문서 지향적 프로세스에서의 SCRUM 프로젝트 적용: 실험 사례연구
(Running a SCRUM project within a Document Driven Process: An Experimental Case Study Report)

조나단 소이어 †
(Seok-Won Lee)

요 약 본 논문은 아주대학교에서 컴퓨터공학과 대학원생들이 SCRUM을 이용하여 고급 소프트웨어 공학 프로젝트를 운영한 결과를 사례 연구로써 다룬 논문이다. 일반적으로 현장의 고객과 그들의 요구사항이 문서에 대하여 제한사항들을 제공하는 환경은 문서 지향적인 단일 단계 방법론에서는 흔하지 않은 일이다. 본 논문에서는 프로젝트를 진행하는 방법론과 실습 과정을 자세히 다루며 SCRUM 프로세스를 적용하는 과정에서 일반적인 소프트웨어 팀 구성원들이 마주하게 되는 몇 가지 문제들을 되돌아본다. 또한 본 논문은 Henrik Kniberg의 SCRUM 체크리스트와 Agile Manifesto를 사용하여 실무적 내용과 태스크의 성공을 평가하는 것으로 결론을 지었다. 이 프로젝트는 아주대학교에서 진행되었다.

키워드: SCRUM, Agile, 제품 책임자, 태이젠, Bottom-up influencing

Abstract
This paper examines how a Computer Engineering Graduate student team ran their Advanced Software Engineering Capstone project using SCRUM. The environment provided contextual challenges in terms of the on-site customer and upfront requirements document, not uncommon in a document driven single-step methodology. The paper details the methodology and practices used to run the project, and reflects on some of the challenges faced by the members of a typical software team when transitioning to a SCRUM process. The paper concludes by evaluating the success of the techniques and practices compared to the Agile Manifesto and Henrik Kniberg’s Scrum checklist. The project was undertaken at South Korea’s Ajou University.

Keywords: SCRUM, agile, product owner, tizen, bottom-up influencing

† This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2013R1A1A2008901)
‡ 이 논문은 2015 한국 소프트웨어공학 학회대회에서 "Running a SCRUM project within a Document Driven Process"의 제목으로 발표된 논문을 확장한 것임

논문접수 : 2015년 3월 17일
(Received 17 March 2015)
논문수정 : 2015년 6월 11일
(Revised 11 June 2015)
심사완료 : 2015년 6월 19일
(Accepted 19 June 2015)
1. Introduction

Decision makers capable of orchestrating top down changes to the way a software function operate are not always going to lead from the front. There are risks associated with change and Agile is hard to understand especially when it is juxtaposed against the simple to understand document driven single-step process of Waterfall [1]. This moves the responsibility for proving the value of the process to the team doing the work, sitting at the core for a pragmatic way of functional bottom up working practices and influencing techniques whether partly supported by management or driven entirely from the coal face of a software team. The challenge lies in the conflict between what Waterfall and SCRUM require. Given the state of many organizations it is essential that we develop techniques for tackling this problem. Successful demonstration of the usage of SCRUM would allow management to confidently make decisions that overtly support the usage of this process.

The project as presented required the creation of two main documents, the requirements and technical design document.

The requirements document was our first and most overt challenge. Traditional document driven single-step environments require software teams to use or produce artifacts that do not necessarily flow into an Agile methodology. A comprehensive upfront design is in conflict with the idea of change. We think about what we do not know, we waste effort on what cannot be understood just yet. Thinking this far ahead restricts our ability to be adaptive and to change and more importantly we do not really know what is needed. Humphreys Requirements Uncertainty principle states “For a new software system, the requirements will not be completely known until after the users have used it.” [2].

The document itself once thrown over the wall to the development team often causes confusion and much to the dismay of the well intentioned analysts it is not uncommon for software engineers and customers alike to state that a document is not comprehensible without significant effort. We wanted our document to be useful to the key stakeholders from executable document perspective. The first level of influence being usefulness, usefulness within this context can be defined as something that is valuable to our customers, analysts, software engineers and testers within a SCRUM environment while being acceptable to the traditional document driven audience.

