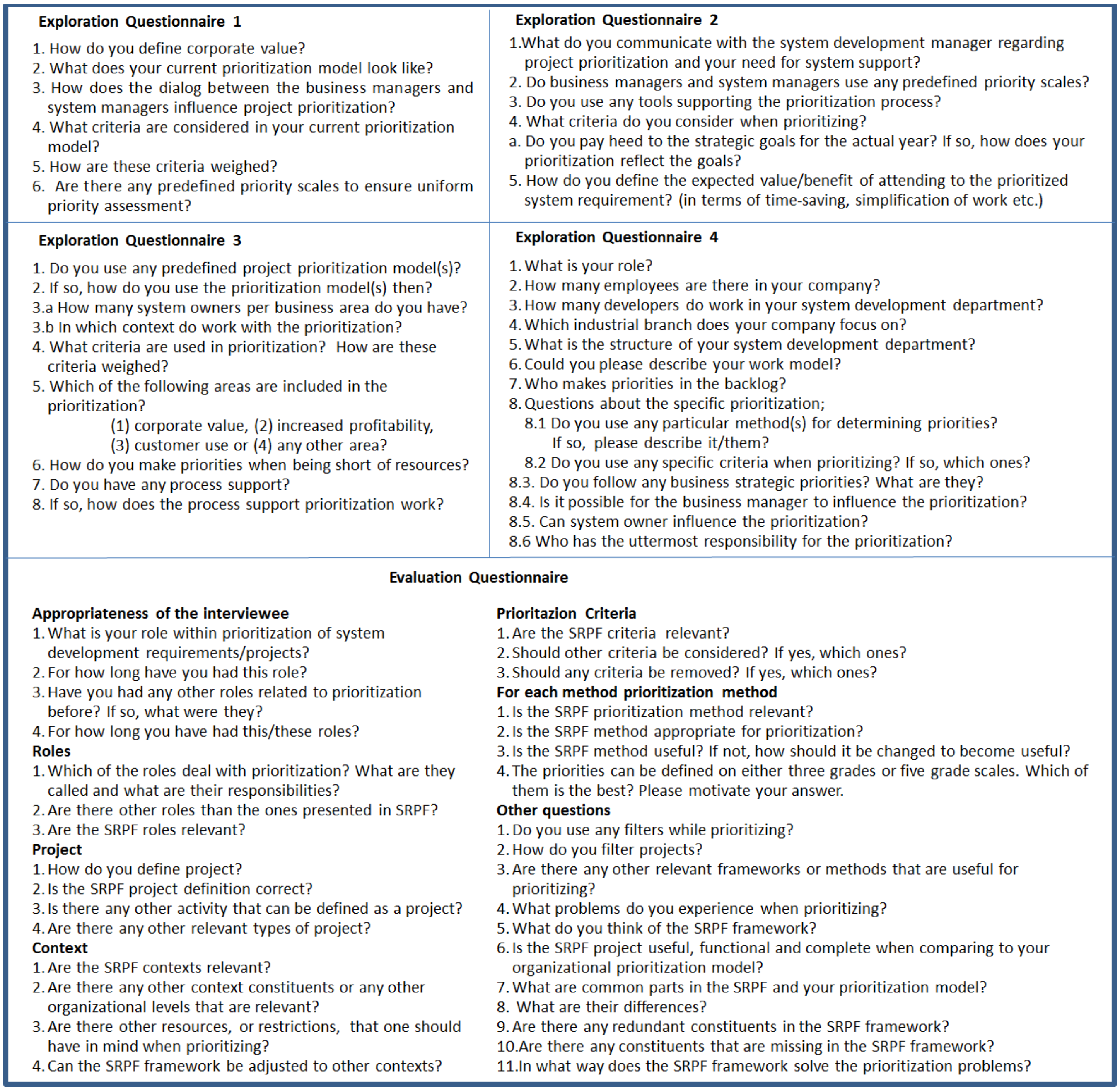
When evaluating SRPF, we used six evaluation criteria. These were:

1. *Appropriateness of the interviewees*: Using *Questions 1–4* in the *Evaluation Questionnaire* in Figure 1, we inquired whether our interviewees were suitable for evaluating our framework.
2. *Roles*: With *Questions 1–3*, we tried to find out what roles were involved in the prioritization work. We also asked our interviewees to express their opinions on the relevancy of the roles as suggested in our framework.
3. *Project*: One of the terms used in our framework is "project". To avoid misunderstanding with respect to its meaning, we inquired how the interviewees defined prioritization projects and whether our definition agreed with theirs. Here, we used *Questions 1–4*.
4. *Context*: Using *Questions 1–4*, we inquired whether the framework's context was (1) relevant, (2) whether anything was missing, (3) whether there were any resources or restrictions one should consider when prioritizing, and finally, (4) whether our framework could be adjusted to other contexts.
5. *Prioritization Criteria:* Using *Questions 1–3*, we wished to find out whether the framework's criteria were relevant or redundant, and whether any other criteria were missing.
6. *Prioritization Methods*: With *Questions 1–4*, we inquired whether the framework's prioritization methods were relevant, appropriate and useful for the interviewees' respective organizations. We also wished to hear their opinions about the number of grading levels to be used.

Other questions: Using *Questions 1–11*, we wished to hear the opinions of the interviewees about SRPF, how much it differed from their prioritization methods, and whether our framework missed any important components.

III. Exploration Results

In this section, we describe the results of the four *Exploration phases*. Due to space restrictions, we only provide additional feedback that got elicited during each consecutive phase.



Exploration Questionnaire 1

1. How do you define corporate value?
2. What does your current prioritization model look like?
3. How does the dialog between the business managers and system managers influence project prioritization?
4. What criteria are considered in your current prioritization model?
5. How are these criteria weighed?
6. Are there any predefined priority scales to ensure uniform priority assessment?

Exploration Questionnaire 2

1.What do you communicate with the system development manager regarding project prioritization and your need for system support?
2. Do business managers and system managers use any predefined priority scales?
3. Do you use any tools supporting the prioritization process?
4. What criteria do you consider when prioritizing?
a. Do you pay heed to the strategic goals for the actual year? If so, how does your prioritization reflect the goals?
5. How do you define the expected value/benefit of attending to the prioritized system requirement? (in terms of time-saving, simplification of work etc.)

Exploration Questionnaire 3

1. Do you use any predefined project prioritization model(s)?
2. If so, how do you use the prioritization model(s) then?
3.a How many system owners per business area do you have?
3.b In which context do work with the prioritization?
4. What criteria are used in prioritization? How are these criteria weighed?
5. Which of the following areas are included in the prioritization?
   (1) corporate value, (2) increased profitability,
   (3) customer use or (4) any other area?
6. How do you make priorities when being short of resources?
7. Do you have any process support?
8. If so, how does the process support prioritization work?

Exploration Questionnaire 4

1. What is your role?
2. How many employees are there in your company?
3. How many developers do work in your system development department?
4. Which industrial branch does your company focus on?
5. What is the structure of your system development department?
6. Could you please describe your work model?
7. Who makes priorities in the backlog?
8. Questions about the specific prioritization;
   8.1 Do you use any particular method(s) for determining priorities? If so, please describe it/them?
   8.2 Do you use any specific criteria when prioritizing? If so, which ones?
8.3. Do you follow any business strategic priorities? What are they?
8.4. Is it possible for the business manager to influence the prioritization?
8.5. Can system owner influence the prioritization?
8.6 Who has the uttermost responsibility for the prioritization?

Evaluation Questionnaire

Appropriateness of the interviewee
1. What is your role within prioritization of system development requirements/projects?
2. For how long have you had this role?
3. Have you had any other roles related to prioritization before? If so, what were they?
4. For how long you have had this/these roles?

Roles
1. Which of the roles deal with prioritization? What are they called and what are their responsibilities?
2. Are there other roles than the ones presented in SRPF?
3. Are the SRPF roles relevant?

Project
1. How do you define project?
2. Is the SRPF project definition correct?
3. Is there any other activity that can be defined as a project?
4. Are there any other relevant types of project?

Context
1. Are the SRPF contexts relevant?
2. Are there any other context constituents or any other organizational levels that are relevant?
3. Are there other resources, or restrictions, that one should have in mind when prioritizing?
4. Can the SRPF framework be adjusted to other contexts?

Prioritazion Criteria
1. Are the SRPF criteria relevant?
2. Should other criteria be considered? If yes, which ones?
3. Should any criteria be removed? If yes, which ones?

For each method prioritization method
1. Is the SRPF prioritization method relevant?
2. Is the SRPF method appropriate for prioritization?
3. Is the SRPF method useful? If not, how should it be changed to become useful?
4. The priorities can be defined on either three grades or five grade scales. Which of them is the best? Please motivate your answer.

Other questions
1. Do you use any filters while prioritizing?
2. How do you filter projects?
3. Are there any other relevant frameworks or methods that are useful for prioritizing?
4. What problems do you experience when prioritizing?
5. What do you think of the SRPF framework?
6. Is the SRPF project useful, functional and complete when comparing to your organizational prioritization model?
7. What are common parts in the SRPF and your prioritization model?
8. What are their differences?
9. Are there any redundant constituents in the SRPF framework?
10. Are there any constituents that are missing in the SRPF framework?
11. In what way does the SRPF framework solve the prioritization problems?

Figure 1 Questionnaires used in our study

A. *Results of Exploration Step 1*

Exploration Step 1 in *Company 1* revealed that none of the two interviewees used any predefined prioritization model. They neither used any predefined criteria nor any priority scales. All project prioritization was conducted in a merely ad hoc manner and varied among the two individuals being interviewed. Despite this, we received some insight into the company's prioritization work.

Typical evidence for lack of common prioritization method is their individual understanding of corporate value. *Interviewee 1* defines it as *profit, satisfied employees* and *satisfied customers* whereas *Interviewee 2* excludes customer satisfaction. This already automatically provides a basis for non-uniformity of their prioritization efforts.

Prioritization in *Company 1* is conducted on three levels: (1) system level implying development of a new system, (2) functionality level implying major change, and (3) minor change level. The predefined budget always constrains all prioritizations.

There is a very poor communication on setting priorities between business and system development managers. Business managers always set priorities first. System development managers then either accept or change them.

Very seldom do they provide feedback on the changes to the business managers.

The fact that *Company 1* does not have any predefined prioritization criteria implies that *Interviewee 2* follows his own subjective prioritization and project effort estimations. It happens that his prioritization choices are not always well motivated. *Interviewee 1*, on the other hand, uses the Information Technology Infrastructure Library´s four priority levels [18] subdivided into 99 sublevels.

B. *Exploration Step 2*

Just because *Exploration 1* did not provide us with much feedback, we once again interviewed individuals in *Company 1*, this time however, using *Questionnaire 2*. We interviewed five people and one of them *(Interviewee 1* in *Exploration 1)* was interviewed anew for confirming that we had understood him right.

The results of *Exploration 2* confirmed the results of *Exploration 1*. In addition, we found out that *Company 1* was strongly controlled by customers, not always in an orderly manner. The customers "*shouting the loudest get their wills easily satisfied*". This puts system development management in a very difficult position when trying to balance customer satisfaction and company's strategic goals. In some cases, the prioritization requests escalate to high-level management.

Regarding communication on the already prioritized projects, prioritizations and re-prioritizations of their individual requirements are being made on almost a daily basis. Here, project teams know the best how to prioritize them in the most effective manner.

None of the five interviewees uses any prioritization tools. Only two out of five interviewees use one common criterion when prioritizing, which is *company strategy*. This criterion is only used if conflicts arise. Finally, only one interviewee was able to state the expected value/benefit of attending to the prioritized system requirements. The value concerned savings in time and money.

C. *Exploration Step 3*

The results of *Exploration Step 3* reveal that even *Company 2* does not follow any predefined prioritization models. Our interviewees use their own individual models instead. The models are simple. They imply either regular meetings with follow ups or budget-controlled models. In any of the cases, the models include a strong interplay among many roles.

Irrespective of the models, all development in *Company 2* follows the *Phase-Gate* process model [19], which is the context of all its prioritization efforts. Just as in *Company 1*, big focus is being put on more important customers. In addition, the company has defined severity levels for each project to be prioritized. The priority is then defined based on severity value and the revenue to be gained. In cases, however, when several projects compete, the criterion that wins is the "*customer bigness*". When short of resources, the projects that hurt the least get the lowest priority.

When making priority decisions, *Company 2* regards areas such as (1) *corporate value,* (2) *increased profit,* and (3) *customer use*. Especially important is the customer use of the product. Big effort is being made to understand how the product is being used for the purpose of understanding the needs of the customers and the value of customer demands, and for making correct prioritizations.

D. *Exploration Step 4*

Altogether, fifteen people were involved in the survey in *Exploration Step 4*. They had the following roles: (1) six system developers, (2) three project leaders, (3) two managers, (4) two product owners, (5) one Unix administrator, and (6) one undefined. Three respondents came from very large companies with more than 500 employees, another three from large companies with more than 100 employees, and the remaining ones came from companies having more or less ten employees. The industries involved were banking and insurance, e-commerce, public services and various branches, such as general tech, gaming, farming, and amusement parks.

All except for one respondent could identify their work models as agile and lean related. In their respective companies, product prioritization is conducted by product owners (8 responses), project leaders (5 responses), and project teams (1 response).

Only two respondents could claim that they had a prioritization method. The prioritization criteria, as mentioned by the respondents, concerned ROI, customer impact, technical debt, emergency status, and the cost.

Regarding the roles responsible for prioritization, the following was provided: (1) business manager (2 responses) and product/system owners (10 responses) could influence the prioritization process, and finally, product/system owners (7 responses) and CTO or CEO (2 responses) had the uttermost responsibility for the prioritization.

E. *Exploration Phase in Summary*

The exploration phase taught us that prioritization was very complex and included several aspects. These are method, roles, context, prioritization criteria and resources.

Although many companies and roles are involved in prioritization, there are still companies who do not have a proper prioritization method. Lack of the method and lack of mutual criteria steering the prioritization effort imply great risk for subjective prioritization that may not always be aligned with the strategic goals.

IV. SRPF FRAMEWORK

In this section, we describe SRPF. We first provide an overall description of all its components. We then describe in detail the SRPF prioritization methods.

A. *Components in SRPF*

The preliminary SPRF consists of eight parts. As illustrated in Figure 2, these are (1) *Input,* (2) *Stakeholders,* (3) *Prioritization Criteria,* (4) *Prioritization Methods,* (5) *Environment*, (6) Resources, (7) *Priority Scales*, and (8) *Urgency Levels*. Below, we briefly describe the parts.

The SRPF *Input* stands for projects to be prioritized. Here, we include all projects that have not been prioritized and projects that need to be reprioritized for various reasons.

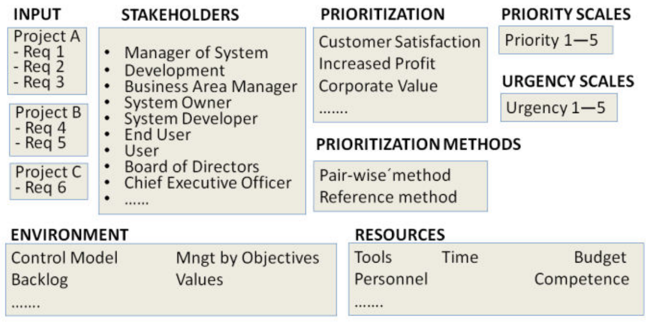


Figure 2. Outline of SRPF. Dots imply that the SRPF users are free to extend the framework parts with their own suggestions

The SRPF definition of a project is a set of requirements to be attended to. A set may consist of at least one requirement. Regarding the term requirement, SRPF defines it as a description of a need to get attended to. This need may either concern an implementation of a new functionality, minor improvement, a corrective or preventive change, and the like.

The SRPF *Stakeholders* correspond to a role or a group of roles that has interest or concern in a prioritization process. Stakeholders can affect or be affected by the prioritization process. Some examples are system managers, product owners, project managers, acquirers, business area managers, users, customers, and the like.

An important component in SRPF are *Prioritization Criteria*. To ensure achievement of strategic goals, the organizations must define criteria that help them identify the most urgent projects at a given point in time. SRPF leaves it open to its users to define their own criteria which they feel are the most suitable ones for their business operation and prioritization contexts. It, however, lists three criteria that are common to most of the organizations. These are (1) *corporate value*, (2) *increased profit,* and (3) *customer satisfaction*. To assure full commitment, the criteria should be well motivated and communicated to all the parties involved. It is only in this way, companies may assure the effectiveness of their prioritization efforts.

