Revisiting the Challenges in Aligning RE and V&V: Experiences from the Public Sector

Jacob Larsson[1], Markus Borg[2]

[1]Capgemini Sverige AB
Växjö Office
Växjö, Sweden
jacob.larsson@capgemini.com

[2]Dept. of Computer Science
Lund University
Lund, Sweden
markus.borg@cs.lth.se

***Abstract*—Successful coordination of Requirements Engineering and Testing (RET) is crucial in large-scale software engineering. If the activities involved in RET are not aligned, effort is inevitably wasted, and the probability of delivering high quality software products in time decreases. Previous work has identified sixteen challenges in aligning RET in a case study of six companies. However, all six case companies selected for the study are active in proprietary software engineering. In this experience report, we discuss to what extent the identified RET alignment challenges apply to the development of a large information system for managing grants from the European Union. We confirm that most of the findings from previous work also apply to the public sector, including the challenges of aligning goals within an organization, specifying high-quality requirements, and verifying quality aspects. Furthermore, we emphasize that the public sector might be impacted by shifting political power, and that several RET alignment challenges are amplified in multi-project environments.**

***Index Terms*—Requirements engineering, software testing, public sector, communication, software process**

I. Introduction

Aligning Requirements Engineering and Testing (RET) is an important aspect in large-scale software engineering. If the "two ends of software engineering" do not work well together, the chances for a successful project drastically decreases [13]. However, aligning RE and testing involves several challenges as highlighted in previous studies [3][1][12], e.g., communication between different organizational units, managing large amounts of RE and test documentation, and poor traceability.

Previous work has focused on alignment of RET in proprietary contexts. Sabaliauskaite et al. presented challenges in aligning RET in a company developing handheld devices for mobile communications [12]. They reported challenges from eight different categories, and particularly emphasize: 1) inefficient tool solutions, 2) poor traceability between RET, and 3) communication between requirements engineers and testers. Building on this study, Bjarnason et al. expanded the number of studied companies to support generalization of the findings. They interviewed practitioners in six companies, and identified RET challenges as well as practices to support RET. While we acknowledge the variation among the six studied companies, we notice that software development in the public sector is not represented in the study.

In this paper, we present our experiences from working in a large development project in the public sector. We base our discussion on the challenges reported in previous work by Bjarnason et al. [3]. The goal of this paper is to share our understanding on *how previously identified RET challenges from proprietary development can be extended to a software development project in the public sector.*

This experience report is organized as follows: Section II presents previous work on RET challenges, i.e., the starting point of our discussion. Our pre-understanding of RET inevitably influences our analysis. Thus, to help the reader assess our interpretations, Section II also reports our background along with the case description. Section III presents the case description. Section IV presents our observations of RET alignment challenges in the organization under study. Finally, Section V discusses our findings in relation to previous work.

II. Background

A. Challenges in RET Alignment

Bjarnason et al. conducted a multi-unit case to investigate challenges concerning RET alignment. They interviewed different roles in six companies in Sweden and Norway to get a broad perspective on the matter, and identified 16 challenges repeatedly mentioned (listed in Table I).

The case study revealed four high level observations. First, Bjarnason et al. report that *the human side of software engineering is central to RET alignment*, i.e., communication and coordination between people. Second, *the quality and accuracy of the requirements are critical* for testing software in line with the original expectations. Third, *the size of the organization is a key variation factor* for which RET challenges are experienced. Fourth, *in domains mandated by rigid development standards*, e.g., safety standards, *documentation and traceability are enforced. In less rigid contexts, the motivation for RET alignment is instead internal*, and it is important for the organization to assess the consequences of poor alignment, e.g., incorrectly implemented requirements, delays, and wasted effort.

B. Authors' Preunderstanding

All experiences reported in this paper belong to the first author. He is a software engineering consultant specializing in testing, specifically test processes and test management. He has worked in software development projects in the public sector for about 10 years, both in Denmark and Sweden. About 80% of his time he has worked in development of healthcare Information Systems (IS), and the rest in projects concerning IS in the public sector. As a consultant he has also worked with RE, mainly elicitation and analysis. The message in this paper is based on the first author's personal thoughts, and do not necessarily reflect the views of Capgemini.

III. Case Description

The experiences reported in this paper originate from the first author's long-term consulting in the public and health sector. We refer to this governmental agency addressed in this experience report as GOV.