Samsung funded our project by providing hardware for the Tizen platform for mobile and the Galaxy Gear. They played no role in directing what we ultimately built. This creative freedom meant we had no customer and we were building software for ourselves. This is a team consisting of mostly technically minded people which presents an interesting risk as there should be a healthy friction between technical and business needs. This lack of friction leads to one side pushing towards their favor. It is not good when a team build a technically sound application that adds little or no value. It is not good when a business get everything they require based on an aggressive and unsustainable timeline often leading to technical debt and an unstable application. Given this we elected one of the members to assume the role of a Product Owner for the product. They played no part in cutting code. Thus we faced two problems common within a Document Driven process. We had to use or produce a document we did not see as necessary and at worst potentially harmful to the development process. The other was access to a real customer. In a business setting it may be that the customer is not readily available or they consider their work done as they were interrogated while the analysts produced the requirements document.

This paper analysis the techniques that were used by the team to be able to operate efficiently given the constraints of a Document Driven Process when used within SCRUM and concludes by evaluating the techniques used.

2. Related Work

The problem of running a SCRUM process within a document driven process is not new and has been examined by other researchers.

Heeager, Lise Tordrup, and Peter Axel Nielsen [3] examine the development process at a Danish pharmaceutical company who are using SCRUM within a Document-Driven process. The implementation of the SCRUM process is not perfect and the practices and
techniques used by the pharmaceutical company are not entirely aligned with the Scrum and Agile process. Unlike this paper the authors are independent from the organization providing an impartial view of how things are.

Hayata, Tomohiro, and Jianchao Han [4] propose a hybrid model and predict that the future of software development lies at the convergence of these two processes. This paper is a theoretical paper with the focus on merging the two processes where as our findings are based on the goal of replacing a traditional process with an Agile process.

Sumrell, Megan [5] provides helpful practical techniques on how a test team should transition to a Scrum process. They highlight 6 key issues a test team should address. They include using experts, expect roles to change, everyone is responsible for the quality of software, use the correct tooling, continuous integration and finally patience. We elaborate on this topic from our point of view and experience in section 11.1.

Lewis, Jeanne, and Kevin Neher [6] have written a paper about how a team is attempting to transition Microsoft to Agile. They used common metaphors that are used in Waterfall to create a common Lingua franca and provide guidance on what worked for them. This is a very pragmatic and relevant paper to a large or even small organization looking to transition to Agile.

3. Case Study Background

The Advanced Software Engineering course in Ajou’s Computer Engineering Graduate School splits the class into several groups who are required to produce useful software on Samsungs Tizen and/or the Galaxy Gear Watch. There are several key checkpoints where the team produces a requirements document, technical design, final document and final product. Teams are left to self-organize and decide on the best technical practices and software development processes to follow. This paper documents a team of four members with varying degrees of experience with SCRUM. Two of the members had not used SCRUM and were not familiar with the process. One of the members had good experience in an academic context. The final member had 10 years of experience working in a Professional setting using Agile and most recently using SCRUM. He is a certified SCRUM master.

In total the project had 10 one week sprints at its disposal. We had a two week break after Sprint 3 due to mid-term exams. Sprints commenced on a Monday with a planning meeting and culminating in an end of sprint review and retrospective on a Friday. Due to other commitments at graduate school we allocated 100 ideal Engineering hours in total per week. Daily stand-up meetings were held in the NiSE Lab meeting room at 11am on Monday and Tuesday and Wednesday through Friday at 10am. We used VersionOne as our SCRUM tool. Activity in VersionOne was updated on our A1 paper SCRUM board with post-it notes and vice versa.

We shared Information regarding retrospectives, analysis and technical design on our project wiki. Most design discussions took place immediately after the stand-up meeting was complete. We were informal in the way we designed and preferred using a whiteboard with a marker and uploading the resulting diagrams or rough sketches to our wiki.