Prioritization Methods are the core of our framework. SRPF suggests two methods. These are (1) pair-wise and (2) reference methods. The pair-wise method compares projects pair-wise with all other projects meanwhile the reference method, which is a simplified version of the pair-wise method, compares all projects with one neutrally chosen reference project. The two methods are described in Section IV.A. As marked with dots in Figure 2, the SRPF users are free to extend the framework with their own methods.

A very important SRPF part is *Context*. It shapes the overall prioritization process. Context is very often neglected in many prioritization efforts or, for some reason, the stakeholders involved do not always attempt to explicitly communicate it.

Lack of a common understanding of a context may lead to many problems. For instance, information communicated by one stakeholder having his/her subjective understanding of a context may be easily misunderstood by some other stakeholder having his/her own subjective understanding. Hence, context must be explicitly identified by the company. Context describes what the organization looks like; where in the organization are decisions made, who has the authority to do prioritizations, whether there is a steering model, corporate values, backlog, and other important issues that are relevant for a specific organization. SRPF suggests the following contextual constituents:

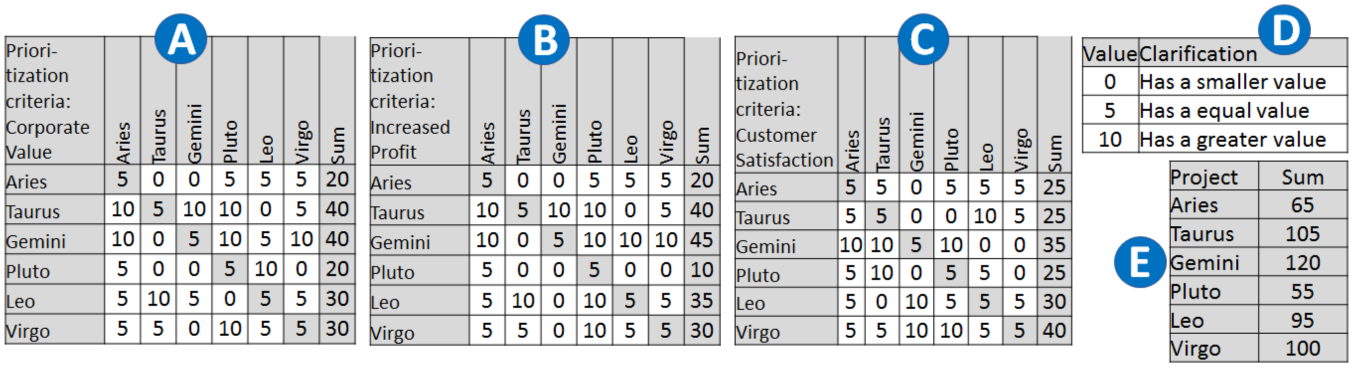
- *Control model* describing the decision making authorities and points in time when decisions are to be made.
- *Management by Objectives* aiming at decision making directed towards specific goals.
- *Backlog* listing all pending projects.
- *Values* referring to the organizational values to be considered in prioritization.

Finally, the SRPF *Resources* are sources of supply and support that are needed for conducting prioritization. Here, we include the following:

- *Tools* assisting the prioritization efforts, such as software, hardware and the like.
- *Personnel* referring to the individuals performing both development and prioritization.
- *Time* assigned to both prioritization and implementation of the pending projects.
- *Competence* standing for the collected organizational capability of attending to the prioritized projects.
- *Budget* referring to the amount of financial resources available for attending to the prioritization.

The SRPF Priority defines the urgency level for corrective action. It should be stated both by the customer and developer. The priority value as stated by the customers indicates how important it is for the customers to get the requirements attended to. Different customers, however, have different needs, different environments, and different safety and security requirements. The development organization cannot consider them all. They must define their own priority values that provide a basis for making their own priorities among the pending projects.

The SRPF Severity measures the effect of the disruption caused by a problem. Severity influences priority. For instance, a problem that could represent danger to human life or could cause failure of a company is most severe, and hence, its resolution should have the highest urgency. The value of priority, however, does not always influence the value of severity. High priority can be assigned even to less severe problems. Less severe yet frequent problems can be very costly and may lead to lowered credibility of the software organization [20][21]. Hence, they should be prioritized.



A

Priori-tization criteria: Corporate Value	Aries	Taurus	Gemini	Pluto	Leo	Virgo	Sum
Aries	5	0	0	5	5	5	20
Taurus	10	5	10	10	0	5	40
Gemini	10	0	5	10	5	10	40
Pluto	5	0	0	5	10	0	20
Leo	5	10	5	0	5	5	30
Virgo	5	5	0	10	5	5	30

B

Priori-tization criteria: Increased Profit	Aries	Taurus	Gemini	Pluto	Leo	Virgo	Sum
Aries	5	0	0	5	5	5	20
Taurus	10	5	10	10	0	5	40
Gemini	10	0	5	10	10	10	45
Pluto	5	0	0	5	0	0	10
Leo	5	10	0	10	5	5	35
Virgo	5	5	0	10	5	5	30

C

Priori-tization criteria: Customer Satisfaction	Aries	Taurus	Gemini	Pluto	Leo	Virgo	Sum
Aries	5	5	0	5	5	5	25
Taurus	5	5	0	0	10	5	25
Gemini	10	10	5	10	0	0	35
Pluto	5	10	0	5	5	0	25
Leo	5	0	10	5	5	5	30
Virgo	5	5	10	10	5	5	40

D

Value	Clarification
0	Has a smaller value
5	Has a equal value
10	Has a greater value

E

Project	Sum
Aries	65
Taurus	105
Gemini	120
Pluto	55
Leo	95
Virgo	100

Figure 3. Illustrating pairwise method

B. *Prioritization Methods*

The pair-wise and reference methods are almost similar in their designs. In this section, we first describe the pair-wise method. We then describe the reference method. Finally, we discuss their similarities and differences.

Pair-wise method is a well-known process of comparing various entities in pairs with the purpose of deciding which of them is better. It has been used in various domains such as education, engineering, energy and water resources, management, and environmental applications [22]–[24]. It has also been used within requirements engineering, however, mainly from the cost perspective. In contrast, the SRPF pair-wise method considers all kinds of criteria to be used in comparison, not only the cost.

The choice of the comparison criteria is to be decided by the company using SRPF. Below, we provide an example based on three values: (1) *corporate value*, (2) *profit*, and (3) *customer satisfaction*. Projects get assessed pairwise using only three numerical values. These are (1) zero standing for "*has lower value*", (2) five standing for "*no difference*" and (3) ten standing for "*has higher value*".

Subfigures A-C of Figure 3 show pair-wise evaluation of the projects with respect to corporate value, profit and customer satisfaction respectively using the values specified in *Subfigure D of Figure 3*. These values are then summed up for all the comparison criteria. As shown in *Subfigure E of Figure 3*, the project called *Gemini* gained the highest score which is 120 points. This project should get the highest priority. If the criteria have different mutually important weights, typically 1 to 3, then each score can be multiplied with the weight before summarization. Then the criterion with the highest weight will be more important in the final sum.

Reference method is also a comparison method. To the knowledge of the authors of this paper, no one has used it in within requirements engineering. SRPF suggests that all competing projects are compared to only one neutral project. As illustrated in the third column in Subfigure A of Figure 4, this neutral project is called *reference project*. All other projects that are to be compared are given a value depending on how similar or different they are to/from the reference project.

Five values are assigned to the competing projects. These are (1) '+' better, (2) '++' much better, (3) 'S' similar, (4) '-' lower, (5) '- -' much lower. When a project is better than the reference project, a plus (+) is assigned to the project. If the difference is judged enormously bigger, then two plusses (++) are assigned. If the project is equal, then an 'S' is given standing for the same value. If the project is worse or much worse, then a minus (-) or two minuses (- -) are assigned.

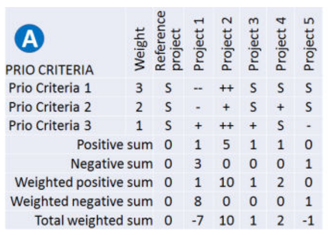
Just as in the pair-wise method, the reference method may use different criteria. Their score may then be summed giving total scores of the projects. An example comparison is illustrated in Subfigure B of Figure 4.

Adding a weight to each criterion is possible. When all projects are compared, a summarization of the respective grading values are made adding the weight and then a total sorting of the points gives the priority value.

C. *Comparing the two methods*

When comparing the methods, reference method is like the pair-wise method. There are however some differences. These are:

- Pair-wise method uses several matrices whereas reference method only uses one for comparison regardless of how many criteria are being compared.
- Pair-wise method does not use any reference project. If the reference project is not a good choice, then the whole prioritization effort is at risk.
- Pair-wise method compares all projects with one another whereas reference method compares all other projects to the reference project. This implies that in the reference method, the number of comparisons is linear to



A

PRIO CRITERIA	Weight	Reference project	Project 1	Project 2	Project 3	Project 4	Project 5
Prio Criteria 1	3	S	--	++	S	S	S
Prio Criteria 2	2	S	-	+	S	+	S
Prio Criteria 3	1	S	+	++	+	S	-
Positive sum		0	1	5	1	1	0
Negative sum		0	3	0	0	0	1
Weighted positive sum		0	1	10	1	2	0
Weighted negative sum		0	8	0	0	0	1
Total weighted sum		0	-7	10	1	2	-1

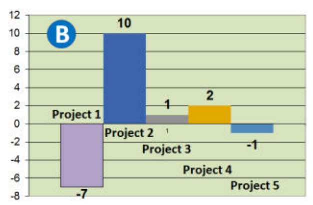


Figure 4. Illustrating reference method ((1) '+' better, (2) '++' much better, (3) 'S' similar, (4) '-' lower, (5) '- -' much lower)

the number of projects, while it is quadratic in the pair-wise method.

When using SRPF, companies are free to choose any of the two methods, or use the two methods, or add their own methods. SRPF suggests that reference method be used first for quick filtering of less important projects and then pair-wise method be used for more meticulous comparisons.

V. Results of the Evaluation interview

In this section, we report on the results of the evaluation phase following the evaluation criteria as defined in Section II and the evaluation questionnaire.

All our seven interviewees were suitable for evaluating SRPF. Five of them have more than ten years and two of them have more than five years of experience of prioritizing within system development. All seven are managers involved in system development.

The roles that have been mentioned as active within prioritization were project leaders, system architects, product managers, product/system owners, line managers, technology managers, functional managers, quality managers, system development managers, business area managers, users, boards of directors, CEOs, and developers.

All the interviewees agreed with the SRPF project definition. However, they all had their own variants. Important is to say that they pointed out that the SRPF definition missed the concept of goal, scope, time and resource constraints and quality requirements. Also, there have been suggestions for differentiating between projects as temporary created for fulfilling some specific goal and permanent dealing with continuous improvements and defect corrections.

All seven interviewees agree that SRPF's context is relevant. Five interviewees find the context very general and all seven interviewees agree that SRPF is useful in almost any prioritization effort. Three interviewees came with highly valuable suggestions. One of them claimed that the technological context was missing and another one suggested that the methodological context should include the identification of the development method used and the placement of projects within its phases. Certain projects could only be placed at the beginning of a development method whereas others could be placed anywhere within the lifecycle. Finally, the third interviewee mentioned the business context for considering business strategic goals.

All the interviewees agreed that the prioritization criteria in SRPF were relevant and that no criteria were redundant. They however pointed out new criteria that were highly relevant. These are *cultural value, politics, customer, business value, customer value, focus on new markets, risks, time,* and *competence*.

All the interviewees agreed that the SRPF prioritization methods (pair-wise and reference) were relevant, appropriate and useful and that they contributed to the objective prioritization. Neither of the interviewees found the methods to be redundant. Some comments were made regarding the reference method. Two interviewees were of the opinion that the method was difficult. The difficulty lied in the choice of a reference project. Regarding the used scale in the method, five interviewees would choose the five-grade scale as suggested by SRPF, one interviewee would rather use a four-grade scale whereas one interviewee claimed that six grades would be the best.

Regarding the answers to the batch of the remaining questions, the results are as follows. All seven interviewees do filter some projects before prioritization. One interviewee knows about a framework similar to SRPF as conducted within the industry.

When prioritizing, the interviewees encounter many problems. In addition to the problems as listed in Section I, the problems that have been mentioned are (1) difficulties in down-prioritizing projects, (2) agreeing on common criteria to base prioritizations on, (3) inability to prioritize all projects in the backlog, and finally, (4) lack of time for making prioritizations.

When being asked about the overall impression of SRPF, all seven interviewees found it interesting and good. Five of them claimed that they did work in a similar way as suggested in the framework. None of the interviewees found anything redundant in SRPF.

VI. Final Remarks

This paper suggests *System Requirements Prioritization Framework*. Our goal was to create an effective support aiding companies in making structured and objective decisions when prioritizing requirements.

As a framework, SRPF is open for various kinds of adaptations and additions to the companies' own development milieus. Its mission is to support companies in their objective prioritization work within system development. Right now, it only outlines the most important prioritization components. Hence, we suggest it be an initial version for both the industry and the academia. We also propose to further evolve it.

Even if SRPF is in its initial phase, it is already more advanced than the existing prioritization methods [8][25]. These methods are very simple in their designs. They mainly state that requirements should be prioritized and, at its most, suggest priority grades. Hence, they are incomparable to SRPF

During the evaluation phase, we discovered that our definition of a project was too broad. For this reason, we broaden it with project goal, scope, and constraints. The new definition of a project *is a set of requirements to be attended to that has a clearly specified goal and scope, and that is bounded by a set of clearly specified constraints. A set consists of at least one requirement.*

When evolving the framework, we suggest the following issues to be further researched on. Priorities and urgencies are complex issues, and little has been done to identify scales and/or variances of scales to reflect their levels and relationships. We suggest that more effort be put into ways of defining priorities and urgencies.

Regarding the context, we must admit that we have forgotten one very important criterion concerning the ethics. Ethics has not been explicitly mentioned during the evaluation interviews. However, by studying the interview results, we understood that it was implicitly hidden behind many answers. We suggest that this criterion be considered and researched on in the context of prioritization. Also, methodological and technological contexts should be considered in the framework.

Considering the roles, we have noticed that prioritization involved a multitude of roles. SRPF has only identified a subset of them. We believe that more research should be conducted on identifying the roles involved and their responsibility portfolios so that each role may contribute to the prioritization efforts in its best possible way.

So far, SRPF has been explored and evaluated via interviews and surveys. Even if industrial professionals accepted the framework, it would be good if the framework were used in an industrial context. Therefore, we warmly welcome anybody to use SRPF and provide feedback from its real-life usage.

REFERENCES

[1] A. Purnus and C.-N. Bodea, ”Project Prioritization and Portfolio Performance Measurement in Project Oriented Organizations,” *Procedia - Social and Behavioral Sciences,* vol. 119, pp. 339-348, 2014.

[2] M. Alexander, ”CIO from IDG,” IDG Communications, Inc., 23 11 2015. [Online]. Available from: https://www.cio.com/article/3007575/project-management/6-proven-strategies-for-evaluating-and-prioritizing-it-projects.html 2018.3.16

[3] M. Schedlbauer, ”Corporate Education Group,” 2017. [Online]. Available from: http://www.corpedgroup.com/resources/ba/ReqsPrioritization.asp 2018.3.16

[4] Sweden, ” World-class foreign affairs: final report”, The governments official investigations, ”Utrikesförvaltning i världsklass : slutbetänkande,” *Statens offentliga utredningar SOU 2011:21*, Stockholm, Offentliga Förlaget : Publit, 2011, pp. 457-459.

[5] S. R. Covey, ”Goodreads,” Goodreads Inc, 2017. [Online]. Available from: https://www.goodreads.com/quotes/706652-the-key-is-not-to-prioritize-what-s-on-your-schedule 2018.3.16

[6] O. Maassen and J. Sonnevelt, "Kanban at an Insurance Company (Are You Sure?)," in *Agile Processes*, Trondheim, Norway, 2010.

[7] B. Nuseibeh and S. Easterbrook, "Requirements engineering: a roadmap," in Proceedings of the Conference on The Future of Software Engineering, Limerick, Ireland, 2000.