A. Overview of GOV and the IS under Development

GOV is responsible for national administration of grants from the European Union (EU) for one important sector. The software development organization within GOV develops a new IS, offering end-users improved grant management using web-based solutions. The IS will replace several existing systems, integrating all steps of the grant process from the initial application to the final payment. The development project aims at using Open Source Software whenever feasible, and the development is based on solutions such as Java, JBoss, and PostgreSQL. The runtime environment for the IS is Red Hat Linux.

The two most important quality attributes for the new IS are *interoperability* and *performance*. The IS will have multiple dependencies to other systems through external interfaces. Thus, providing high quality APIs is an explicit requirement on the IS. Reliable integration with other is a necessity during operation. Also, as the number of future users is high, approximately 500 simultaneous users during the peak periods, the network performance is critical.

We define two important stakeholders of the IS. *End users* are domain experts working at either GOV or county administrative boards around the country. They have detailed knowledge about the grant application process and assist *clients*. Clients (i.e., individuals and enterprises active in the sector) use the IS to apply for grants and track the progress. They have limited access rights in the IS. Furthermore, we define the *business* as the organization at GOV that will use the IS to fulfil their operational needs. The business employs *business analysts*, i.e., requirements engineers responsible for elicitation and specification of the requirements on the IS.

Development at GOV is governed by framework agreements. Several subcontractors develop a large fraction of the IS together, and also perform RE and V&V activities. Thus, as framework agreements are used instead of outsourcing to subcontractors, GOV are responsible for all risks within the development project. The number of engineers in the development organization is 100-200. There are about twice as many consultants as internal developers. Development of the IS is organized as 12 different projects, each project employing 10-20 engineers.

B. GOV Development Process

The development process at GOV is based on the Rational Unified Process (RUP) [11], and further inspired by Domain Driven Design (DDD) [8] and Test Driven Development (TDD) [1]. Also, six agile practices are used in the process, expressed as the six goals below:

1) End users and the business work close together. Their work is transparent within the entire project.
2) High quality is obtained through continuous integration and automated testing.
3) Incremental development enables frequent acceptance testing.
4) Continuous delivery of system documentation.
5) Cross-functional teams with integrated competences
6) Retrospective meetings as part of every sprint planning.

Planning within the development project at GOV is performed on three different levels. *Strategic planning* (6 months) involves resource allocation and specification of dates for integration, referred to as milestones. *Operational planning* (9 weeks) covers three future sprints and includes user story workshops and prototype demos. Finally, *sprint planning* (3 weeks) deals with detailed planning of what is to be delivered in the sprint. Sprint planning is practically organized on a Scrum task board.

GOV has collected several lessons learned from retrospective meetings during the development of the IS. They have condensed their experiences into four key factors for a successful delivery. First, *good relations between project members as well as members of other related projects* are required. Openness, cooperation, communication and other human aspects are vital for timely project deliveries at GOV, in line with findings from Bjarnason et al. [1]. Second, the projects should conduct *regular prototype demos of the IS under development*, inviting all project members (and other related projects). The feedback from the demo sessions is highly valuable. Third, the project should acknowledge the value of the *retrospective part of the sprint planning*. The potential of retrospectives in software engineering is well-known, especially emphasized in agile development contexts [2]. Fourth, GOV establishes *teams that work across projects*. While the original study by Bjarnason et al. present cross-role review meetings and cross-functional involvement [3], GOV instead stresses coordination of multiple projects running in parallel.

IV. RET Alignment Challenges at GOV

This section reports how the challenges identified by Bjarnason et al. [3] affect the development at GOV. Table I shows an overview of our comparison. Outsourcing of components or testing (Ch10) does not apply to the development at GOV, and it is not elaborated further.

Table I. Summary of previously reported challenges in RET alignment and their relevance at GOV. M denotes major challenges, while m are less critical. A dash ('-') denotes a challenge that has not been experienced at GOV.