Early on the project had an extremely slow velocity. We resolved our velocity problems by reducing the amount of work scheduled based on a previous Sprint velocity. Velocity being the amount of work we completed against what we set out to achieve. The back-end of the Sprints for this release saw a gradual and then dramatic improvement in estimation and thereby meeting our commitments to our Product Owner.

The introduction of a Product Owner and In-Sprint tester resulted in a higher quantity of bugs being detected and thereby the final Sprints were used to fix defects and harden the product.

4. Development Process and Techniques

We elected to build a tool to facilitate the management of tasks within a group. This is not dissimilar to the way a SCRUM board works. The scenario we used to define how to use the product was a family organizing a party.

To give the reader a basic idea of the system the high level feature names and story titles are defined in Table 1 below.
Tizen has two options for building applications, the native application framework and web framework. The web framework was used for the following reasons:

1. The Tizen Mobile Web Browser is the most HTML5 compliant web browser in the market.
2. We believe that the future of mobile development does not lie with specific frameworks but instead with open frameworks like Angular.
3. The HTML5 capabilities comfortably supported our objectives.

We used Google’s Angular MVC framework for our client side application framework and Node.js for the server.

Figure 1 breaks down the broad technical components of the system. The SCRUM process uses a number of key meetings. The planning, daily stand-up, retrospective and sprint review meeting were the formal meetings we used to manage our process. We held a planning meeting first thing on a Monday morning to discuss what we would be working on in the week to come. This is where it was important for us to have a Product Owner who could drive the direction and success of the product. We used a lab meeting room for our daily stand-up meeting.

The meeting was usually a quick one lasting no longer than 5 - 10 minutes with everyone going through the usual format of:

1. What have I done since the last Scrum meeting (yesterday)?

<table>
<thead>
<tr>
<th>Features</th>
<th>Story</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Management</td>
<td>- View Task</td>
</tr>
<tr>
<td></td>
<td>- Create Task</td>
</tr>
<tr>
<td></td>
<td>- Modify Task</td>
</tr>
<tr>
<td></td>
<td>- Delete Task</td>
</tr>
<tr>
<td></td>
<td>- Archive Task</td>
</tr>
<tr>
<td></td>
<td>- Create Account</td>
</tr>
<tr>
<td></td>
<td>- Login</td>
</tr>
<tr>
<td></td>
<td>- Change Password</td>
</tr>
<tr>
<td></td>
<td>- Forgotten Password</td>
</tr>
<tr>
<td></td>
<td>- Assign Task Management Board user to Task</td>
</tr>
<tr>
<td></td>
<td>- Remove Task Management Board User from a Task</td>
</tr>
<tr>
<td></td>
<td>- Invite Others to Workspace</td>
</tr>
<tr>
<td></td>
<td>- Remove Task Management Board User from Workspace</td>
</tr>
<tr>
<td>Collaborative</td>
<td>- Create Account</td>
</tr>
<tr>
<td></td>
<td>- Login</td>
</tr>
<tr>
<td></td>
<td>- Change Password</td>
</tr>
<tr>
<td></td>
<td>- Forgotten Password</td>
</tr>
<tr>
<td></td>
<td>- Assign Task Management Board user to a Task</td>
</tr>
<tr>
<td></td>
<td>- Remove Task Management Board User from a Task</td>
</tr>
<tr>
<td></td>
<td>- Invite Others to Workspace</td>
</tr>
<tr>
<td></td>
<td>- Remove Task Management Board User from Workspace</td>
</tr>
<tr>
<td>Application as a Service</td>
<td>- Import Tasks</td>
</tr>
<tr>
<td></td>
<td>- Export Tasks</td>
</tr>
<tr>
<td></td>
<td>- Authenticate</td>
</tr>
<tr>
<td></td>
<td>- Mobile Notifications</td>
</tr>
<tr>
<td></td>
<td>- Galaxy Gear Notifications</td>
</tr>
<tr>
<td></td>
<td>- View Activity Feed</td>
</tr>
<tr>
<td>End user as a voice</td>
<td>- Create a Reason</td>
</tr>
<tr>
<td></td>
<td>- View a Reason</td>
</tr>
<tr>
<td></td>
<td>- Update a Reason</td>
</tr>
<tr>
<td></td>
<td>- Delete a Reason</td>
</tr>
<tr>
<td></td>
<td>- Link Tasks to a Reason</td>
</tr>
<tr>
<td></td>
<td>- Unlink Task from Reason</td>
</tr>
<tr>
<td>Planning Management</td>
<td>- Prioritize Tasks</td>
</tr>
<tr>
<td></td>
<td>- Task Estimation</td>
</tr>
<tr>
<td></td>
<td>- Due Date Specification</td>
</tr>
<tr>
<td></td>
<td>- Automatic Scheduling</td>
</tr>
<tr>
<td>Workspace Management</td>
<td>- List Workspace</td>
</tr>
<tr>
<td></td>
<td>- Create New Workspace</td>
</tr>
<tr>
<td></td>
<td>- Select Workspace Details</td>
</tr>
<tr>
<td></td>
<td>- Delete Workspace</td>
</tr>
<tr>
<td>Quality Requirements</td>
<td>- Efficiency</td>
</tr>
<tr>
<td>(Non-Functional Requirements)</td>
<td>- Usability</td>
</tr>
<tr>
<td></td>
<td>- Uncertainty of Network Availability</td>
</tr>
<tr>
<td></td>
<td>- Security</td>
</tr>
<tr>
<td></td>
<td>- Privacy</td>
</tr>
<tr>
<td></td>
<td>- Perceived Availability</td>
</tr>
</tbody>
</table>
2. What will I do before the next Scrum meeting (tomorrow)?
3. What prevents me from performing my work as efficiently as possible?