[8] E. T. Game, P. Kareiva, and H. P. Possingham, ”Six Common Mistakes in Conservation Priority Setting,” *Conservation Biology,* vol. 27, nr 3, p. 480–485, 2013.

[9] I. Sommerville and P. Sawyer, ”Viewpoints: principles, problems and a practical approach to requirements engineering,” *Annals of Software Engineering,* vol. 3, nr 1, pp. 101-130, 1997.

[10] C. C. Chou, ”A fuzzy MCDM method for solving marine transshipment container port selection problems,” *Applied Mathematics and Computation,* vol. 186, nr 1, pp. 435-444, 2007.

[11] S. L. Chang, R. C. Wang, and S. Y. Wang, ”Applying fuzzy linguistic quantifier to select supply chain partners at different phases of product life cycle,” *International Journal of Production Economics,* vol. 100, nr 2, pp. 348-359, 2006.

[12] X. D. Luo, J. H. M. Lee, H. F. Leung, and N. R. Jennings, ”Prioritised fuzzy constraint satisfaction problems: axioms, instantiation and validation,” *Fuzzy Sets and Systems,* vol. 136, nr 2, pp. 151-188, 2003.

[13] J. Vähänitty and K. T. Rautiainen, ”Towards a conceptual framework and tool support for linking long-term product and business planning with agile software development,” *Association for computing Machinery,* pp. 25-28, 2008.

[14] J. Karlsson, C. Wohlin, and B. Regnell, ”An evaluation of methods for prioritizing software requirements,” *Information and Software Technology,* vol. 39, no 14-15, pp. 939-947, 1998.

[15] M. Kajko-Mattsson, ”Corrective maintenance maturity model : problem management,” Stockholm, 2001.

[16] M. Kajko-Mattsson, "Conceptual model of software maintenance," Forging New Links, Proceedings of the 1998 International Conference on Software Engineering (ICSE 98) Apr. 1998, pp. 422-425, ISBN:0-8186-8368-6

[17] M. Kajko-Mattsson, ” Experience Paper: Maintenance at ABB (I): software problem administration processes,” 1999 International Conference on Software Maintenance (ICSM 1999), Sep. 1999, pp. 167-176, ISSN: 1063-6773, ISBN: 0-7695-0016-1

[20] B. Beizer, Software System Testing and Quality Assurance, Van Nostrand Reinhold Company, 1984.

[21] C. Kaner, J. Falk, and H. Q. Nguyen, Testing Computer Software, John Wiley & Sons, Inc., 1999.

[22] L. Dias, M. Pirlot, P. Meyer, R. Bisdorff, and V. Mousseau, International handbooks on information systems. Evaluation and decision models with multiple criteria, Berlin: Springer, 2015.

[23] J. Wallenius, J. Dyer, P. Fishburn, R. Steuer, S. Zionts, and K. Deb, ”Multiple criteria decision making, multiattribute utility theory: recent accomplishments and what lies ahead,” *Management Science,* vol. 54, nr 7, pp. 1336-1349, 2008.

[24] C. Zopounidis and P. Pardalos, Handbook of multicriteria analysis, Berlin: Springer, 2010.

[25] V. Ahl, ”An Experimental Comparison of Five Prioritization Methods,” Blekinge Sweden, 2005.

Data-Driven Testing using TTCN-3

Bernard Stepien, Liam Peyton
School of Engineering and Computer Science
University of Ottawa
Ottawa, Canada
Email: {bstepien | lpeyton}@uottawa.ca

Mohamed Alhaj
Computer Engineering Department
Al-Ahliyya Amman University
Amman, Jordan
Email: m.alhaj@ammanu.edu.jo

***Abstract*—Complex software systems orchestrate interactions between components of the system. Integration testing of such systems involves making individual unit tests for individual components that work together to test the interactions between components. Unit tests for different components often consist of heterogeneous representations of test data and test behavior written in various implementation languages. As a result, in integration testing it is an advantage to use a single formal testing language like TTCN-3 (Testing and Test Control Notation Version 3). We propose a transformation tool for Data-Driven Testing to generate TTCN-3 test suites that include data types, templates and test behavior from tables. This process is relatively straightforward for relational data bases and XML (eXtensible Markup Language) because they are based on well-defined data models. Excel is more complex because it has no such data models. We have developed a tool that assists the tester in extracting TTCN-3 typing information from Excel tables to produce TTCN-3 templates and test behaviors and optimize their re-usability.**

Keywords: Data-driven Testing; Testing; TTCN-3; re-usability.

I. INTRODUCTION

Complex software systems orchestrate interactions between components of the system. Integration testing involves making individual unit tests for individual components that work together to test the interactions between components. Unit testing alone does not guarantee that components interact correctly. Unit tests for different components often consist of heterogeneous representations of test data and test behavior written in various implementation languages. Ideally, integration testing would use a single formal testing language like TTCN-3 (Testing and Test Control Notation Version 3).

Data-Driven Testing (DDT) is well known in industry. There are a variety of industry-oriented definitions online and the concept is discussed and explained in detail in Web sites [4][6], user forums [5][11], frameworks [9][12], patents [13][14], application domains [10] and linked with other testing models [3][15]. The basic principle consists of separating test data (inputs and expected outputs) from test scripts (test behavior) as shown in Figure 1. The test data is stored as tables in relational databases, XML (eXtensible Markup Language) documents or Excel spreadsheets. More advanced test technologies such as TTCN-3 [3] allow a flexible separation of concerns between an abstract layer that consists of test data and test logic and a concrete layer that consists of codecs to encode and decode data into the specific format and protocol needed to test a component. In particular, the TTCN-3 concept of template to represent test data and expected responses is reusable whereas simple DDT is not, and TTCN-3 strong typing enables early detection of errors in test data.

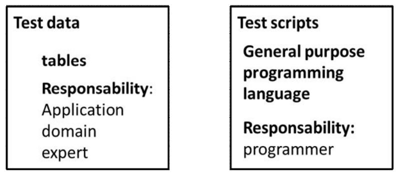


Figure 1. DDT separation model

Thus, we propose a transformation tool to generate TTCN-3 test suites that include data types, templates and test behavior, from DDT tables. This process is relatively straightforward for relational data bases and XML because they are based on well-defined data models. However, the case of Excel [1] is more complex because such data models do not exist. We have developed a tool that assists the tester in extracting TTCN-3 typing information from Excel tables to produce TTCN-3 templates and test behavior and optimize their re-usability.

The rest of the paper is structured as follows. In section II we present an overview of data-driven testing and TTCN-3. In section III, we present our approach for transforming data-driven test tables into TTCN-3 test suites. In section IV, we present our tool implementation and evaluation. And finally, in section V, we present the conclusion.

II. DATA-DRIVEN TESTING AND TTCN-3

The main goal of DDT is to allow application domain experts without programming skills to prepare test data and to reduce maintenance costs. Test data is commonly stored in tables using one of the following three mechanisms:

- Relational databases
- XML documents
- Excel tables

While the two first approaches provide data models (table column descriptions for relational databases and XML schema for XML documents) and are thus unambiguous, Excel spreadsheets do not. The data models are absent because tables contain only data with column headings. Although one can set data types for the cells of a column mostly to specify the display format for numeric types (number of decimal digits for numbers), the default data type is the *general* data type. Also, there is no explicit definition of field names. Only column headings hint at what the fields in a structured data type could be.

Another challenge in DDT is that tests are strictly sequential as it is impossible to describe alternatives easily with tables only. Thus, a test step consists of reading a row of data, performing the test by either sending a message to the system under test (SUT) or invoking a function with parameters and checking the response message of the SUT or the return value of the function against a test oracle (expected response). It is the responsibility of the programmer of the test script to determine the exact location of the various pieces of data in the table to transfer them to the fields of some structured type variable and distinguish what is test data from what is a test oracle.

The test scripting language TTCN-3 has been used for model-driven testing in general, and has many features that make it an effective tool for DDT. TTCN-3 is based on a separation of concerns between an abstract layer and a concrete layer. The basic elements of an abstract layer consist mainly in the following components:

- Data typing definitions
- Templates definitions
- Behavior definitions

As shown in Figure 2, separate template definitions for Test Data, and separate test behavior definitions for test scripts means that TTCN-3 has the same separation model that DDT has (as shown in Figure1).

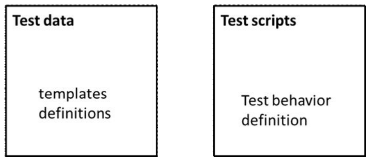


Figure 2. TTCN-3 separation model

A TTCN-3 template defines test data (stimuli or test oracle). Each template has a name that can be referred to in behavior definitions or reused in further template definitions like a variable. For test oracles, that variable contains program code used to verify that a response corresponds to the test oracle. The concrete layer consists of codecs that translate abstract into concrete data and vice versa and communicates with the SUT. We present three examples next.

Data type definition example for a structured type

The main difference with Excel-based DDT is that in TTCN-3 we define data types as shown in Figure 3 to be able to define templates. This is because in TTCN-3, the matching of test oracles is achieved at once for all test data as opposed to the DDT approach of using an atomic assertion mechanism for each individual piece of data .

```
type record MyCarRequestType {
   integer nbDoors,
   charstring model,
   charstring brand }

type record MyCarResponseType {
   charstring model,
   charstring brand,
   float listPrice }
```

Figure 3. TTCN-3 Data Typing Example

XML or Database [16] based DDT is handled via a built-in mechanism of TTCN-3 tools that translates for example an XML schema directly into TTCN-3 data types.

Template definition example

A TTCN-3 template as show in Figure 4 resembles a structured type variable assignment but in essence it is very different from a typical programming language variable. The values being assigned to the fields of this structured data type have two different meanings depending of the direction of a message in the communication system. When using the template for sending data, they are plain data that is either encoded to be sent in the case of messages or values of parameters for a function being invoked. When using the template as a test oracle, the values mean that the response message or return value must match the values given in the template. The matching mechanism itself is a built-in feature of TTCN-3 execution tools and thus does not need to be programmed by the users. Thus, a TTCN-3 template is more like an implicit program.

```
template MyCarRequestType
             myTiguanRequest := {
   nbDoors := 5,
   model := "Tiguan",
   brand := "VW"
}

template MyCarResponseType
             myTiguanResponse := {
   model := "Tiguan",
   brand := "VW",
   listPrice := 35000.00
}
```

Figure 4. TTCN-3 Template Example

Behavior definition example

Behavior definitions as shown on Figure 5 consist in sending data to the SUT and trying to match a response or return value to a test oracle. The TTCN-3 *send* and *receive* commands use template names where data or test oracles are defined. TTCN-3 *receive* statements are usually contained in an *alt* statement (alternative). This is to handle various potential responses and assign a corresponding verdict (pass or fail). The generic *receive* without parameters means receive any value and tester typically assign a fail verdict with such a construct.

```
myPort.send(myTiguanRequest);
timer myTimer = 5.0;
alt {
    [] myPort.receive(myTiguanResponse)
            { setverdict (pass)}
    [] myPort.receive
            {setverdict(fail)}
    [] myTimer.timeout
            {setverdict(inconc)}
}
```

Figure 5. TTCN-3 Test Behavior Example

TTCN-3 also has timers that can be set and timeouts are part of an alternative. If any of the receive statements in the alternative do not match the response, eventually the timer will time out and a corresponding verdict can be set. Also, the receive statement is not fully equivalent to an assertion. When a receive statement fails, TTCN-3 merely tries the next alternative. This is similar to a rule based system.

Because a template is like a variable, it is fully re-usable either in different tests but also in the definition of other templates where a field is of the data type of the re-usable template. Another interesting aspect of templates is that since templates are referenced by name, when performing tests with the same data, it doesn't need to be redefined or read for each test like in DDT. More important is the feature that allows deriving a template from another existing template by specifying only the delta, thus avoiding specifying portions of the same data several times.

```
template MyType myGolfRequest
      modifies myTiguanRequest := {
                 model:= "Golf" }
```

Figure 6. TTCN-3 template modification Example

Transforming DDT into TTCN-3 has several benefits. From a language point of view, TTCN-3 is based on strong typing. Strong typing allows one to restrict the usage of data by type. In other constructs, such as templates data, being sent or received can be set to a precise type. In loose table formats such as Excel, there is no way to specify such restrictions which inevitably leads to undetectable errors at design time. Relational databases or XML documents are typed but not always strongly. For example in relation databases there is no way to specify exactly which values are allowed in a specific data type. In our *MyCarRequestType*, we could have further refined this type definition by restricting the brand field type. Instead of using the generic *charstring* type, we could have defined a *brandType* as follows:

```
type charstring brandType
( "VW", "Mercedes", "Renault", "Fiat",
"Ford", "Chrysler" );
```

Figure 7. TTCN-3 type Restriction Example

Then this *brandType* could have been used in the *MyCarRequestType* as follows:

```
type record MyCarRequestType { integer
nbDoors, charstring model, brandType
brand }
```

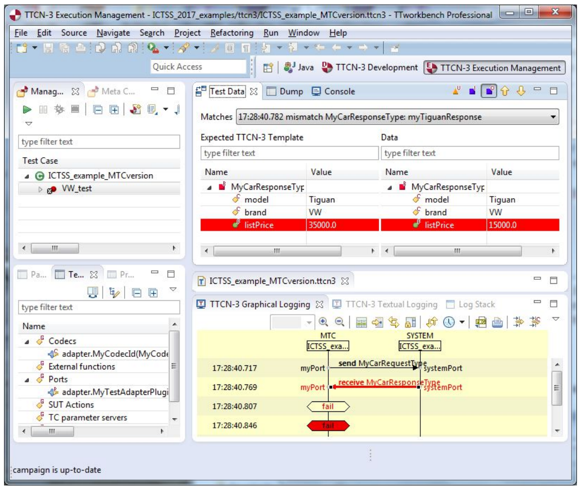
Figure 8. TTCN-3 data sub-typing Example

The use of a brand name, other than the one found in the list of the data type *brandType,* would cause a compile error. In DDT, the same error would be detected only at run time. The following example would trigger a compile error.

```
template MyCarRequestType
myToyotaRequest := {
   nbDoors := 5,
   model := "Corolla",
   brand := "Toyota"
}
```

Figure 9. TTCN-3 template with Restricted sub-type Example

The other benefit of TTCN-3 is in its test results display. Each test event (send or receive) is displayed and TTCN-3 tools allow for inspection of the results by providing a comparison between the response data received and the test oracle as shown in Figure 10 where the expected *listPrice* of $35000.00 did not match the response value of $15000.00.


TTCN-3 Execution Management - ICTSS_2017_examples/ttcn3/ICTSS_example_MTCversion.ttcn3 - TTworkbench Professional
File Edit Source Navigate Search Project Refactoring Run Window Help
Quick Access
Java
TTCN-3 Development
TTCN-3 Execution Management
Test Data
Dump
Console
ICTSS_example_MTCversion.ttcn3
TTCN-3 Graphical Logging
TTCN-3 Textual Logging
Log Stack
MTC
SYSTEM
ICTSS_exa...
campaign is up-to-date

Matches 17:28:40.782 mismatch MyCarResponseType: myTiguanResponse

Expected TTCN-3 Template: Name	Value	Data: Name	Value
MyCarResponseTyp		MyCarResponseTyp	
model	Tiguan	model	Tiguan
brand	VW	brand	VW
listPrice	35000.0	listPrice	15000.0

```
17:28:40.717   myPort   send MyCarRequestType      systemPort
17:28:40.769   myPort   receive MyCarResponseType  systemPort
17:28:40.807   fail
17:28:40.846   fail
```

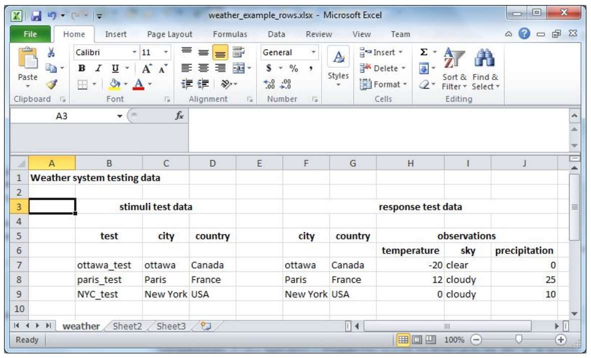
Figure 10. TTCN-3 tools results inspection feature

Transforming relational data bases into TTCN-3 have already been handled by Stepien et al. [7, 8]. They are also supported by most TTCN-3 tools. However, until now, the conversion of Excel tables into TTCN-3 has not been addressed in TTCN-3 or the academic literature.