Id	Challenge		Comment
Ch1	**Aligning goals within an organization**	M	GOV is a large organization affected by long term political decisions. At the same time, the development projects and the many consultants establish short term goals. Furthermore, aligning goals across projects is even more challenging.
Ch2	**Cooperating successfully**	M	Team work supported by cross-functional members, daily stand-up meetings, and open landscapes. Still team work could improve further. Again, cooperation between projects even harder.
Ch3.1	**Defining clear and verifiable requirements**	M	GOV has a history of poorly specified software requirements, causing considerable financial corrections. RE process has improved. Now FURPS+ is applied, but quality requirements needs more attention.
Ch3.2	**Defining complete requirements**	M	Requirements elicitation is hard, and GOV relies on an iterative process with formal validation of user stories. Several roles are involved in reviews of user stories, at least business analysts, developers, and testers.
Ch3.3	**Keeping requirements documentation updated**	M	Enterprise Architect is used to model the IS and store user stories. Changes involve a cumbersome process and regeneration of documents. Also, much effort is spent on maintaining the product backlog.
Ch4.1	**Full test coverage**	m	Different roles interpret "full coverage" differently. The test strategy states the goal of 80% statement coverage by automated unit tests using the FitNesse framework. Automated testing focuses on verifying complex decision tables that otherwise involves tedious manual work.
Ch4.2	**Defining a good verification process**	m	GOV has a mature V&V process, including cross-role review meetings, automated unit testing, exploratory testing, and formal acceptance testing. Main challenges involve long time between RE and testing activities, and creation of dummy test data.
Ch4.3	**Verifying quality requirements**	M	Verification of quality aspects is difficult as they often cannot be assessed until the IS is in operation. The project at GOV struggles mainly with reliability, usability, and maintainability.
Ch5	**Maintaining alignment when requirements change**	M	Changes are inevitable, e.g., originating from legislative changes or misinterpretations. GOV has a well-defined change management process, but an important challenge is communicating changes and impact.
Ch6.1	**Defining requirements at abstraction level matching test cases**	-	GOV has mature RE and V&V processes, and mismatching abstraction levels is not a challenge.
Ch6.2	**Coordinating requirements at different abstraction levels**	-	Coordination of RE artifacts at different abstraction levels is not a challenge at GOV.
Ch7.1	**Tracing between requirements and test cases**	m	Traceability within the project involves a high degree of manual work, but is not considered a major challenge.
Ch7.2	**Tracing between requirements abstraction levels**	-	The mature RE process and the close collaboration with the business mitigates this challenge.
Ch8	**Time and resource availability**	m	GOV uses a mix of internal developers and external consultants as part of framework agreements. The main resource bottleneck is key employees from the business, that frequently are needed in the development project both for RE and V&V activities.
Ch9	**Managing a large document space**	-	The organization has an established information policy, relying on tagging of information entities in a central repository. Information access is not a major challenge.
Ch10	**Outsourcing of components or testing**	N/A	GOV does not outsource any development activities.

Ch1) Aligning goals within an organization

It is *difficult to align goals within large organizations as GOV*. A major challenge comes from the *different time horizons* considered within the organization. While certain roles consider how a development project affects the future of the organization, other roles focus on short term deliveries. In the specific development project under study, the *long term goals are decided by external political factors*, i.e., GOV has clear expectations on the IS under development. On the other hand, because of political decisions on the national or European level, GOV might also order changes to the IS during the development project. In the development projects, employing a high number of consultants, short term goals dominate.

The business must ensure that the development project stays on track even though all formal decisions have not yet been made. Also, the business has the responsibility to create the overall strategic project plan (referred to as "the business plan"). The strategic project plan clarifies dependencies between development activities, and specifies when different components of the IS should be implemented and delivered.

For a successful project, the business must have the required knowledge to feed the project with well-specified requirements (see also Ch3.1). To establish correct expectations on the IS, the business must collaborate with the business analysts requirements for elicitation and specification of requirements. Unfortunately, the collaboration is not always successful, often *because the business has not prepared enough* or because *a common language has not yet been established* (e.g., discrepancies in terminology exist).

Moreover, aligning goals within a project is difficult, but it might be *even more challenging between different projects in an organization*. GOV is a large organization and multiple projects are conducted in parallel.

Ch2) Cooperating successfully

Successful cooperation is difficult, but also vital for a successful project. Project members must be willing to cooperate and show initiative. Unfortunately, the anticipated *level of team work is not always reached*, as some engineers prefer to work on their own with little transparency. On the other hand, we have also seen flexible engineers, such as business analysts supporting test engineers with formal verification.