One of the goals of a SCRUM team is for it to be self-organizing. To prevent team members from reporting status like a command-and-control structure we changed the chair or SCRUM master hat around in order to create the feeling that everyone is responsible for delivery. The difference being the members who report status will see the team as hierarchical and expect the SCRUM master to solve all their problems. We needed everybody to feel responsible for delivery.

A large A1 paper sheet was used to create our SCRUM board with columns for Ready, In Progress, Done and Verified. We used post-it notes for stories. A fixed place was used to meet every day but we did not want to share our ideas with other teams, therefore after the stand-up we removed the large A1 paper sheet from the board and safely stored it in a team member’s desk. It would have been better to have it visible to all team members however the environment necessitated this approach. On a Friday a review meeting of what was built throughout the week was held. The latest version of the application was deployed to the Tizen Mobile phone and used to conduct a demo of the product to our Product Owner. Lastly, on a Friday we held a Retrospective meeting following the Review meeting. This meeting was used to reflect on the following:

1. What went well?
2. What we must improve on?
3. What we must stop doing?

This was a good time for the team to get things out in the open and figure out ways on how to improve performance.

A traditional style requirements document was required as a delivery during the 4th sprint forcing us to create all the specifications for the whole product in a few sittings. This prompted us to make one of our team members a Product Owner as well as the In-Sprint Tester. In hindsight we should have done this from day one. Given more resources I would have elected to have a separate Product Owner and In-Sprint Tester. The dual function worked well for us given our limited resourcing. They have similar functions regarding their independence in viewing the system however testers do not drive the success of the final product.

Given the relative freedom of how to produce this document we created something that neatly feeds into our Backlog. The goal we set ourselves was to create a specification that is lean. It would be both useful for the development team and something our Product Owner could easily engage with. In traditional teams especially with cross functional silos, developers do not usually have the ability to contribute to the document and its structure in the early phases. Giving us full control of the document creation process meant we could create something acceptable to all stakeholders. We wanted it to be lean with little wasted effort. It is important to point out that producing this document is mandatory. Waterfall organizations place similar mandatory requirements on development teams.