III. TRANSFORMING TABLES INTO TTCN-3 TEST SUITES

Transforming Excel tables into TTCN-3 test suites consists of determining data types which include field names of the implicit structured type that a table represents and the type of each such field. Also, we need to distinguish what is data to be sent from data that represents a test oracle.

For example, in the Excel table shown in Figure 11, we can find two sub-tables, one for stimuli test data and one for response test oracles. The stimuli sub-table is a simple structured type while the response test oracle table is a complex structured type where the observations field is itself a structured type.



	A	B	C	D	E	F	G	H	I	J
1	Weather system testing data									
2										
3		stimuli test data				response test data				
4										
5		test	city	country		city	country	observations		
6								temperature	sky	precipitation
7		ottawa_test	ottawa	Canada		ottawa	Canada	-20	clear	0
8		paris_test	Paris	France		Paris	France	12	cloudy	25
9		NYC_test	New York	USA		New York	USA	0	cloudy	10
10										

Figure 11. Excel table to be converted example

The problem is how to automate the process of determining the data types and location where to read data and then transforming them into TTCN-3. While there have been cases of Excel tables converted to TTCN-3 in some industrial projects, there are no publications about the process because typically each Excel spreadsheet was handled manually on an ad hoc basis to determine data type and where to read the appropriate data.

We have approached this conversion problem in two different ways: first we considered a fully automated conversion using principles of artificial intelligence where the system would locate the table of data automatically by for example discovering that a column contains data of the same data type, then consider the data found in the rows preceding the data as headings and any other loose and isolated row as comments. However, one major problem with this approach is that there is no indication in Excel tables as to what a stimulus is and what a response is. This results in inconsistencies. In a second approach we have used an interactive mock-up of the Excel table for the tester to delimit the portions of the table that corresponds to either column heading, stimuli data and response data. This gives the tester control over the specification of stimuli and responses.

Such a tool is more efficient than the traditional approach of hard coding the locations of data in test logic and creating the data type definitions manually. Also, this process is of value when considering economies of scale with large numbers of Excel tables. It large projects with extensive use of DDT there could be tens of thousands of such tables.

This is a fundamental choice based on the principle of strong typing. Effectively, if we would follow the DDT model of reading data from tables and applying them to the test script directly, we would detect errors in tables at run time only. This inevitably increases the testing cycle where tests have to be run several times and test results analyzed. By comparison, the TTCN-3 template approach would detect a number of errors already at compile time when the converted templates are compiled.

Also, if the process is fully automated, the user, in this case the application domain expert, not the programmer, can correct the errors in the Excel table and the TTCN-3 test suite can be automatically re-generated and thus re-compiled without any additional efforts from the programmer. It has to be noted that the original DDT Excel table approach is not completely eliminated because it is still a benefit to have a non-programmer domain expert to code test data. Actually, with this automated Excel table conversion process, the coding effort of the programmer is quasi null. The only task for the TTCN-3 programmer is to direct the application domain expert to the elements in the Excel table which have errors.

A. Extracting TTCN-3 Typing from Excel Spreadsheets

TTCN-3 typing can be derived from the tables quasi automatically. The data can be scanned to determine their type (alphanumeric, numeric or boolean). Also, the field names of a structured data type can be derived from the headings of the columns as for example in the range of row/column B5 to J6 in Figure 11. Complex data types containing fields that are themselves of a structured sub-type can be derived using the indication of Excel spans of a cell, here in cell H5 for the *observation* field that covers the range H6 to J6 for the field names of this sub-type. The generated data types contain comments that indicate their origin on the table to improve traceability.

```
type record StimuliType {
    charstring city,       // cell C5
    charstring country     // cell D5
}

type record ResponseType{
    charstring city,       // cell F5
    charstring country,    // cell G5
    ObservationType observations
                          // cell H5
}

type record ObservationType {
    float temperature,    // cell H6
    charstring sky,       // cell I6
    integer precipitation // cell J6
}
```

Figure 12. TTCN-3 Generated Datatypes Example

B. Generating TTCN-3 templates

Each piece of data of a table is assigned the value of a field of a template. Each row of the table generates separate templates in addition to separate templates for stimuli and response test oracles as follows:

```
template StimuliType ottawa_test_stimuli
:= {
    city := "Ottawa",       // cell C7
    country := "Canada"     // cell D7
}

template ResponseType
            ottawa_test_response := {
    city := "Ottawa",       // cell F7
    country := "Canada",    // cell G7
    observations := {
       temperature := -20,// cell H7
       sky := "cloudy",    // cell I7
       precipitation := 0 // cell J7
}    }
```

Figure 13. TTCN-3 Generated Templates Example

C. *Generating test behavior*

Finally, DDT tables can be interpreted as behavior of the sequential form unless indicated as shown in Figure 14.

```
testcase weather_service_test()
runs on MTCType  system SystemType {
   timer myTimer :=5.0;
   map(mtc:myPort, system:systemPort)
   // row 7
   myPort.send(ottawa_test_stimuli);
   alt {
     [] myPort.receive
            (ottawa_test_response){
          setverdict (pass) }
     [] myPort.receive
         {setverdict(fail)}
     [] myTimer.timeout
         {setverdict(inconc)}
   }
   // row 8
   myPort.send(paris_test_stimuli);
   alt {
      [] myPort.receive
               (paris_test_response){
                setverdict (pass)}
      [] myPort.receive
                {setverdict(fail)}
      [] myTimer.timeout
                {setverdict(inconc)}
   }
   // row 9
   myPort.send(NYC_test_stimuli);
   alt {
      [] myPort.receive
                  (NYC_test_response){
             setverdict (pass) }
      [] myPort.receive
             {setverdict(fail)}
      [] myTimer.timeout
             {setverdict(inconc)}
   }
   unmap(mtc:myPort, system:systemPort);
}
```

Figure 14. TTCN-3 Generated Test Behavior Example

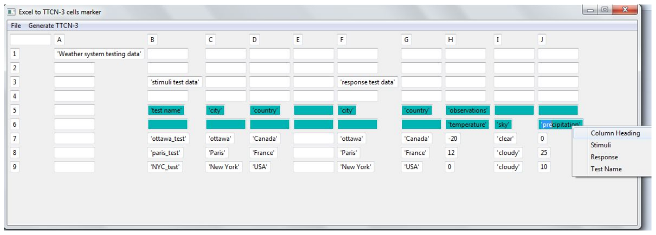
Thus, each row can produce a stimuli being sent and an alternative of a response test oracle with both any value and timeout alternatives. Here again for traceability reasons, we show the row number in the table that corresponds to the test step. If we generate templates with names found in the column with the heading *test* like *ottawa_test*, the table shown in Figure 11 would generate the test behavior shown in Figure 14. The advantage of a TTCN-3 template approach for conducting DDT is that everything is clearly defined and thus is easily traceable at run time without having to go through trace stacks.

IV. TOOL IMPLEMENTATION AND EVALUATION

We have developed and validated these techniques in the testing of an avionics software system. In particular, we implemented a tool to automate the transformation of the tables into a TTCN-3 test suite. As can be seen in Table [15] and, the tool provides an interactive marking mechanism. Each portion of the table can be highlighted and a pull down menu provides categories to choose from in order to indicate how to use the selected portion of cells to the tool. There are three categories of markings required to generate a correct TTCN-3 test suite:

- Delimiting column headings to be used as field names for structured data types code generation
- Delimiting the two sub-tables of stimuli and response test oracles for templates code generation
- Delimiting the test names column if present to generate template names.

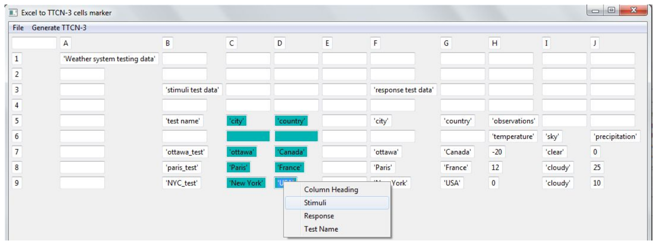
Our marking tool is a mock-up of the Excel spread-sheet in that it shows the rows and columns with the content of the cells as placed in the spread sheet. However, these cells are used for only one purpose, delimiting each zone according to their functionality in the TTCN-3 code generation. No other functionality, like calculations provided by the Excel sheets, can be performed. Also, the code generation makes use mostly of combinations of such markings.



	A	B	C	D	E	F	G	H	I	J
1	'Weather system testing data'									
2										
3		'stimuli test data'				'response test data'				
4										
5		'test name'	'city'	'country'		'city'	'country'	'observations'		
6								'temperature'	'sky'	'precipitation'
7		'ottawa_test'	'ottawa'	'Canada'		'ottawa'	'Canada'	-20	'clear'	0
8		'paris_test'	'Paris'	'France'		'Paris'	'France'	12	'cloudy'	25
9		'NYC_test'	'New York'	'USA'		'New York'	'USA'	0	'cloudy'	10

Figure 15. Delimiting column headings

For example, the marking of column headings shown in Figure 15 is not enough for generating data types because there are two separate groups of data types to be defined, one for stimuli and one for response test oracles. Thus, one must separate the table, shown in Figure 16, and select the portions of the table that belong to either stimuli or responses. This includes the column headings since both data types and test data need to be separated into stimuli and responses.



	A	B	C	D	E	F	G	H	I	J
1	'Weather system testing data'									
2										
3		'stimuli test data'				'response test data'				
4										
5		'test name'	'city'	'country'		'city'	'country'	'observations'		
6								'temperature'	'sky'	'precipitation'
7		'ottawa_test'	'ottawa'	'Canada'		'ottawa'	'Canada'	-20	'clear'	0
8		'paris_test'	'Paris'	'France'		'Paris'	'France'	12	'cloudy'	25
9		'NYC_test'	'New York'	'U		York'	'USA'	0	'cloudy'	10

Figure 16. Delimiting the stimuli sub-table

Manual creation of test scripts in TTCN-3 to execute the tables before the tool was implemented took on average one day per test script. With the tool a complete suite of test scripts was created in one hour. As well, the manual process was error-prone and inconsistent whereas the automated scripts were standardized and needed far less maintenance.

From an implementation point of view, it might have been ideal to use Excel for the highlighting and subsequent export to TTCN-3. However, the export would depend on what commercial tool is available. Thus, we decided on a model to convert the table into a two dimensional array or more precisely different parallel arrays, one containing the data itself and others containing properties such as data types or formatting instructions such as spans that are important to detect sub-structured data types.

Finally, our tool produces only the abstract test suite in TTCN-3. The concrete layer of codecs and communication software specific to the application domain needs to be in place. This is built once (based on TTCN-3 abstract data types) and is reusable by any test suite generated by our tool. This provides a structured approach with a clean separation of concerns (abstract tests vs domain-specific coding/encoding) enabling full re-usability. Traditional unit-testing, by comparison tends to mix test event checking with coding/decoding and communication activity in an ad hoc manner that does not facilitate re-use.

V. CONCLUSION

DDT is an important testing approach for generation and automation of test campaigns. For such benefits to scale it is important that such generation and automation be systematic and strongly-typed. It is also important that the full complexity of parallel test scripts be supported. TTCN-3 provides strong features to support such an approach to TTCN-3 and we have demonstrated how it can be integrated and applied even when the approach to DDT specifications is relatively low-tech and ad hoc through the use of Excel tables. Our approach and tool prototype greatly reduced the manual effort in generating test campaigns, allowed flexible support of Excel for non-technical testers while integrating standardization, strong type and parallel text execution with TTCN-3.

ACKNOWLEDGMENT

The authors would like to thank the Spirent company for providing us the necessary tool *TTworkbench* to carry out this research as well as funding from CRIAQ, MITACS and ISONEO SOLUTIONS for which this research has been conducted.

REFERENCES

[1] Microsoft Excel, 2018. Accessed March 2018 at https://support.office.com/en-us/excel

[2] ETSI ES 201 873-1, The Testing and Test Control Notation version 3 Part 1: TTCN-3 Core Language, May 2017. Accessed March 2018 at http://www.etsi.org/deliver/etsi_es/201800_201899/20187301/04.09.01_60/es_20187301v040901p.pdf

[3] P. Shinde and A. Sathe, Data-Driven Software Testing for Agile Development, in PhUSE 2011, Brighton, United Kingdom, June, 2011. Accesed March 2018 at https://www.lexjansen.com/phuse/2011/ts/TS08.pdf

[4] Microsoft Corporation, How To: Create a Data-Driven Unit Test in Visual Studio, 2015. Accessed March 2018 at https://msdn.microsoft.com/en-us/library/ms182527.aspx

[5] StackExchange, What are some good approaches to separating test data from test scripts, 2013. Accessed March 2018 at https://sqa.stackexchange.com/questions/6678/what-are-some-good-approaches-to-separating-test-data-from-test-scripts

[6] Smartbear, Introduction to Data-Driven Testing, 2018. Accessed March 2018 at https://smartbear.com/learn/automated-testing/introduction-to-data-driven-testing

[7] I. Schieferdecker and B. Stepien, Automated Testing of XML/SOAP Based Web Services. In: Irmscher K., Fähnrich KP. (eds) Kommunikation in Verteilten Systemen (KiVS). Informatik aktuell. Springer, Berlin, Heidelberg, 2003. https://doi.org/10.1007/978-3-642-55569-5_4

[8] B. Stepien and L. Peyton, Integration Testing of Web Applications and Databases Using TTCN-3. In: Babin G., Kropf P., Weiss M. (eds) E-Technologies: Innovation in an Open World. MCETECH 2009. Lecture Notes in Business Information Processing, vol 26. Springer, Berlin, Heidelberg, 2009. https://doi.org/10.1007/978-3-642-01187-0_26

[9] Chandrapabha, A. Kumar and S. Saxena, Data Driven Testing Framework using Selenium WebDriver, in International Journal of Computer Applications (0975-8887) vol. 118-No. 18, May 2015. Accessed March 2018 at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.9076&rep=rep1&type=pdf

[10] E. G. Gomez, M. Casado, M. Stanka and S. Korner, Automated Regression Testing of Complex Mission Control Applications, SpaceOps 2010, Huntsville, Alabama, April 2010. Accessed March 2018 at https://arc.aiaa.org/doi/pdf/10.2514/6.2010-2289

[11] D. Cheney, Writing Table Driven Tests in Go, 2013. Accessed March 2018 at https://dave.cheney.net/2013/06/09/writing-table-driven-tests-in-go

[12] N. Daley, D. Hoffman and P. Strooper, A framework for table driven testing of Java classes. Softw: Pract. Exper., 32: 465–493, 2002. doi:10.1002/spe.452. Accessed March 2018 at http://onlinelibrary.wiley.com/doi/10.1002/spe.452/full

[13] J. S. Schaefer, Systems and methods for table driven automation testing of software programs, Capital One Financial Corporation, 2006. U.S. Patent 6,993,748. Accessed March 2018 at https://patents.google.com/patent/US6993748B2/en

[14] J. J. Haswell, R. J. Young, and K. Schramm, System, method and article of manufacture for a table-driven automated scripting

architecture, Accenture Llp, 2002. U.S. Patent 6,502,102. Accessed March 2018 at https://patents.google.com/patent/US6502102B1/en

[15] J. Zander, Z. R. Dai, I. Schieferdecker and G. Din, From U2TP Models to Executable Tests with TTCN-3 - An Approach to Model Driven Testing -. In: Khendek F., Dssouli R. (eds) Testing of Communicating Systems. TestCom 2005. Lecture Notes in Computer Science, vol 3502. Springer, Berlin, Heidelberg, 2005. https://doi.org/10.1007/11430230_20

[16] G. Adamis, A. Wu-Hen-Chang, G. Németh, L. Eros and G. Kovacs, Data Flow Testing in TTCN-3 with a relational Database Schema, in International SDL Forum, SDL 2013: Model-Driven Dependability Engineering pp 1-18, Springer Verlag

The Antecedents and Feedback Loops Contributing to Trust in Agile Scrum Teams

Trish O’Connell
School of Science & Computing
Galway-Mayo Institute of Technology
Galway, Ireland
trish.oconnell@gmit.ie

Owen Molloy
Dept. of Information Technology
National University of Ireland
Galway, Ireland
owen.molloy@nuigalway.ie

***Abstract*— Scrum has become the dominant Agile way of developing software products and systems. To ensure the team achieves the goals of the Sprint, the team needs to collaborate effectively and share knowledge optimally. To do this, McHugh, Conboy and Lang, amongst others, have claimed that trust is “of increased importance” to the Agile Scrum team. This paper describes the contributions to the academic discourse on trust and subsequently hypothesizes how these may apply to the Scrum team. Whilst some of the antecedents are straightforward contributors to building trust, others may function as reinforcing feedback loops. A preliminary conceptual model is presented, and further research is underway to refine and validate the model.**

Keywords-Agile; Scrum; Team; Trust; Collaboration; Knowledge-sharing.