We have observed three aspects at GOV that support cooperation between roles and then in turn also support RET alignment. First, *low thresholds between roles enable spontaneous collaboration*. This is stimulated in the projects by encouraging flexibility and communication across teams. Second, *the daily stand-up meeting* is key for efficient communication and cooperation between project members. However, for the stand-up meeting to fulfil its potential, it is important that all project members are present. Third, *physical distances are minimized in the projects*. All project members work in the same open landscape. Thus, business analysts, developers, and testers are all close to each other, and also employees from the business itself are present.

Cooperation also involves inter-project aspects. Projects at GOV often depend on deliveries from other projects, e.g., updated processes guidelines or a new server setup for a specific environment. GOV tackles inter-project cooperation by specifying detailed processes and scheduling regular meetings. It is critical that the expectations on all different projects are clear.

Ch3.1) Defining clear and verifiable requirements

Often the overall goals of the system are clear for the business analysts, but *GOV has a history of poorly specified software requirements*. Poor RE has contributed to imperfect procurement, and in turn resulted in inadequate ISs. Evaluations by the EU Commission have been negative, and reprimands have been issued. The EU Commission has determined substantial *financial corrections* (repayment of EU grants), exceeding 100,000,000€ during the first 15 years as a member state in the EU. However, GOV has improved their RE process considerably in recent years, and we now consider it mature.

Requirements are classified according to the FURPS+ quality model [9]. FURPS+, a well-known model in industrial practice [4], classifies software quality attributes as one out of five categories: Functionality, Usability, Reliability, Performance, and Supportability. Using the FURPS+ model supports the overall understanding of the requirements. In particular, *explicitly "flagging" quality requirements raises the awareness of their importance in the organization* (verification of quality requirements is further elaborated in Ch4.3). Still, the business analysts in the project mainly focus on functional requirements. Consequently, the *requirements engineers put most of their effort on developing extensive specifications for the functionality of the system, and less effort on quality aspects.*

All detailed functional requirements are combined in *user stories*. The user stories follow the priority decided during the strategic planning to ensure that the most important business goals are implemented first.

Ch3.2) Defining complete requirements

Defining complete requirements is hard and requires an iterative process involving business analysts, developers, and testers. Requirements change over time as new requirements are identified. The business analysts and the business decide when the detailed requirements of a user story are good enough to initiate an *audit*, i.e., a meeting conducted to validate the user story. Audits are managed by the test leader and documented in an *audit log* that lists the issues found in the user story. The audit includes at least representatives from the business analysts, the developers, and the testers. Sometimes also the employees from the business itself participate in the audits.

If major issues are identified during an audit, e.g., gaps in the specified functionality or conflicting requirements, the user story needs rework. The business analysts must then improve the user story and call for a new audit with the same participants. At the new meeting, a revised version of the audit

log is created. This iterative process is repeated until no issues are found and then the validated user story is handed over to the developers for implementation.

Ch3.3) Keeping requirements documentation updated

Even though GOV uses modern RE tools, *keeping the requirements documentation updated is a challenge*. GOV uses Enterprise Architect[1] by Sparx Systems to model the IS under development. As described in Ch3.1, the requirements are organized in the form of user stories that include the following components:

- pre-requisites for the user story to start
- actors involved
- basic flow(s)
- alternative flow(s)
- error flow(s)

The detailed requirements are specified as steps of the different flows. Requirements documentation is then generated as MS Word documents from Enterprise Architect. Modifying a requirement requires updates in the tool, and regeneration of the impacted requirements documents. Unfortunately, *updating the requirements is a cumbersome process.* All developers do not have access to Enterprise Architect, both because of the high price of user licenses, and because the tool is complex and thus demands specific training.

Functionality identified during the strategic-, and operational planning is *maintained in the product backlog document*. The backlog document serves as a master document that outlines the main focus areas within the development project. The business and the business analysts must know when the different areas are scheduled for development so that they can obtain the right knowledge before the involved requirements are specified. The backlog is a living document, and *much effort is spent on its maintenance*.

Ch4.1) Full test coverage

Full test coverage has an intrinsic challenge as it is an ambiguous concept, *understood differently among the project members* at GOV. The project defines five levels of testing: 1) *unit testing*, 2) *integration testing*, 3) *system integration testing*, 4) *acceptance testing,* and 5) *load and performance testing*. The test strategy document, written by the test architect, outlines start and exit criteria for the different levels of testing

The test strategy lists the different types of testing for the requirements, as organized and classified by the FURPS+ quality model. To achieve what is referred to as "full test coverage", *80% statement coverage shall be met in the unit testing*. The tool FitNesse[2] is used to reach this goal by automated testing. However, not all test cases can be automated, and thus the stop criterion is set to 80% statement coverage. Still, *reaching this compromised goal can be challenging*.