The final document was used to directly create our Product Backlog and used for Verification and Validation by our Product Owner and In-Sprint Tester.

The goal was to create no waste however we produced an excess of requirements we never ended up building. It could be argued this set a vision for future development and helped us to prioritize the most valuable features to build first.

Requirements documents should focus on what the business wants and not how to implement the detail. This seems to be a challenge for many but the finest analysts. We combined two techniques to help the author think about the usefulness of functionality and the scenarios that it would be used in.

What was really encouraging about this approach is you could not tell that there were four authors of the document. This approach forces people to use a very specific way of defining requirements.

I picked out the “End user as a voice” to illustrate how we decomposed the value the customer was asking for. The following template was used for defining stories.

As a type of user  
I want some goal  
So that some reason

Table 2 illustrates how we defined the Story for
the Creation of a Reason behind a Task.

This approach gave us the flexibility of not thinking too far ahead but at the same time providing a conversational placeholder. The details are enough for us to understand what needs to be built but not how it should be built. During our Monday morning planning meeting our Product Owner provided details of how best to implement the story.

We were able to directly input the stories from the document into VersionOne. A big problem when operating within a traditional environment is documents that agile teams are provided do not neatly fit into this specialized way of working. A specification that is implementable within the context of Agile would be a step in the right direction [7]. The following template was used for defining acceptance criteria.

**Given** some initial context (the givens),
**When** an event occurs,
**Then** ensure some outcomes.

Table 3 illustrates how the Acceptance Criteria for the creation of a reason behind a task.

The significance of this approach being we have something testers, developers and business can clearly understand and secondly use. It is been communicated at the right level. We wanted to use Dan North’s Behavior Driven Development which defines Acceptance Criteria as Given, When, Then [8].

His definition of GWT came about through his experiences working with Test Driven Development. What is particularly interesting is the observation that this is a form of analysis. GWT is powerful because we can take that analysis and directly define the passing criteria for the unit tests in code. This very same specification can be used by the test team to check whether the software is behaving as expected. This is an implementable and executable specification.

Once a Story was completed and its status changed to Done in VersionOne it was deployed to the device and handed over to our tester. In order to reduce the amount of tests that failed and defects raised we used the following process before handing over to test.

1. Code gets peer reviewed
2. Functionality peer reviewed on the mobile device and met Story and its Acceptance tests
3. Unit Tests written that resulted in an executable requirement
4. Code successfully integrated with the Jenkins Build Server telling us we had successfully built the product

These steps defined the concept of DONE within development. The goal being to have as few defects as possible and the team’s collective goal from development to test functions was to see that a story gets passed. If a story failed one or more of its acceptance tests this was immediately raised with the stories developer. If the issues were not immediately resolvable but the story was acceptable we raised a defect to track the issue and prioritized the defect for the following sprint. This loose definition defined when something was ACCEPTED by test.

As can be seen from Figure 2 below, introducing the In-Sprint tester increased the number of defects
raised in the product. This focused attention on fixing these and improving the quality of the application.

SCRUM requires someone to assume the role of a Product Owner. This key person decides what needs to be built and its priority. They are in a large part responsible for the success of the software. Given we did not have a Customer for this product we setup the notion of scenarios that our Product Owner would use to drive the thinking behind deciding on what the team would work on.[9]

Scenarios meant we had real life usage example of the tool that we could commonly use to add context to the stories we were building. Scenarios add consistency into development [10]. In our scenario it allowed our Product Owner to think clearly about why we are doing things. As an example, what is more important to build the Workspace Management Feature or the Collaboration Feature?

The reference to a product scenario allowed us to identify why we were building this product and the goals of our customers.

The scenario we setup was of a family planning a birthday party.