I. INTRODUCTION

Software development has always been a task-oriented activity. With the advent of Agile, it has become a task-oriented, social activity. Moe, Dingsøyr and Dybå state, “the basic work unit in innovative software organizations is the team rather than the individual [1].” In Scrum (an Agile framework for managing the development process often referred to as a methodology), software development can be considered as a collective team effort, where teamwork requires cooperation and therefore, social interaction. A fundamental characteristic of a good team is that the team members collaborate well. The co-creators of the Agile Manifesto [2] referred to the fact that Agile teams are characterized by “intense collaboration” where collaboration refers to “actively working together to deliver a work product or make a decision.” It is through collaboration and knowledge-sharing that software development tasks may be accomplished successfully. Nerur concurs, “A cooperative social process characterized by communication and collaboration between a community of members who value and trust each other is critical for the success of agile methodologies [3].”

Whereas cooperation between team members involves the “smooth transfer of work in progress, work products, and information from one member to another [4],” collaboration, by contrast, “elevates groups beyond cooperation, adding an essential ingredient for emergent, innovative, and creative thinking [4].”

A. *Collaboration*

To increase collaboration and facilitate knowledge sharing, Agile methods such as Scrum rely heavily on face-to-face communication and a high degree of interaction between the team. The Agile Manifesto advocates “Individuals and interactions over processes and tools [2].” Highsmith states “Most traditional ‘methodologies’ place 80 percent of their emphasis on processes, activities, tools, roles, and documents. Agile approaches place 80 percent of their emphasis on ecosystems—people, personalities, collaboration, conversations, and relationships [5].”

Whilst the Agile software development framework referred to as XP promotes developers working together in a technique known as ‘pair programming’ to achieve this face-to-face communication, the Scrum approach relies on the three key practices which McHugh, Conboy and Lang describe as “sprint/ iteration planning, daily stand-up, and sprint/iteration retrospective [6].”

The iteration planning session is where the team collectively plans and agrees on what will be delivered at the end of the Sprint.

The daily stand-up is a team status meeting where team members describe progress and obstacles (if any) to meeting commitments.

The sprint retrospective is effectively a post-partum review of the sprint that has been completed. It is supposed to allow the team to collectively review what went well and what did not, during the sprint. It should serve as the baseline for improvement. [7]

Ghobadi and Mathiassen posit, “Software development is a collaborative process where success depends on effective knowledge sharing [8].”

B. *Knowledge sharing*

Liu and Phillips expound that trust and collaboration are “essential for effective knowledge sharing to occur [9].” It is essential that the Scrum team shares knowledge during all phases of the Sprint. Park and Lee postulate, “The time spent on problem solving can be reduced significantly because the project participants' benefit from the shared and accumulated knowledge [10].” The three Agile practices which are used in Scrum all involve communication and sharing information, to varying degrees. Following their research study McHugh, Conboy and Lang conclude “All three practices provide an open forum for sharing knowledge and obtaining feedback [7].” The purpose of knowledge sharing in Scrum is that it moves the development process along. The team members do not need to pause in their development efforts due to obstacles. As Park and Lee explain, “more frequent communication creates opportunities to develop and enhance knowledge sharing. 10]” This “frequent communication” is the hallmark of Agile with the Agile Manifesto recommending. “Individuals and interactions over processes and tools [2].”

For collaboration to be successful a climate of trust needs to exist in the team Ceschi, Sillitti, Succi, and De Panfilis, highlight the fact that “Knowledge sharing through communication requires a high level of mutual trust among team members and frequent interactions [11].” Indeed, it may be argued that trust is a vital component, and “important supporting mechanism of teamwork [12],” according to Weimar, Nugroho, Visser, Plaat and Goudbeek.

Many authors have cited trust as being important to collaboration, with Mishra claiming, “trust has been found to be a critical factor facilitating collaboration [13].”

Park and Lee also see trust as imperative for knowledge transfer and successful team performance asserting, “the sharing of knowledge in an IS project has become a requirement for the completion of a successful IS project [10].”

Whilst much has been written about the importance and need for trust in Agile teams, e.g. Mayer, Davis and Schoorman posit, “The emergence of self-directed teams and a reliance on empowered workers greatly increase the importance of the concept of trust, as control mechanisms are reduced or removed, and interaction increases [14],” there has been little to no direct research into trust in Agile teams. As, McHugh, Conboy and Lang state, “Agile methods have been the subject of much research, as has trust, but the impact of trust on agile teams has not [6].”

This paper attempts to fill this void in the trust construct as applied to Agile Scrum teams.

The remainder of this paper is organized as follows: Section II of this paper briefly considers team formation and the development of interpersonal trust. Section III examines the notion of trust as presented in the academic discourse. Section IV addresses the application to the Scrum team and presents a conceptual model of how trust can be depicted in a Scrum team setting. Finally, the paper concludes with a brief discussion and plans for future work.

II. TEAM FORMATION

Depending on the context, there are many characterizations of trust. In terms of a team, the most crucial type of trust is likely to be interpersonal which facilitates and fosters collaboration and knowledge sharing between team members. Rotter defines interpersonal trust as, “an expectancy held by an individual or a group that the word, promise, verbal or written statement of another individual or group can be relied upon [15].” From a Scrum team perspective, it is imperative that a team member fulfils his commitment which is made at the Scrum Daily standup meeting. Another oft-quoted definition of trust is attributed to Mishra, “Trust is one party's willingness to be vulnerable to another party based on the belief that the latter party is 1) competent, 2) open, 3) concerned, and 4) reliable [13].”

Interpersonal trust does not tend to just ‘happen’ in a team. The preeminent treatise on team formation was proposed by Tuckman in 1965. He proposed that teams progress through four distinct phases: “Forming, Storming, Norming and Performing [16].”

“Forming” is the phase where team members are first brought together and whilst they may agree on goals they are predominantly working as individuals with no sense of the common purpose. Individuals assess each other’s boundaries in what Tuckman refers to as “testing”. In addition, Tuckman expounds, “Coincident with testing in the interpersonal realm is the establishment of dependency relationships with leaders, other group members, or preexisting standards [16].”

The second developmental phase in team development is termed “Storming” and it is often characterized by “conflict and polarization around interpersonal issues, with concomitant emotional responding in the task sphere. These behaviors serve as resistance to group influence and task requirements [16].” At this stage, trust is predominantly invested in the team leader.

On exiting the preceding phase, the team comes to the realization that they have a common goal. Tuckman describes how “in-group feeling, and cohesiveness develop, new standards evolve, and new roles are adopted. In the task realm, intimate, personal opinions are expressed [16].” At this stage, referred to as “Norming,” interpersonal trust is beginning to develop. Once the team norms are understood the team begins to develop trust in the process.

“Performing” is the final and most crucial stage for the team. As Tuckman explains, “group energy is channeled into the task. Structural issues have been resolved, and structure can now become supportive of task performance [16].” At this stage, the team members should be sufficiently comfortable with each other that a degree of interpersonal trust is established.

In Scrum, teams are often pulled together based on the projects requirements, the domain expertise needed, the availability and experience of personnel. Scrum teams will inevitably progress through the four phases as described above.

Scrum teams are self-managing. Moe, Dingsøyr and Dybå describe how “a Scrum team is given significant authority and responsibility for many aspects of their work, such as planning, scheduling, assigning tasks to members, and making decisions [1].”

It would not be possible for the team to function effectively, in pursuance of the above, without trust.

III. TRUST IN THE ACADEMIC DISCOURSE

Whilst many have written about trust it would still appear to be confusing, predominantly because much of the research has been context specific, ranging from sociological (Simmel [17], Luhmann [18], Barber [19], Lewis and Weigert [20], Mayer et al. [14], Dirks and Ferrin [21]) to psychological (Rotter [15], Rempel, Holmes and Zanna [22], McKnight and Chervany [23]). Confusing, also, because for there to be trust between team members there must be conditions, which facilitate this trust to grow.

Some authors refer to these as the *antecedents* of trust (Costa, [24]), but trust also gives rise to consequences. In this authors opinion, some of the consequences also function as reinforcing feedback mechanisms for enhancing trust in a team.

Whilst Simmel [17] referred to trust as a mysterious “faith” of man in man. Deutsch [25] equated trust to a reciprocal, cooperative, relationship between people who make the decision to trust. By this he means that a person will meet the expectations of another, and in return, expect his/her expectations to also be fulfilled. Furthermore, Deutsch expounds that fulfilling another’s expectations also involves the notion of competence. There is nothing to be gained from trusting someone to do something in which they have no competence to succeed.

Once a degree of mutual trust has been established, knowledge sharing and collaboration should follow. Zand concurs that persons who trust one another “will provide relevant, comprehensive, accurate, and timely information, and thereby contribute realistic data for problem-solving efforts [26].”

Granovetter [27] refers to relationships between two individuals as “dyadic ties” and defines the strength of a tie as “a (probably linear) combination of the amount of time, the emotional intensity, the intimacy (mutual confiding), and the reciprocal services which characterize the tie. Each of these is somewhat independent of the other, though the set is obviously highly intracorrelated [27].” Gabarro highlighted the importance of “openness about task problems or task related issues [28]” as being highly influential in the development of trust. Moreover, Gabarro echoes Deutsch [25] in that he posits “competence, reliability and openness more than compensated for a lack of initial liking [28].”

Furthermore, Gabarro listed integrity and judgement as being equally as important as competence in the perception one forms of another when considering whether to trust them [28].

Working from the premise that one trusts people with whom one is familiar, Luhmann [18] argued that familiarity serves as the “prerequisite for trust.” Another train of thought expounded by Luhmann concerns the motivations of the trustee in the trust situation. It seems to be the first mention of a rational calculation on which to base trust since Luhmann refers to “motivational structures” which can be focused on when we do not “know the future actions of the other party[18].” He postulates that “on the one hand he (the trustor) will find it worthwhile to ask himself with what prospects for gain and loss his partner (the trustee) can reckon, if he is trusted[18].” This harks back to Deutsch who referred to “behavior which the individual perceives to have greater negative motivational consequences if the expectation is not confirmed than positive motivational consequences if it is confirmed [25].”

Ultimately, Luhmann acknowledges the situation in which trust is required and he further expounds on the role of uncertainty and ambiguity in building trust. Undoubtedly, this encompasses the realm of software development.

> “There has to be defined some situation in which the person trusting is dependent on his partner; otherwise the problem doesn’t arise. His behaviour must then commit him to the situation and make him run the risk of his trust being betrayed. In other words, he must invest in... a ‘risky investment’. One fundamental condition is that it must be possible for the partner to abuse the trust; indeed, it must not merely be possible for him to do so but he must also have a considerable interest in doing so. It must not be that he will toe the line on his own account – in the light of his interests. In his subsequent behaviour the trust put in him must be honoured and his own interests put to one side [18].”

From this description, it is evident that trust occurs when there is an element of uncertainty in the relationship or task. The trust process as described by Luhmann evidences a two-way street in terms of firstly the trustor must confer trust and then the trustee accepts and fulfils the trust proposition. Luhmann concludes that the process “demands mutual commitment and can only be put to the test by both sides becoming involved in it, in a fixed order, first the truster and then the trustee [18].”

Barber reiterates Deutsch’s [25] position on the need for competence but goes further by including an expectation that “partners in interaction will carry out their fiduciary obligations and responsibilities [19].”

In 1991, Butler postulated that trust is "multidimensional as a construct as well as being activated by a multidimensional set of conditions [29]." By reviewing the work of those that had contributed to the academic discourse on trust, Butler was able to develop and publish his content theory "consisting of a multidimensional set of conditions that activate and sustain trust in a specific person [29]." In 1994, Butler and Cantrell ranked the conditions of trust in the following order of importance: "*competence* (technical and interpersonal skills required for one's job), *integrity* (honesty and truthfulness), *consistency* (reliability, predictability, and good judgement), *loyalty* (having motives for protecting and making the target person look good, and *openness* (freely sharing ideas and information) [30]." Further research led to the identification of ten categories and from these ten conditions of trust were inferred: "*availability*, *competence*, *consistency*, *discreetness*, *fairness*, *integrity*, *loyalty*, *openness*, *promise fulfilment*, and *receptivity.[30]"* As Butler commented, "the inferred conditions were conceptually similar to most of the trust conditions identified by Jennings (1971) and Gabarro (1978) [29]." However, it should be noted that whilst *promise fulfilment*, *fairness* and *receptivity* were not specifically listed by the authors above they arose from either inferred/implied comments from respondents or from direct mention.

Building on the work of Simmel [17], Luhmann [18], and Barber [19], Lewis and Weigert present trust as "a property of collective unit, not of isolated individuals [20]." These authors perceive trust as an attribute which is "applicable to the relations among people." In this sense the academic discourse is moving closer to the social relationships present in teams.

Similar to Butler [29], Lewis and Weigert acknowledge the "multifaceted character" of trust. However, they differ insofar as they describe the facets as "distinct cognitive, emotional, and behavioural dimensions that are merged into a unitary social experience [20]." They explain the cognitive aspect of trust as discriminating "among persons and institutions that are trustworthy, distrusted, and unknown. In this sense, we cognitively choose whom we will trust in which respects and under which circumstances and we base the choice on what we take to be 'good reasons', constituting evidence of trustworthiness [20]."

Deutsch hypothesizes that an increase in communication will increase 'trust' and also that "we can expect that there will be some tendency for trustworthiness to increase with trust [25]."

Gabarro deviates from the academic discourse by theorizing that trust may be "better understood as a result rather than a precondition of cooperation [28]." Trust, according to Gabarro [28] would thus exist in groups simply *because* the group is successful and able to cooperate. It should be noted that Gabarro lays the foundation for much of the theory of trust that comes next when he states, "There is a sense in which trust may be a by-product, typically of familiarity and friendship, both of which imply that those involved have some knowledge of each other and some respect for each other's welfare [28]."

Shapiro, Sheppard and Cheraskin argue that "the benefits associated with establishing trust in the right conditions should result in increased quality of output, greater efficiency of process, more flexibility, and an enhanced strategic focus [31]." The authors promulgate three bases of trust as follows: deterrence based trust, knowledge based trust and identification-based trust. In situations where monitoring and control are used to ensure compliance, these form the basis of deterrence based trust. Knowledge based trust is also based on Deutsch's [25] belief that trust is the underpinning or foundation of cooperative behaviour. Shapiro, Sheppard and Cheraskin postulate that if we know a person and how they will act or respond we have an element of predictability upon which we have a "basis of trust" since as the authors state "At its core, trust is simply dependability. The benefits of dependability are reduced uncertainty and less need for contingent planning [31]." Unsurprisingly, Shapiro et al. advocate regular communication as a method of achieving knowledge-based trust.

The third base of trust according to the authors is identification based trust. This is explained as "the highest order of trust assumes that one party has fully internalized the other's preferences [31]." It is often mentioned in the literature on successful teams that having a shared goal or vision is crucial to success.

Lewicki and Bunker expand on the theories of Shapiro et al. [26] by positing that "the three types of trust are probably linked and sequential [32]." Whereas Shapiro et al. identify the three types of trust as separate and independent. Lewicki and Bunker propose that this linkage is sequential and iterative, "achievement of trust at one level enables the development of trust at the next level [32]."

Additionally, Lewicki and Bunker describe the process of trust beginning with calculus based trust. The authors describe how calculus based trust is arrived at in a stepwise process with each trusting endeavour being used as the basis for the next level. In this sense it is described as "tactical climbing [32]." Once a certain level of understanding has been achieved, it is possibly for knowledge based trust to evolve in that, having 'tested the waters' so to speak, the trustor has knowledge of the trustee and can reasonably predict their behaviour vis à vis a given expectation. Once this level of trust has been attained, slight breaches of trust may even be tolerated. Finally, the highest level of trust, identification based trust, occurs when the parties involved share the same wants and needs, what Lewicki and Bunker refer to as a "collective identity develops [32]." At this point a healthy degree of synergy has developed which facilitates cooperative and productive teamwork.