[1] http://www.sparxsystems.com.au/products/ea/

[2] http://www.fitnesse.org/

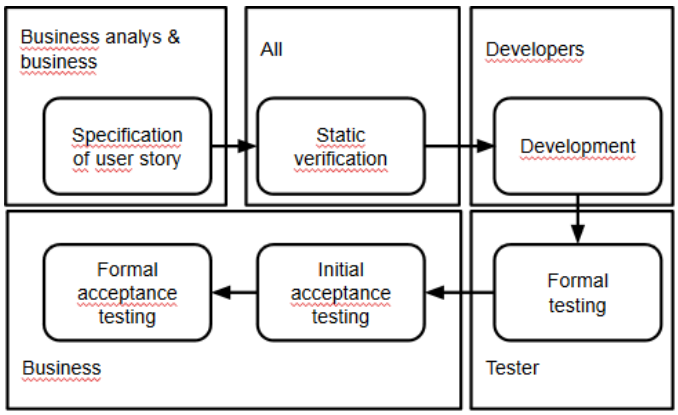

Figure 1: Overview of the testing process at GOV.

The automated test cases are used for regression testing of the continuous integration. The selection of test cases in the suite of regression tests is focused on covering decision tables related to the most complex parts of the IS. Also, automating the testing of the decision tables is in general prioritized, as verifying them manually is tedious and time-consuming activity.

Ch4.2) Defining a good verification process

Early testing activities is a cornerstone of the test strategy at GOV. The test strategy explicitly states the goal that verification should start as early as possible in the project. Figure 1 shows an overview of the testing process at GOV. Normally early verification implies *static verification*, i.e., document reviews. However, static verification requires the engineers responsible for test (i.e., both the test leader and the testers) to have *good knowledge within the agriculture domain*. To gain more knowledge within the domain, it is important for the testers to be actively involved with the business as much as possible. The formal static verification is performed as soon as the business analysts and business consider the user story to be "good enough" for the developers to start the implementation.

GOV conducts *static verification through formal review meetings*. The test leader invites the following roles to the meeting: 1) a business analyst, 2) a representative for the business, later also responsible for the acceptance testing of the user story, 3) a developer as well as 4) a tester from the responsible development team. The test leader takes notes during the meeting, and writes a formal document, i.e., a *review protocol*, listing all issues that must be addressed before development of the user story can start. All invitees must prepare for the review meeting in advance and provide issues to discuss.

Development starts when the review meeting concludes that a user story is well-specified. In parallel *the tester within the team starts designing test cases for the user story*. Thanks to continuous integration, i.e., nightly builds on the development server, the tester can execute the test cases to verify the user story the day after the developer commits the related code. The test strategy encourages testers to verify user stories as soon as possible to ensure *rapid feedback to the developers*.

When the tester within the team has verified the implementation of the user story, the user story is *assigned to formal testing*. GOV conducts formal tests in a dedicated test environment, and the outcome of the activity is a formal test report.

When the formal testing has verified the user story, the representative from the business (i.e., the person involved in the initial static verification of the user story) performs initial acceptance testing. *Initial acceptance testing is conducted using exploratory testing*, and no documentation is required. Finally, if the user story passes the exploratory testing, it is released to *preproduction*, the final phase prior to release. In preproduction *formal acceptance testing* with the business commences, using specified test cases, resulting in formal acceptance test protocols. The formal acceptance testing with the business is performed as part of the final activities in every sprint.

One of the challenges of this formal verification activity with the business is the *too long time between the specification of the requirements and when the business can actually test it*. When the business are verifying the functionality through exploratory testing an indirect regression test is performed and other issues or defects might be found. This means that the developers (who might have started implementing another user story) might have to work on source code related to user stories they already considered completed.

Another challenge is to *create the test data required to verify user stories*. As the project develops a new system, no old data exists, and realistic data mimicking the future IS in operation must be manually created by the testers. As creation of test data prior to acceptance testing is typically important only for the testers, i.e., the business analysts do not value it unless it is specifically given by the detailed requirements, there are sometimes mismatching priorities within the project.