The family consists of Mom, Dad, two kids, Grandma and Grandpa. It starts off with Mom adding tasks to the task board and inviting Dad who invites the kids and Grandma who invites Grandpa. They all collaborate on the tasks until they were finished.

Given this extra knowledge its clear why you would build the Collaboration package first. This family is only focused on this one time event and the Collaboration Feature allows them to work together to achieve the goal of planning their party.

This approach proved extremely useful in focusing our Product Owners mind. It provided a platform for discussion amongst ourselves given a specific goal we were trying to achieve for the product. Lifting us from detail and forcing us to look at the system from a high level.

If we consider the product as a whole the Story is a fairly low level requirement. The scenario gave our Product Owner perspective and allowed them to think about the system through the eyes of the people using it. Scenarios gave our Product Owner a tool to better prioritize the work. Automated Unit testing is particularly important when it comes to Agile projects. Coupled with refactoring they drive emergent design and given the real possibility of change it is essential that developers can refactor while not breaking existing functionality.

The code in Figure 3 is written using the Mocha test framework. The tests are written using the Behaviour Driven Development style. What is important to notice is the specification is captured in the text output of a successfully passed test. There are better ways of composing tests however for our purposes it allowed the writer of the test to ensure that they explicitly met the Acceptance Criteria for a Story.

The output in Figure 4 contains details of the passing tests. This shows how the specification directly maps to the output from the tests. A future evolution on this approach would be updating Version One when a test passed. This empowers testers to know which tests were automated by the development team and more importantly which tests were not. It gives all stakeholders a clear definition of what was built for a specific feature. This is real-time accurate documentation of what the system does. These unit tests result in self-documenting code. A new developer could run the BDD style tests and reading them understand exactly what the system does on a high to detail level.

The Quality of the final product was one of our core goals. A number of key practises from the XP realm were implemented. We did not use XP however we borrowed from the XP stream of thought in the way we setup our technical environment.

We used User Stories, Small Releases, System Metaphors, Collective code ownership, Coding Standards, Simple Design, Automated Unit Testing and we did
Fig. 3 Mocha based framework code used to write the Unit Tests in the BDD style format

Fig. 4 Output of the BDD style Unit Tests
not overwork ourselves.

We built up some technical debt due to putting off refactoring. The project was fairly short and we felt that this was acceptable.

We initially Pair Programmed but decided to work alone. One of the team members suffers from very poor eye sight and therefore needs the editor font enlarged to such an extent it becomes inconvenient for his pair.

Other Technical practises
• Continuous Integration through the use of Jenkins
• Always having a working copy of the software on the Tizen Mobile Device.
• Peer Code Reviews
• Frequent source control commits

5. Analysis of findings

The empirical nature of this study meant we could experiment with preconceived concepts and ideas. Poppendieck [11] talks about the lean concept The Last Responsible Moment and how it applies to Requirements Engineering. The principle regards starting software development before all requirements are known as not doing so increases the length of the feedback loop. It is hard to be truly unbiased in this empirical experiment because you cannot roll back the hands of time and see how things would have turned out if you did less upfront planning. When we started Sprint 1–3 we were coming up with the next Sprints stories during the Sprint planning meeting. Sprint 4 onwards allowed us to draw from a comprehensive backlog. Having more work than we had time to do forced us to carefully prioritise our work.

Multicultural teams are something Koreans have to deal with as they welcome more foreign nationals to their shores. This presents the issues of language and cultural differences to working practises. Korea is widely considered to be one of the most cultures in the worlds and having a servant leader in a SCRUM team means that there is no manager issuing instructions. Initially one member held the SCRUM master role but we noticed that Korean members instead waited for work to be assigned to them, switching the role resulted in a greater level of involvement. Agile and SCRUM are cooperative processes and anyone who has observed how Koreans work together to win battles in League of Legends will know that when it comes to cooperation Koreans are world beaters.