The model of organizational trust proposed by Mayer, Davis and Schoorman in 1995 is one of the most cited models of trust in the literature. In their research, the authors examined "why a trustor would trust a trustee." The authors view trust as "a trait that leads to a generalized expectation about the trustworthiness of others [14]." Mayer et al. refer to this trait as "propensity to trust [14]." Continuing in this vein the authors' state "People differ in their inherent propensity to trust. Propensity might be thought of as the general willingness to trust others. Propensity will influence how much trust one has for a trustee prior to data on that particular party being available [14]." Thus, whilst Deutsch [25], Lewicki and Bunker [32] and Shapiro et al. [31] argue for the existence of a calculated decision to trust Mayer et al. [14] introduce the concept of a propensity to trust which the trustor may or may not have. According to Mayer et al. [14] the propensity to trust cannot be taken in isolation. As if describing two sides of the same coin, the authors also argue for the trustee to possess the characteristic of trustworthiness. The trustee must show themselves as meriting or warranting trust being placed in them. Mayer et al. describe three characteristics of a trustee as: "ability, benevolence and integrity [14]." Ability has already been introduced by Deutsch [25] but this time Mayer et al. argue that an individual may not have competence in all areas but often a specific area. In addition to this Mayer et al. introduced the ideas of a "willingness to be vulnerable to the actions of another [14]" and furthermore a trustee must have benevolence towards the individual who is trusting. The Mayer et al. model of trust is one of the first that clearly discriminates between trust and its antecedents.

However, the authors themselves note that this particular model is limited to a unidirectional treatment of trust between a trustor and a trustee. Consequently, there is no mention of reciprocity in this model as it was not explicitly designed to examine trust relationships in a team context.

Watson [33] describes McAllister's work as "influential." His work on trust recognises the importance of "developing and maintaining trust relationships [34]." Basing his theories on the work of the sociological researchers on trust (Barber, [19]; Lewis and Weigert, [20]; Luhmann, [18]; Shapiro et al. [31]; Mayer et al. [14];) McAllister distinguishes two principal forms of interpersonal trust "cognition-based trust, grounded in individual beliefs about peer reliability and dependability, and affect-based trust, grounded in reciprocated interpersonal care and concern [34]." The introduction of an affective or emotional component to the trust model proposed by McAllister was ground-breaking.

Whilst Mayer et al. see trust as unidimensional and largely cognitive, based in so far as they advocate that one would judge the ability, benevolence and integrity of the person upon whom they would confer trust. McAllister, by contrast, whilst conceding the cognitive aspect and its antecedents argues also for an affective basis on which to confer trust stating "emotional ties linking individuals can provide the basis for trust[34]." This reiterates Lewis and Weigert in their conclusion that trust is multifaceted with "distinct cognitive, emotional and behavioural dimensions that are merged into a unitary social experience [20]." Similarly, Johnson-George and Swap [35] referred to two dimensions of trust "Reliableness" and "Emotional Trust."

From having worked and led teams it is the author's opinion that there is merit in all of the antecedents as listed. The next section reviews these antecedents with specific focus on Agile Scrum teams. Building on the work of Gabarro [28] it is hypothesized that the antecedents of trust effectively form a reinforcing feedback loop.

IV. SCRUM TEAM TRUST

Whilst the antecedents of trust have been described in Section III, it is somewhat surprising that there is a dearth of research in the domain of Agile Scrum teams. McHugh et al. clarify, "While there have been many studies of trust in software development teams few have examined trust in an agile context [6]." Although many authors cite trust as necessary, Moe et al. explain succinctly the rationale for this "without sufficient trust, team members will expend time and energy protecting, checking, and inspecting each other as opposed to collaborating to provide value-added ideas [1]." What follows attempts to explain how the antecedents of trust might function in a Scrum team. This is shown in Figure 1. At this point in the author's research, Figure 1 represents a first stage conceptual model of trust in the Scrum team.

A. *Perception*

In a team setting trust is initially most likely to be based on perception. How a new team member comports himself on day one will lead the rest of the team to make a calculated judgement based largely on observation. What the new team member says and also how he says it is all used to form a perception and consequently an initial judgement of the individual. This initial phase closely resembles Tuckman's seminal work on stages of group development. Tuckman describes how in the 'Forming' phase members engage in "ritual sniffing" in order to get to know a new member and make a preliminary determination of their credibility [16].

B. *Reputation*

The new team member's reputation, if this is known to the team, will also be brought to bear in forming an opinion as to whether the individual can be trusted. Stemming from this a degree of what Lewicki and Bunker [32] refer to as "calculus based trust" comes into play. This type of trust is predominantly what Lewicki and Bunker [32] describe as "deterrence based trust" in which the team member is effectively being evaluated to ascertain if they will do what they say they will do. The authors argue that an individual will comply not only because of the fear of "punishment for violating the trust" but also due to the "rewards to be derived from preserving it [32]." Acceptance or rejection by the Scrum team would be of paramount importance to a new team member.

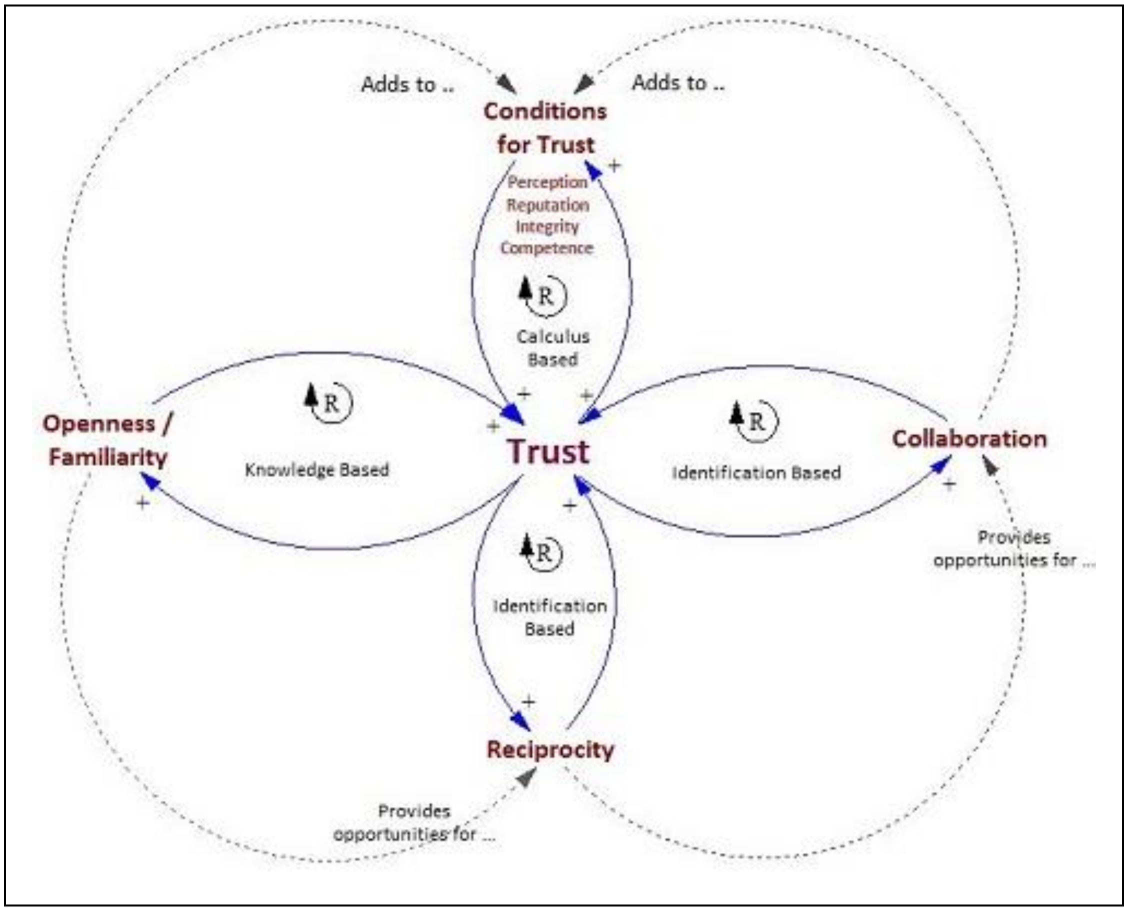


Figure 1. Conceptual Model of Trust

C. Integrity

As time passes, the team member’s credibility is tested and retested during Sprints. If the team member keeps his commitments his integrity is acknowledged by the team. He becomes predictable insofar as he is known to keep his word on what he says he can deliver [25].

D. Competence

It should be noted, however that a team member’s technical competence in their team role is crucial to the Scrum team’s performance and success [19]. A competent Scrum team will succeed in delivering the Sprint backlog.

As time passes and the new team member is proving/has proved himself as being trustworthy it is envisaged that the first reinforcing feedback loop becomes operational. A team member who has proven himself to act with competence and integrity will find that both his reputation and his team mates’ perception of him and his ability to deliver is enhanced and they he trusted more than he was initially.

By the time the team has progressed through Tuckman’s stage of ‘storming, norming and performing’ the Scrum team has hopefully learned to work well together.

E. Familiarity

Once team members have developed a good rapport, the team can move beyond calculus based trust to where they have developed what might be thought of as an emotional bond between each other. Santos et al. explain, “Agile values and principles foster changes in team members’ attitudes and strengthen their relationships. These changes happen as a result of greater trust and better communication and transparency in the relationships among team members [36].” Ideas may be shared without fear of ridicule and the team should be set for a degree of knowledge sharing and collaboration.

Moving from working cooperatively to collaboratively is a key milestone for a Scrum team. Scrum teams work closely together and are frequently co-located. Given the emotional intensity involved in keeping commitments, delivering on time and helping each other to deliver artefacts from the Product backlog it is unsurprising that strong dyadic ties begin to develop among the team [27]. Team members become interdependent in order to realize the goals of the Sprint and rely on each other to a high degree. The familiarity that results reinforces the trust within the team.

F. Openness

As a consequence of this familiarity, Scrum team members tend to be open and act with integrity in their dealings with each other. This level of "trusting behaviour invites the attributes of trustworthiness [29]" according to Butler.

Allied to this level of familiarity and openness it is unsurprising that an affective bond begins to develop between the Scrum team members. They begin to know each other, and a degree of predictability ensues. This "knowledge based trust [32]" is the core of the second reinforcing feedback loop. As team members come to know each other better, trust is enhanced.

The benefit of moving into this phase is postulated by Shapiro and Sheppard as the "primary advantage of knowing that a partner is reliable, i.e., will keep his/her word, is that it shifts one's focus from monitoring to problem solving [31]."

G. Reciprocity

DeVries, Van Den Hooff and Ridder describe "a cycle of reciprocity, in which team members are more likely to exchange (i.e., both donate and collect) knowledge with each other [37]." As the team bonds become deeper, it would be expected that a Scrum team member would not feel exposed in asking for assistance on an aspect of the development with which difficulty was being experienced. In similar vein the team member who receives help would most likely be happy to help the individual who had given help. As a highly functioning team it is the team goal that is of paramount importance and the degree of benevolence (Mayer et al. [14]) that team members feel towards each other would ensure that help is both given and received in equal measure as required.

In this stage yet more positive reinforcing feedback occurs. The team members can set aside the cognitive approach to trust and opt rather for an emotional connection between each other. McAllister, [34], Martin, [38], Lewicki and Wiethoff [39] have referred to this as "Identification based trust."

Once the team has moved into the 'Identification based trust' realm the team members fully identify with each other and with the common goals of the Sprint.

V. CONCLUSION AND FUTURE WORK

The academic literature on trust is vast as many studies have examined it from a variety of contexts. This paper has presented the findings of the main contributors to the academic discourse on trust and has attempted to apply their contributions to the Agile Scrum team in the form of a preliminary conceptual model.

The next step in the research is to ascertain using a constructivist grounded theory methodology if this hypothesis is, indeed, valid or whether there are other elements of the trust equation which lead to successful Sprints.

REFERENCES

[1] N. B. Moe, T. Dingsøyr, and T. Dybå, "Overcoming barriers to self-management in software teams," *IEEE software*, vol. *26*, no.6, pp.20-26, 2009.

[2] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, and J. Kern, Manifesto for agile software development, 2001.

[3] S. Nerur, R. Mahapatra, and G. Mangalaraj, "Challenges of migrating to agile methodologies," Communications of the ACM, vol. *48* no.5, pp.72-78, 2005.

[4] K. Collier, Agile analytics: A value-driven approach to business intelligence and data warehousing. Addison-Wesley, 2012.

[5] J. Highsmith, Adaptive Software Development Ecosystems. Boston, MA: Pearson Education Inc, pp. 244-245, 2002.

[6] O. McHugh, K. Conboy and M. Lang, "Agile practices: The impact on trust in software project teams," IEEE Software, vol. *29*, no. 3, pp. 71-76, 2012.

[7] O. McHugh, K. Conboy, and M. Lang, "Using agile practices to influence motivation within IT project teams," Scandinavian Journal of Information Systems vol. 23, no. 2, pp. 85–110 (Special Issue on IT Project Management), 2012.

[8] S. Ghobadi, and L. Mathiassen, "Perceived barriers to effective knowledge sharing in agile software teams," Information Systems Journal, vol. 2, no. 2, pp.95-125, 2016.

[9] Y. Liu, and J. S. Phillips, "Examining the antecedents of knowledge sharing in facilitating team innovativeness from a multilevel perspective." International Journal of Information Management, vol. 31, no. 1, pp. 44-52, 2011.

[10] J. G. Park**,** and J. Lee, "Knowledge sharing in information systems development projects: Explicating the role of dependence and trust." International Journal of Project Management, vol.*32*, no. 1, pp. 153-165, 2014.

[11] M. Ceschi, A. Sillitti., G. Succi, and S. De Panfilis, "Project management in plan-based and agile companies," IEEE software, vol. *22,* no.3, pp. 21-27, 2005.

[12] E. Weimar, A. Nugroho, J. Visser, A. Plaat, M. Goudbeek, and A. P. Schouten, "The Influence of Teamwork Quality on Software Team Performance," Proc. 17th International Conference on Evaluation and Assessment, 2017.

[13] A. Mishra,."Organizational response to crisis: The centrality of trust," In. R.Kramer, and T. Tyler, (Eds.), Trust in organizations: Frontiers of theory and research, pp. 261-287, 1996.

[14] R. C. Mayer, J. H. Davis. and F. D. Schoorman, The Academy of Management Review, Vol. 20, No. 3, pp. 709-734,1995.

[15] J. B. Rotter, "A new scale for the measurement of interpersonal trust," Journal of Personality, vol. 35, no. 4, pp. 651-665, 1967.

[16] B. W. Tuckman, "Developmental sequence in small groups," Psychological bulletin, vol. 63, no. 6, pp. 384-399, 1965.

[17] G. Simmel**,** "The Sociology of Georg Simmel". K.H. Wolff (trans., ed. and intr.). New York: Free Press, 1950.

[18] N. Luhmann, “Familiarity, Confidence, Trust: Problems and Alternatives”, in D. Gambetta, (ed.) Trust: Making and Breaking Cooperative Relations, electronic edition, Department of Sociology, University of Oxford, chapter 6, pp. 94-107, 2000.

[19] B. Barber, The logic and limits of Trust, Rutgers University press, 1983.

[20] J. Lewis, and A. Weigert,, “Trust As a Social Reality,” Social Forces, Vol. 63, No. 4, pp. 967-985, 1985.

[21] K. Dirks, and D. Ferrin, “The Role of Trust in Organizational Settings,” Organization Science, vol. 12, no. 4, pp. 450-467, 2001.

[22] J. K. Rempel, J. G. Holmes, and M. P. Zanna, “Trust in close relationships,” Journal of personality and social psychology, vol. 49, no.1, pp. 95-98, 1985.

[23] D. H. McKnight, and N. L. Chervany, “The Meanings of Trust,” Technical Report MISRC Working Paper Series 96-04, University of Minnesota, Management Information Systems Research Center, 1996.