Ch4.3) Verifying quality requirements

Verifying quality requirements during system development is difficult, as *many quality aspects cannot be assessed until the system is in operation*. While the FURPS+ quality model increases the awareness of the quality requirements (see Ch3.1), the actual verification activity is challenging. Quality aspects that GOV struggles with verifying include:

- Reliability. The IS shall more or less always be available and proper backup must be in place.
- Usability. The user interface shall provide the same look-and-feel for all subsystems that are developed in parallel.
- Maintainability. The IS will be in operation for decades to come, and it must continue to evolve. An example of a related quality requirement is that the same version of the JavaServer Faces shall be used for the entire IS.

Ch5) Maintaining alignment when requirements change

Additional requirements, or updates to existing requirements, will inevitably be identified during the project. Triggers of changes to requirements include new *legislation* and *misinterpreted requirements* within the project. The static verification (presented in Ch4.2) shall normally identify misinterpretations, but sometimes it is not enough.

Two major activities occur when new requirements are identified. First, a *change impact analysis* is conducted to understand how the additional requirement affects the implementation. GOV considers this activity critical, thus it is assigned a high priority. Second, new requirements are *prioritized as part of a sprint planning* (presented in Section III.B). Verification of new and modified changes to requirements then follows the same verification process stated in Ch4.2.

A particular challenge involved in changed requirements is to *communicate changes and impact within the project*. At GOV, the main channel for this communication is the stand-up meetings in the mornings.

Ch6.1) Defining requirements at abstraction level matching test cases

Compared to the previously discussed challenges, Ch6.1 is less of an issue at GOV, as t*he verification activities match the abstraction levels of the requirements well*. GOV has reached both a mature level of RE and testing, and we consider the two activities aligned from an abstraction perspective.

Testers perform formal testing, i.e., testing that results in a test report, and shall provide *full traceability to the detailed requirements outlined in the user stories*. A test case can either have a one-to-one or one-to-many relationship to requirements. The tracing from test cases to requirements is a bottom-up activity that is first performed during the formal testing. This is closest abstraction level where formal tests are performed with more focus on the question "what is delivered follows the detailed requirements".

The business performs acceptance testing of the functionality of the IS, not only focusing on "what?" but also on "how and why?". Acceptance testing involves how the requirement is realized in the system and why the requirement is implemented as it is. One aspect of the acceptance testing is thus to identify possible improvements. The business also verifies the overall business processes, which can be seen as a top-down approach starting from a higher abstraction level.

Ch6.2) Coordinating requirements at different abstraction levels

Coordination of requirements across abstraction levels is *not an explicit challenge* in aligning RE and testing at GOV. As stated in Ch6.1, the RE process in the organization is mature. The business is the main role in the coordination. The acceptance testing, performed by the business, verifies that the detailed requirements fulfil the high level process and related high-level quality aspects.

Ch7.1) Tracing between requirements and test cases

The tracing between RE and testing *requires considerable effort* at GOV, *but is not one of the prioritized RET alignment challenge*s. GOV applies both tool supported and manual tracing practices. Enterprise Architect is used to visualize relations between user stories and detailed requirements.

Traceability between test documentation (on a user story level) and detailed requirements is manually maintained in the formal test cases.

Ch7.2) Tracing between requirements abstraction levels

Tracing within RE is *not a challenge* at GOV. The tracing, as well as the coordination (Ch6.2), is supported by an established RE process and close collaboration with the business.

Ch8) Time and resource availability

At GOV, *the main challenge regarding resource availability is that key project members from the business get overloaded*. GOV has at least *one employee of each role in the development organization, except in testing where only consultants are used*. The GOV employees in the development organization are important, as they bridge the knowledge between the software development and the business. They support communication and cooperation (Ch2), by knowing key individuals in the organization, making the right connections when needed. Furthermore, employing GOV engineers in the development project helps business and development to understand each other, and support aligning goals within the organization (Ch1).

It happens that the *key project members from the business are assigned too much work*. The business is involved in the IS development in several ways. A few people from the business shall be involved in the requirements elicitation and specification, outlining user stories, static verification, exploratory testing, and formal acceptance testing. Moreover, they are assigned for other tasks at GOV up to 25% of their time. Thus, the business needs a high tolerance of stress, be able to prioritize their time well, to keep focused during the time allocated, as well as context switching between different roles.

Another resource that is involved in the project is IT infrastructure. Engineers from infrastructure are available on site twice per week. The availability of resources (typically engineers work in the same project landscape) supports both RET alignment and cooperation in general.