We never intended to produce a document that resulted in complete traceability. This however was one of the key things that resulted when we created an implementable specification. Reconciling incomplete or unimplemented requirements was a matter of matching BDD output against Story style specifications. This is a process that can be done through automation given the exact same usage of a ubiquitous language.

6. Practical Challenges

Running SCRUM within a Document Driven Process is not ideal and without significant trust and communication amongst organizational silos it will be very hard to achieve success. Document Driven processes like Waterfall compartmentalize different aspects of the process and organizational structures tend to reflect this. Teams are created for different functions such as Test, Development, Reporting, Analysis and Project Management.

Organizational antibodies often start gathering as soon as changes appear [12]. Given this is a significant change to existing none collaborative working practices, this does nothing to improve trust amongst organizational silos which hampers the free flow of ideas and open communication. Communication being an essential part of Agile working practices and the SCRUM process. Silos with their own culture and working practices will find it hard to accept influence so significant that it breaks down the barriers so easily introduced with the resource pooling concept of silos.

We had ideal conditions to implement SCRUM within a document driven process as we had control of all aspects of the Software Engineering process. There was no old guard looking to protect their way of working and we could create a process without the spectre of political interference and antibodies looming.

Waterfall segregates the processes for Requirements, Technical Design, Implementation and Test. Agile takes all these functions and brings them together.

6.1 Test Team

We used the concept of an In-Sprint tester. This
role is very different from that used within a gated approach. Testers within a green field gated project are allocated time after the software has been built where they perform verification.

The iterative approach to working might not be wholly trusted and they may want to still perform final product integration testing. Coupled with Unit Tests and In-Sprint testing it should not be necessary. This type of argument might be difficult to have with a test team who believe the only thing worth testing is the final product. There is an additional threat to the Test Managers power base. His resources are being insourced to a different team, whether real or not this may be perceived as a loss of power.

Testing with In-Sprint testers and end of product testers may push up the overall test costs of the project as we duplicate effort during a transition.

6.2 Analysis Team

Analysis teams usually enjoy first contact with a Customer and can build up a privileged relationship. Analysis teams may use templates approved by various standards bodies and may not see the value in creating a document using a different unapproved format. They may also feel their role is diminished due to the increased involvement of the full team in understanding customer requirements.

The stories defined as part of building up the backlog whether through the Requirements Document, backlog grooming or another means are ultimately just conversational placeholders. The role for an analyst within a SCRUM project is still important. A good analyst would be able to work slightly ahead in understanding what needs to be built and when it comes to the Iteration Planning meetings they could be well prepared in facilitation of discussion and details regarding implementation.

Analysts are sometimes accustomed to directing the project and the Analysts role within a SCRUM project should not be confused with that of the Product Owner and SCRUM Master.

6.3 Project Management

Scrum uses a SCRUM master instead of a Traditional Project Manager. Project Managers will need to adjust to less downward management of the team. They need to address the conventional project management concerns such as budget, scheduling, and resourcing but instead they need to consider them within the context of SCRUM. What is helpful is if a Project Manager can champion the needs of the Scrum team with management and ensure traditional silos are dissolved as much as possible.

6.4 Reporting Team

We did not have a reporting function within our team however Report Developers will also face a number of challenges to transitioning from a Document Driven process to an iterative and incremental. Creating Reports is a specialized skill and silos will generally have a specialised Business Intelligence or Reporting team. Report Developers usually come in towards the end of a project and create the reports as part of the final delivery. SCRUM teams need to focus on delivering the highest priority stories that add value to the customer. Reports are often an essential part of the software and customers will want to see these as early as possible. Agile practises result in the database being frequently refactored and the reporting team will need to adjust to this more volatile way of working. They need to incorporate agile techniques and especially stay away from directly accessing database tables. This is certainly a big challenge and adequate support should be provided to help them transition to an Agile way of building reports.

6.5 Development Team

This paper discusses change driven from within the development team. Development teams transitioning to SCRUM will have their own set of challenges to contend with.