[24] A. C. Costa, ″Understanding the nature and the antecedents of trust within work teams″, in B. Noteboom, (Ed.), The Trust Process in Organizations, Cheltenham und Northhampton, pp. 105 - 24, 2003.

[25] M. Deutsch, “Trust and suspicion,” Journal of conflict resolution, pp. 265-279, 1958.

[26] D. E. Zand,“Trust and Managerial Problem Solving,” Administrative Science Quarterly, pp. 229-239, 1972.

[27] M. S. Granovetter, “The Strength of Weak Ties,” American Journal of Sociology, pp. 1360-1380, 1973.

[28] J. J. Gabarro, “The Development of Trust, Influence, and Expectations,” Interpersonal behavior: Communication and Understanding in Relationships, edited by Anthony Athos and John J. Gabarro. Englewood Cliffs: Prentice Hall, pp.290, 303,1978.

[29] J. K. Butler, “Toward Understanding and Measuring Conditions of Trust: Evolution of a conditions of trust inventory,” Journal of Management, vol. 17, pp. 643-663, 1991.

[30] J. K. Butler Jr, and R. S. Cantrell, “A behavioral decision theory approach to modeling dyadic trust in superiors and subordinates,” Psychological reports, vol. *55*, no. 1, pp. 19-28, 1984.

[31] D. L. Shapiro, B. H. Sheppard, and L. Cheraskin, “Business on a handshake,” Negotiation Journal, vol. 8, no. 4, pp. 365-377, 1992.

[32] R. J. Lewicki, and B. B. Bunker, “Trust in Relationships, Proc. 19th Australian Conference on Software Engineering, IEEE 2008 pp. 76-84, 1995.

[33] M. L. Watson, “Can there be just one trust? A cross-disciplinary identification of trust definitions and measurement,” The Institute for Public Relations, Gainesville, Florida, pp. 1-25, 2005.

[34] D. J. McAllister, Affect-and cognition-based trust as foundations for interpersonal cooperation in organizations. Academy of management journal, vol. 38, no. 1, pp. 24-59, 1995.

[35] C. Johnson-George, and W. C. Swap, Measurement of specific interpersonal trust: Construction and validation of a scale to assess trust in a specific other. Journal of personality and social psychology, vol. *43*, no. 6), p. 130,1982.

[36] V. Santos, A. Goldman, and C. R. De Souza, “Fostering effective inter-team knowledge sharing in agile software development,” Empirical Software Engineering, vol. 20, no. 4, pp.1006-1051, 2015.

[37] R. E. De Vries, B. Van den Hooff, and J. A. Ridder, Explaining knowledge sharing: The role of team communication styles, job satisfaction, and performance beliefs. Communication research, vol. *33* no. 2, pp. 115-135, 2006.

[38] D. Martin, “Towards a model of trust,” Journal of Business Strategy, vol. 35, no.4, pp. 45–51, 2014.

[39] R. J. Lewicki, and C. Wiethoff, “Trust, trust development, and trust repair,” *The handbook of conflict resolution: Theory and practice*, vol. *1* no. 1, pp.86-107, 2000.

Client-Side XSS Filtering in Firefox

Andreas Vikne and Pål Ellingsen
Department of Computing, Mathematics and Physics
Western Norway University of Applied Sciences
Bergen, Norway
Email: andreas.svardal.vikne@stud.hvl.no, pal.ellingsen@hvl.no

Abstract—**One of the most dominant threats against Web applications is the class of script injection attacks, also called cross-site scripting. This class of attacks affects the client-side of a Web application, and is a critical vulnerability that is difficult to both detect and remediate for website owners, often leading to insufficient server-side protection, which is why the end-users need an extra layer of protection at the client-side, utilizing the defense in depth principle. In this paper, a client-side filter for Mozilla Firefox is presented, with the goal of protecting against reflected cross-site scripting attacks while maintaining high performance. By conducting tests on our implemented solution, although still in an early phase, we can conclude that our filter does provide more protection than the original Firefox browser, at the same time achieving high performance, which with further development would become an effective option for end-users of Web applications to protect themselves against reflected cross-site scripting attacks.**

Keywords–cross-site scripting; client-side filtering; Web browser protection.

I. Introduction

Cross-site scripting has for long been among the top threats against Internet security as defined in the Open Web Application Security Project (OWASP) Top 10 report, which presents the 10 most common security vulnerabilities found in Web applications [1]. Even if cross-site scripting has fallen to 7th place in the OWASP Top 10 2017 report [1], cross-site scripting remains one of the most serious attack forms. Another report being published annually for the past 12 years by WhiteHat Security, WhiteHat Security Application Security Statistics Report [2], also identifies that cross-site scripting is among the top two most critical Web vulnerabilities. An interesting observation made in this report is that even though cross-site scripting is considered one of the most critical vulnerabilities, it is not being prioritized for remediation by websites. The statistics being presented suggest that the vulnerabilities receiving most remediation are vulnerabilities that are easy to fix, which is not the case for cross-site scripting. It is suggested organizations must adopt a risk-based remediation process, to prioritize the most critical vulnerabilities first, like cross-site scripting. A report [3] published by Bugcrowd Inc., a Web-based platform that uses crowdsourced security for companies to identify vulnerabilities in their applications, analyze data from their platform, including information about the most common vulnerabilities found. The data in this report is based on all Bugcrowd data from January 2013 through March 2017, which contains of over 96 000 submissions, where the most reported vulnerability is cross-site scripting with a submission rate of 25%. They also have data on the most critical vulnerabilities by type, where cross-site scripting is considered the second most critical, which is the same result found in WhiteHat Security's report. These are some of the most recent numbers regarding cross-site scripting, but there have been published numerous of studies done on XSS vulnerabilities and attacks. One study [4] from 2014 conducted a systematic literature review were they reviewed a total of 115 studies related to cross-site-scripting. They concluded that XSS still remains a big problem for Web applications, despite all the proposed research and solutions provided so far. As seen from the recent numbers from OWASP, WhiteHat and BugCrowd, this conclusion still holds true, that XSS vulnerabilities remains to be at large. With the observation about how prevalent this type of attack is, and the fact that it is not prioritized nor easy for websites to fix and remediate it, it becomes clear that the user needs some means of protecting themselves at the client-side, since it is mainly the end-users of vulnerable Web applications that are affected by potential attacks.

A. *Cross-Site Scripting Attacks*

Cross-site scripting vulnerabilities are caused by insufficient validation/sanitation of user submitted data that is used and returned by the website in the response, which could compromise the user of the site. An attacker could potentially use this vulnerability to steal users' sensitive information, hijack user sessions or rewrite whole website contents displaying fake login forms. The end-users of websites are the main victims of these attacks, but the actual websites are also affected, as the attacks might negatively impact the reputation of the site, which again could lead to fewer visitors. There exist three main types of cross-site scripting attacks, which is one of the reasons why remediation for such vulnerabilities is not an easy task, as each of the different types operate differently and thus require small differences in how to properly handle and secure them. All three types rely on insecure handling of JavaScript code, and are called Reflected, Stored and Document Oject Model (DOM) Based Cross-Site Scripting (XSS) attacks [5]:

Stored XSS occurs when user input attack code is stored on a publicly accessible area of a website, typically in a comment section, message board post, visitor log or in chat rooms. When a user visits a page where such an attack is stored, the browser will retrieve the data and render it, which in turn will execute the stored XSS attack in the browser's context. This type of XSS is very difficult to protect against on the client-side, as the client have no means to identify whether the JavaScript code coming from a website is legitimate, or if it is malicious JavaScript code injected by an attacker. From the client's perspective, all JavaScript code coming from a website is legitimate and should be rendered accordingly.

Reflected XSS occurs when the user input data is sent in a request to a website, which immediately returns data in the response to the browser, without the site first making the data safe. Reflected XSS attacks are performed by entering data into search fields, creating an error message or by other means where the response use data from the request. In a reflected XSS attack, the JavaScript attack code is not stored on the website itself. For this attack to work, a user needs to visit a specially crafted URL, containing the exploit code, for the attack to be successfully done, executing the attack in the user's browser. A Reflected XSS attack thus contains a request to and response from a website, where the code inserted in the request is being used in the response. Client-side filters can, therefore, compare the contents of the request with the response, to identify a potential attack. The proposed filter in this paper utilizes this technique, which means it focuses on primarily stopping Reflected XSS attacks.

DOM Based XSS is a type of XSS attack where the malicious data that exploits a flaw never leaves the browser. This means that from an attacker inputs malicious data to a website until the code is executed in the browser, the malicious data is not part of neither the request or the response of the website, but rather part of the DOM of the Web-page. This is because DOM based attacks rely solely on flaws using JavaScript code.

B. *Counter-Measures for XSS Attacks*

Counter-measures for XSS attacks can be achieved in several ways. The first step would be to properly identify and map the attack surface of the Web application, before implementing the desired option for protection, ideally a combination of several of the following methods:

Validation/Sanitization of all untrusted data input to a Web application makes sure that malicious input is either being rejected or manipulated into being safe for usage in the output. It might be difficult to implement this properly as it can be challenging to know what a malicious input looks like, considering all the possible attack vectors that use advanced obscuration techniques.

Output encoding is the most effective remediation for cross-site scripting attacks when done properly. It is important to implement the output encoding according to the context it is being used in, because different encodings are needed depending if HTML or JavaScript code is being used.

Content Security Policy (CSP) is another common way for preventing cross-site scripting attacks, which is a declarative policy that let Web application owners create rules for what sources the client is expecting the application to load resources from. As stated in the World Wide Web Consortium (W3C) Recommendation [6], CSP is not meant as a first line of defense mechanism, but rather an element in a defense-in-depth strategy.

Disabling JavaScript is also a possibility that would totally stop XSS, since these attacks rely on a JavaScript environment for execution. This solution can be effective for simple static websites, but most dynamic websites require some sort of JavaScript support for basic functionality, which means this remediation would not be suited as an overall solution.

In the following Section II, different filter techniques are being discussed before presenting a client-side filter implementation for the Mozilla Firefox browser in Section II-A. Then, in Section III, the presented filter is analyzed, and finally, we end the article in Section IV with the conclusion and further work.

II. Client-Side XSS Filtering

When a website is vulnerable to cross-site scripting attacks, an attacker could exploit this vulnerability and possibly steal sensitive information or hijack sessions of the users accessing the exploited website. Filters try to stop these attacks by utilizing a set of rules to detect potential malicious input data, before either blocking it or sanitizing it for safe usage. There exists many XSS filter implementations, with varying focus on the different areas such as security, performance, low false-positives and usability. All of these areas are in focus of most filters, but it is not common for a filter to be best in all categories, as they do not necessarily compensate each other. There is, however, one clear way to differentiate between filters, by dividing them into two groups, server-side and client-side filters:

Server-side filters are implemented on the server side of a website, which means it can only detect input data that are sent via the server. The DOM based XSS attack, as discussed in Section I, is an attack only relying on client-side code, which means a server-side filter would not be able to detect the attack at all, which implies it would not be able to stop the attack. This is one of the reasons why only relying on server-side protection is not enough, and why we need client-side filters.

Client-side filters are located in the client, which typically would be the Internet browser used to access the website. Client-side filtering would be able to detect DOM based XSS attacks, providing the extra protection server-side filters are missing. However, even though client-side filters could possibly detect all types of XSS attacks, it should not be used alone, without server-side filtering. By placing the filter on the client-side, it means that the user might be able to modify it to circumvent the filtering. It is, therefore, strongly recommended to utilize both server- and client-side filtering, to be able to detect all attack types and achieving defense in depth protection.

Filtering techniques: There exist several implementations for cross-site scripting filters both on the client-side and server-side of Web applications, which use many different techniques, but where most also contain some limitations [7]. This paper focuses on client-side filtering, where some of the most used techniques will be discussed here. A popular technique is to use regular expressions, which has been proved to contain several flaws in its design [8]. A popular client-side XSS filter using regular expressions is NoScript [9] for Mozilla Firefox, first released in 2005 and actively updated by the maker Giorgio Maone. The filter is matching HTML code for injected JavaScript in the request by utilizing regular expression rules for simulating the HTML parser, which would potentially lead to false-positives, as it is better to over-approximate these rules than to let an attack bypass the filter [8]. Another method for client-side XSS filtering is string-matching, used by the filter in the Google Chrome browser, XSS Auditor [8]. Auditor works by matching the HTML code for injected JavaScript code for the request with the response from the website after it is been parsed by the browser's HTML parser, see [8] for more details. This means that Auditor does not need to approximate any of the HTML parser rules, since the parsing is already done when the matching algorithm starts. This is achieved by the location of Auditor, which is between the HTML parser and the JavaScript engine, which makes it possible to block scripts after parsing, by blocking them from being sent to the JavaScript engine for execution.

Regular expressions and string matching is among the techniques being implemented in the top five most used Web browsers for desktop, which according to the online measurements from StatCounter [10] are Chrome, Firefox, Internet Explorer/Edge and Safari. Both Chrome and Safari use the mentioned string matching based XSS Auditor filter. XSS Auditor was first build into the browser engine WebKit, which Safari uses, before also being integrated into a fork of WebKit called Blink, which Chrome uses. Internet Explorer and Edge both have a filter implemented based on the regular expression technique, first introduced in Internet Explorer 8 [11]. Firefox however, being the second most used Web browser, does not have a built-in filter, but rather relies solely on CSP support, which again relies on websites to properly define the CSP rules. By not having a client-side filter the defense in depth principle is also lost, where a potential filter would provide an extra layer of security for the end-users of the application. In this paper we present an implementation for a built-in client-side filter for this extra layer of security.

A. Implementation of Client-Side Filter in Firefox

The client-side XSS filter for Firefox proposed in this paper is based on the Google Chrome browser's XSS Auditor, but with some design modifications. Due to various differences in Chrome's and Firefox's internal architecture, the proposed filter in this paper is tightly coupled to Firefox and is, hence, not meant to be a copy of XSS Auditor. The basis of the filter is to first get the input data to the website, before checking if any of this data is considered dangerous, in which case a matching comparison is done for all the scripts before they are sent to the browser for execution. Both filters are doing the filtering after the HTML parser, but the proposed Firefox filter is doing the actual matching later in the rendering process than Auditor. Whereas Auditor is doing the matching before the JavaScript engine, by examining all the DOM tree nodes, the proposed Firefox filter is not doing the matching before it is actually prepared to be sent to the JavaScript engine, in Firefox's internal `ScriptLoader.cpp` class, as seen in Figure 1 below. This means that the Firefox filter is only doing matching on the scripts sent to Firefox's internal script handler, and not the whole DOM tree.

The implementation of the proposed filter is focusing on the most common way to inject and execute JavaScript on a webpage, by using the HTML script tag. The rules for filtering are based on different ways of making JavaScript code from script tags execute in the browser. OWASP's guidelines XSS Filter Evasion Cheat Sheet [12], which contains many attack vectors trying to circumvent typical XSS filtering techniques, provided a lot of examples for the creation of this paper's filtering rules.

The filter is implemented as its own class, which could then be used in parts of Firefox requiring filtering protection. This class contains several methods for detecting potential attacks, as inline scripts and external scripts needs to be processed differently. When using the filter, it start by fetching all the input data to the website in form of GET- and POST-parameters, before checking each of these parameters if they contain any potential malicious code that can be used for executing a cross-site scripting attack. In this case, the filter checks for opening HTML script tag, `<script`. If there are any occurrences of this tag in any of the input parameters, the filter will continue its examination of the input. There are now two cases in which the input data will be considered and marked as dangerous. Either if the script tag is non-empty or it contains a non-empty attribute `src`. If any of these two conditions are being fulfilled, the filter marks the input parameter as dangerous before a matching algorithm is started to try and find the input data in any of the JavaScript code sent to Firefox for execution. This is done by comparing the actual string representation of the parameter with the string representation of all JavaScript code entered through Firefox. If this matching algorithm does find a match, the whole script that contains the input data will be blocked from execution in the browser, stopping a potential attack. If no match is found, the webpage and all its contents will load and function without any intervention from the filter.