Ch9) Managing a large document space

Project documentation is generally easy to find. GOV keeps all documentation in a central repository. The organization has an information management policy that relies on tagging information entities instead of organizing information in folder structures. GOV has a well-defined terminology for the tag space, supporting quick and concise access to project documentation. Moreover, GOV has a number of defined *views* that are used to locate the documents needed for common work tasks, e.g., system requirements and acceptance testing.

Ch10) Outsourcing of components or testing

GOV *does not outsource any software development*.

V. Summary and Concluding Remarks

Alignment of RET is important in all software engineering contexts, also in the public sector. In this experience report, we analyzed how 16 previously identified challenges [3] in RET alignment affect development of an IS at a governmental agency (GOV). We have experienced most of the challenges (11 out of 16), and the most important challenges involve: 1) aligning goals within an organization, 2) successful cooperation, 3) various activities related to RE, 4) verification of quality requirements, and 5) maintaining RET alignment as requirements change.

A number of previously reported challenges in RET alignment are not as obvious at GOV. Working with requirements at different abstraction levels is not as challenging as reported by Bjarnason et al., neither coordination nor tracing between abstraction levels. A possible explanation is that both the RE and V&V processes at GOV have matured in recent years. Also the information management at GOV appears successful, as managing the large amount of RE and V&V artifacts is not an explicit challenge. Finally, GOV does not outsource any development, thus no related challenges exist.

We reported two main differences regarding RET alignment at GOV compared to previous challenges in proprietary contexts. First, the goals of the organization depend on politics both on the national and the EU level. External directives might change as political powers shift. Also legislative changes might impact the IS development. Second, while previous work focuses on RET alignment challenges within projects, we highlight that challenges might be further amplified in contexts where the organization executes multiple projects in parallel. Example challenges that are amplified include: 1) aligning goals within an organization, 2) successful cooperation, and 3) time and resource allocation.

Acknowledgements

This work was funded by the Industrial Excellence Center EASE – Embedded Applications Software Engineering (http://ease.cs.lth.se).

References

[1] K. Beck, *Test-Driven Development: By Example*, 2003, Addison-Wesley.

[2] E. Bjarnason, and B. Regnell. “Evidence-Based Timelines for Agile Project Retrospectives - A Method Proposal”, *Agile Processes in Software Engineering and Extreme Programming*, XP 2012, pp. 177-184, 2012.

[3] E. Bjarnason, P. Runeson, M. Borg et al., “Challenges and Practices in Aligning Requirements with Verification and validation: A Case Study of Six Companies”, *Empirical Software Engineering*, July 2013.

[4] H. P. Breivold, and I. Crnkovic, “Analysis of Software Evolvability in Quality Models”, in *Proc. of the 35th Euromicro Conference on Software Engineering and Advanced Applications*, 2009, pp. 279-282.

[5] D. Damian, and J. Chisan, “An Empirical Study of the Complex Relationships between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality,

and Risk Management", *Transactions on Software Engineering*, 32(7), 2006, pp 433–453.

[7] *Proc. of the 1st International Software Metrics Symposium*, 1993, pp. 141–152.

[8] E. Evans, *Domain Driven Design: Tackling Complexity in the Heart of Software*, 2004, Pearson Education.

[9] Grady, R.B., and Caswell, D.L., *Software Metrics: Establishing a Company-wide Program*, 1987, Prentice-Hall.

[10] M. I. Kamata, and T. Tamai, "How Does Requirements Quality Relate to Project Success or Failure?", in *Proc. of the International Conference on Requirements Engineering*, 2007, pp. 69-78.

[6] A. Davis, S. Overmyer, K. Jordan et al., "Identifying and measuring quality in a software requirements specification", in

[11] P. Kruchten, *The Rational Unified Process: An Introduction*, 2004, Pearson Education.

[12] G. Sabaliauskaite, A. Loconsole, E. Engström et al., "Challenges in Aligning Requirements Engineering and Verification in a Large-scale Industrial Context", In *Proc. of REFSQ*, 2010.

[13] M. Unterkalmsteiner, R. Feldt, and T. Gorschek, "A Taxonomy for Requirements Engineering and Software Test Alignment", *Transactions on Software Engineering and Methodology*, 23(2), 2014, pp. 16:1-16:38.