B. Mozilla Firefox Architecture

For implementing this filter into Firefox, it is important to know how the source code is built up and how the scripts are being evaluated. Mozilla Firefox source code has a layered architecture where the code is organized as separate modular components. Firefox is multi-threaded and follows the rules of object-oriented programming, where access to internal data is achieved through public interfaces of the classes [13]. One of the primary requirements of Firefox is that it must be entirely cross-platform, which is why the browser consists of several components focusing on this area, like making sure the operation system dependent logic is hidden from the application logic. The main components can be divided up into the user interface XML User Interface Language (XUL) [14] and the browser and the rendering engine Gecko [15]. XUL is Mozilla's own language for building portable user interfaces, which is an XML language. Gecko is Mozilla's browser engine built to support many different Internet standards, including HTML 5, CSS 3, DOM, XML, JavaScript and others. Gecko contains many different components for document parsing (HTML and XML), layout engine, style system (CSS), JavaScript engine called SpiderMonkey, image library, networking, security, as well as other components. The implementation of the proposed filter is located in Gecko, right before JavaScript code from a site is being sent to SpiderMonkey for processing. Both inline and external scripts from HTML script tags are being loaded into the class `ScriptLoader.cpp`, where they are passed on to the JavaScript engine for compiling and execution. Because all scripts from script tags pass through this class, this is the main area where the filter will be used. The flow of such scripts through the application is shown in Figure 1. This makes sure that all scripts are caught and can be effectively stopped from executing by simply not sending them to the

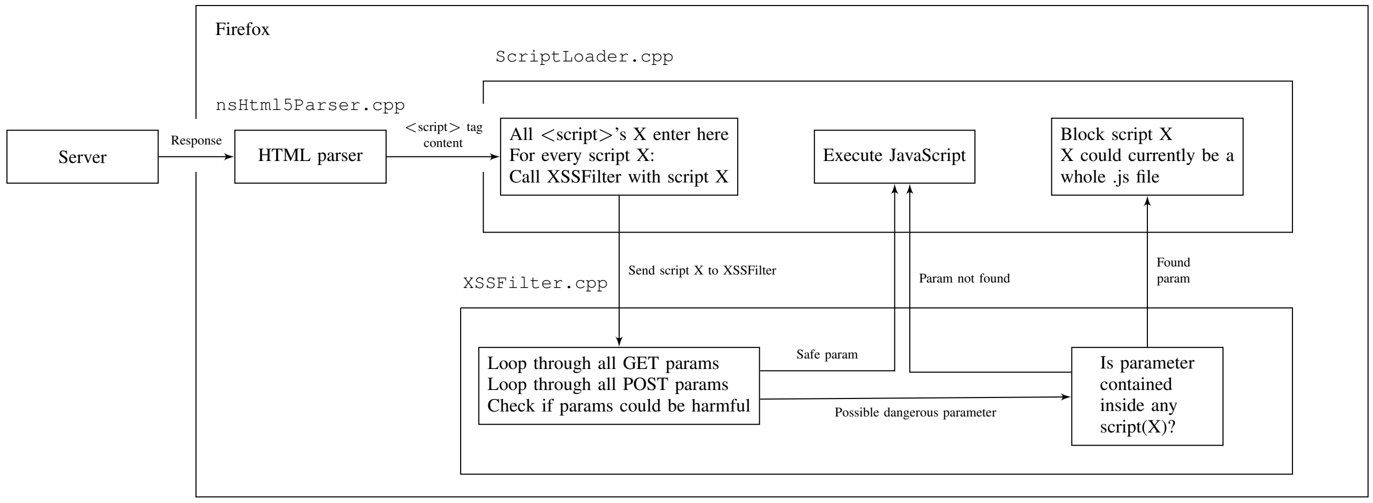

Firefox
ScriptLoader.cpp
nsHtml5Parser.cpp
Server
Response
HTML parser
<script> tag content
All <script>'s X enter here
For every script X:
Call XSSFilter with script X
Execute JavaScript
Block script X
X could currently be a whole .js file
Send script X to XSSFilter
Param not found
Found param
XSSFilter.cpp
Loop through all GET params
Loop through all POST params
Check if params could be harmful
Safe param
Possible dangerous parameter
Is parameter contained inside any script(X)?

Figure 1: Information flow in application

JavaScript engine at all. Even though all script content from script tags enter through the class `ScriptLoader.cpp`, not all input that should be interpreted as JavaScript's gets sent here. Gecko handles scripts differently based on where they originate from. HTML event handlers are being processed in another class `EventListenerManager.cpp`, before sent to the JavaScript engine. This means that for the proposed filter to work on all possible scripts from a website, it would be necessary to also use the filter in this location.

III. Analysis

A main challenge during the implementation was to properly understand the application architecture. Depending on how JavaScript code is inserted into a website, Firefox is processing the input in different modules in the application, which proved challenging to identify. The proposed filter for Firefox presented in this paper is as described in the previous section only focusing on the HTML script tag, which means all the script processing could be done in the same place in the Firefox source. However, by neglecting other means of injecting JavaScript code into a website, the filter is not capable of detecting all possible XSS attacks. Some other common HTML tags used for cross-site scripting attacks are tags like 'svg', 'object' and the usage of event handlers. It is however very possible to locate where in the Firefox source JavaScript code from other HTML tags is being processed, and to add filtering capabilities to those areas in a similar fashion done with the 'script' tag. A similar limitation is the fact that the filter only considers GET- and POST- parameters for the input. It is possible to use other input entry points like cookies, local storage, or HTTP header fields for executing cross-site scripting attacks. Neglecting support for these alternative attack vectors is also a limitation in XSS Auditor [16], but since they are valid attack vectors, they should at least be considered for improving the proposed filter in this paper.

A. Attack mitigation efficiency

When testing the implemented filter in practice, Firefox was able to successfully detect and block simple cross-site scripting attacks using the script tag for the injection point. Simple attack vectors like `<script>alert(xss)</script>` and `<script src=http://xss.rocks/xss.js></script>` both were successfully blocked by the filter when injected into a sample vulnerable website. Other more advanced attack vectors from the OWASP XSS Filter Evasion Cheat Sheet [12], like embedding spaces or tabs within the injected input, neglecting to include closing tags or substituting space with a non-alpha character were also tested, which were successfully detected and blocked by the filter. However, there is a case where the filter only was able to block parts of the injected input using only the script tags: when the input contains more than one occurrence of the script tag. An example would be the input
`<script>alert(1)</script>`
`<script>alert(2)</script>`.
In this case, the first script tag sequence containing the `alert(1)` would be blocked, at which point the filter would stop examination and hence the `alert(2)` from the second script tag would be executed, which could effectively be used to launch a successful cross-site scripting attack. This is due to the filter being limited to only detect and block the first script tag found.

As seen with the implemented filter, there is a lack of filtering rules and conditions, which makes it quite ineffective in its current form. Even with a case of only using the script tag, the filter was unable to detect all injected attacks. Not to mention all the other ways attack vectors using different HTML tags an attacker could use. By studying the OWASP's XSS Filter Evasion Cheat Sheet, where a lot of these different attack vectors are shown, with the purpose of evading common filters, the cheat sheet is effectively showing that for every attack vector, it is possible to properly detect and block the attack by using the correct rules and conditions. This also applies to the proposed filter in this paper, that it is possible to implement all these rules and their variations to be able to filter away most cross-site scripting attacks.

Another property of the implemented filter is how it handles a detected injected script, and how that affects its functionality. When the filter detects that a script from the input is found in any script loaded into the browser engine for processing, the whole script loaded for processing is being stopped before it is executed. The rest of the scripts on the website would still be loaded and executed as usual. There is however also another approach that is common for XSS filters, which is to block loading the entire website where a potential XSS attack was discovered. There are advantages and disadvantages to each of these methods. An advantage to only block specific parts of a website is that the user is still able to browse and view the other parts of the website not affected by the injection, making it a better user experience with less disruptions in case of an attack. In cases of false-positives, where there are no real attacks, this technique is more forgiving by not blocking all the website's content. A disadvantage of only blocking parts of the webpage is that in the case of a detected attack, it is not unlikely that an attacker would probably try to use different advanced attack vectors, which could trick the filter like the case described above, using double script tags. Therefore in regards of security, it is best to block the whole webpage when a potential injection attack is detected. By utilizing blocking of the whole website instead of only the parts where the script was detected would effectively allow the implemented filter described to successfully block the attack with double script tags, making the filter much more secure by just this single modification to its design. There are however negative effects by blocking the whole webpage, which is the user experience would greatly be affected by a lot of discovered attacks, which would disrupt the user from normal browsing activities and where the user ultimately maybe choose to disable the filter altogether. This is especially the case with false-positives, where there actually is no attack, but the filter still blocks the entire page from loading.

There is no simple answer to which of these techniques to use, but there are ways that websites themselves can choose what to do. By setting the HTTP header X-XSS-Protection, webpages could choose to either allow, sanitize or block detected cross-site scripting attacks [17]. This header is currently supported by other major Internet browser, but not Firefox, as Firefox does not supply built-in XSS filtering. By adding support for this header in Firefox and the implementation of the proposed filter in this paper, it would be possible also for Firefox to let the webpages themselves choose how to deal with detected cross-site scripting attacks, either allowing everything, only blocking assumed affected content, or block the whole webpage from loading.

An additional limitation of the implemented filter is the support for different input encodings. When receiving input into a webpage, the input might be encoded with different encodings, like hex encoding, which is not currently supported by the filter. This is however easily fixed by first adding a check for what encoding is used, if any, before properly decoding the input. This is a very important feature that needs to be taken into consideration, as using different character encodings is a common way to obscure cross-site scripting attacks.

B. Performance

The performance of the implemented filter is an important factor for its usefulness. We followed Mozilla's own methodology for comparing page load times across browsers [18], using popular websites to load in the browser, repeated several times, while measuring the loading time for each page. We chose 10 of the most popular news websites from Alexa [19], knowing that news sites typically contain a lot of scripts for ads and tracking. To make sure the modified browser actually ran the code for our implemented filter, we used the search function on each of the websites and tested with two different parameters, one safe and one unsafe, which would activate the filtering. We also wanted to conduct a performance test for actual vulnerable Web applications. From a website containing a list of Web applications vulnerable to XSS attacks [20], even though it was an old archive, we collected four different websites all vulnerable to XSS attacks, and then injecting them with the simple script `<script>console.log(1)</script>`. This simple script injection was chosen because it makes it easy to compare the load time between the modified Firefox and original Firefox, as this simple script would not alter the rendering of the page itself, but still be a valid cross-site scripting attack. As we also injected the 10 chosen news site with a script input, we did not expect any big different in performance between these sites and the acutual vulnerable sites, as the filter would run the same matching algorithm on all sites. As expected, even though the filter from this paper successfully detected and blocked this injection on all these vulnerable websites, there were no overhead compared to the news sites. For the full performance test, a total of 1040 page loads were performed for each browser, including both the 10 news sites and the four vulnerable sites. This resulted in an average difference of only 32.1 ms for each page load, which equals a performance overhead for the modified browser of about 0.7 % compared to the average loading times for the original Firefox browser, which is an insignificant overhead. As there are several limitations with the current implementation of the filter, a more complete version addressing these limitations would probably incur a higher overhead, but at a starting point at 0.7 % it is reason to believe the added overhead would not be of any significance. This was however a test with several limitations, as there might have been too few total page loads for each browser, the visitor traffic to the tested websites might be different and there might have been small interferences in the Internet connection when performing the test. Although there these factors might have affected the results, it is worth noting the the two browsers were tested in the same time span, which should not incur too much variation. After taking the average of the 1040 page loads for each browser, the achieved results do highly indicate that the modified browser do not incur any significant performance overhead.

IV. Conclusion and Future Work

Information flow vulnerabilities can occur when applications handle untrusted data. When this happens, users of the application might be negatively affected, without any means of protecting themselves. By utilizing client-side filtering, like proposed in this paper, the user do have a means to protect themselves from malicious attackers. By default, the Firefox browser have no such protection mechanism built in, which this paper has a proposal for adding. As seen in the analysis in Section III, there are still many important additions to be done before the filter is ready for everyday usage, but the filter do work for basic cases, which already provides more protection than the default Firefox browser, proving a solution efficient enough to work, achieving high performance with almost no overhead. When the rest of the presented additions is implemented, this filter would work as an important extra protection for the end-users of vulnerable Web applications, efficiently protecting against reflected cross-site scripting attacks.

A reasonable next step would be to further expand the filtering capabilities of the filter. This would be achieved by implementing the proposed improvements from Section III, covering all attack vectors from all possible injection points, adding more rules and conditions for the filtering, have proper decoding of input and adding support for the X-XSS-Protection header. After making these improvements, it is also necessary to do further testing, for both loading speeds and focus on security, with specially crafted attack vectors, and to make sure the filter is as robust and secure as desired, making it an effective way for protecting the end-users of websites from cross-site scripting attacks on the client-side.

REFERENCES

[1] OWASP Foundation, "Owasp top 10 - 2017 the ten most critical web application security risks," accessed: 2017-12-27. [Online]. Available: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_(en).pdf.pdf

[2] WhiteHat Security, Inc., "2017 whitehat security application security statistics report," 2017, accessed: 2017-12-21. [Online]. Available: https://info.whitehatsec.com/rs/675-YBI-674/images/WHS 2017 Application Security Report FINAL.pdf

[3] Bugcrowd Inc., "2017 state of bug bounty report," 2017, accessed: 2018-01-09. [Online]. Available: https://pages.bugcrowd.com/hubfs/Bugcrowd-2017-State-of-Bug-Bounty-Report.pdf

[4] Hydara, Isatou and Sultan, Abu Bakar Md and Zulzalil, Hazura and Admodisastro, Novia, "Current state of research on cross-site scripting (XSS)–A systematic literature review," Information and Software Technology, vol. 58, 2015, pp. 170–186.

[5] OWASP Foundation, "Types of cross-site scripting," March 2017, accessed: 2018-03-05. [Online]. Available: https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting

[6] The World Wide Web Consortium, W3C, "Content security policy level 2," December 2016, accessed: 2018-01-11. [Online]. Available: https://www.w3.org/TR/2016/REC-CSP2-20161215/

[7] S. Gupta and B. B. Gupta, "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art," International Journal of System Assurance Engineering and Management, vol. 8, no. 1, 2017, pp. 512–530.

[8] D. Bates, A. Barth, and C. Jackson, "Regular expressions considered harmful in client-side xss filters," in Proceedings of the 19th international conference on World wide web. ACM, 2010, pp. 91–100.

[9] G. Maone, "NoScript - JavaScript/Java/Flash blocker for a safer Firefox experience! - features - InformAction," accessed: 2017-12-28. [Online]. Available: https://noscript.net/features

[10] StatCounter, "Desktop browser market share worldwide dec 2016 - dec 2017," December 2017, accessed: 2018-01-11. [Online]. Available: http://gs.statcounter.com/browser-market-share/desktop/worldwide

[11] D. Ross, "Ie8 security part iv: The xss filter," July 2008, accessed: 2018-01-11. [Online]. Available: https://blogs.msdn.microsoft.com/ie/2008/07/02/ie8-security-part-iv-the-xss-filter/

[12] OWASP Foundation, "Xss filter evasion cheat sheet," October 2017, accessed: 2017-12-27. [Online]. Available: https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

[13] Mozilla Developer Network, "An introduction to hacking mozilla," Mars 2017, accessed: 2017-12-28. [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/An_introduction_to_hacking_Mozilla

[14] ——, "Introduction," September 2014, accessed: 2017-12-28. [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XUL/Tutorial/Introduction

[15] ——, "Gecko faq," September 2015, accessed: 2017-12-28. [Online]. Available: https://developer.mozilla.org/en-US/docs/Gecko/FAQ

[16] Stock, Ben and Lekies, Sebastian and Mueller, Tobias and Spiegel, Patrick and Johns, Martin, "Precise Client-side Protection against DOM-based Cross-Site Scripting." in USENIX Security Symposium, 2014, pp. 655–670.

[17] Mozilla Developer Network, "X-xss-protection," October 2017, accessed: 2017-12-28. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection

[18] D. Strohmeier, P. Dolanjski, "Comparing browser page load time: An introduction to methodology," November 2017, accessed: 2018-01-15. [Online]. Available: https://hacks.mozilla.org/2017/11/comparing-browser-page-load-time-an-introduction-to-methodology/

[19] Alexa Internet, Inc., "The top 500 sites on the web," January 2018, accessed: 2018-01-15. [Online]. Available: https://www.alexa.com/topsites

[20] "Xss archive," accessed: 2018-03-05. [Online]. Available: http://www.xssed.com/archive