It would be highly toxic if team members within this unit did not really want to use SCRUM. The development team should Champion the process. The SCRUM master or a Technical Lead must be aware of technical practises that make SCRUM work and ensure the team understands and follows them.

SCRUM requires a working copy of the software at all times and especially for the Sprint Demo. This process should be automated and is best illustrated by the usage of Continuous Integration and Test Driven Development. The working copy of the software however sits on the back of quality technical processes. SCRUM is a project management process
and not a technical process however SCRUM works better with agile high quality technical practises.

If these technical and people issues are not adequately addressed any attempt to introduce SCRUM could implode from the centre of change.

7. Evaluating Our Efforts

We examined our efforts in two ways. We used the Agile Manifestos 12 principles [13] and Henrik Kniberg’s Scrum checklist [14] which has received favourable feedback from Jeff Sutherland the co-creator of the Scrum process.

7.1 Scrum Checklist

The Scrum Checklist provides an interesting barometer of how far you are to implementing Scrum. We did not use it throughout the project but in hindsight it would have been a very useful tool to analyse how well we were using the Scrum process.

The bottom line (Table 4)

These are said to be the most important outcomes.

| PO has direct contact with team | Except for one member everyone was collocated. |
| PO has direct contact with stakeholders | The course Professor and Samsung were our two key stakeholders however this project was a little precarious as we were building the software as our own concept. We were essentially the key stakeholders of the tool. |
| PO speaks with one voice | Yes |
| Team has a sprint backlog | VersionOne contained our Product Backlog. |
| Sprint backlog is highly visible | Electronically visible through VersionOne |
| Sprint backlog is updated daily | Yes |
| Sprint backlog owned exclusively by the Team | Yes |
| Daily scrum happens | Yes |
| Whole team participates in daily scrum | Yes |
| Problems & impediments are surfaced at daily scrum | Yes |
| Demo happens after every sprint | We stated incorporating demos after Sprint 3. |
| Demo shows working, tested Software | Yes |
| During Demo feedback received from stakeholders & PO | Yes |
| Have Definition of Done (DoD) | As described earlier in this paper we clearly defined what Done was within our context. |
| DoD achievable within each Iteration | Yes |
| Team respects DoD | Yes |
| Retrospective happens after every sprint | Yes |
| Retrospective results in concrete improvement proposals | Yes |
| Some proposals actually get implemented | The greatest example of this was highlighting the need for a PO. This was swiftly implemented. |
| Whole team + PO participates | Yes |
| PO has a product backlog (PBL) | We had a shared PBL |
| Top items are prioritized by business value | Educating members on the Scrum process takes time and concepts like the importance of |

| Table 4 Agile 체크리스트에 의한 핵심 요점 |
| Delivering working, tested software every 4 weeks or less | We started delivering working software after Sprint 3. |
| Delivering what the business needs most | Using Scenarios combined with our POs prioritization we built the highest business value items first. In hindsight it would have been useful for our PO to start using the tool post Sprint 3. |
| Process is continuously improving | We discontinued pair-programming. Introduced an in-sprint tester. Created a fixed role of a PO. |

Core Scrum (Table 5)

Central practises to call yourself a Scrum team.

| Clearly defined product owner | One of our members assumed the full time role of a PO and In–Sprint tester. |
| PO is empowered to prioritize | We created a PO specifically because we were prioritizing based on our individual visions for the project. |
| PO has knowledge to prioritize | The PO used Scenarios to assist them with deciding what needed to be built next. The Scrum Master strongly communicated the amount of Sprint remaining to help focus the prioritization efforts. |
PBL prioritization only really hit home towards the end of the project. Top items are estimated Yes
Estimates written by the team This was mostly done by the Scrum master. Final estimation took place during the planning meeting.
Top items in PBL small enough to fit in a sprint Yes
PO understands purpose of all backlog items Yes
Have sprint planning meetings Held every Monday morning.
PO participates Yes
PO brings up-to-date PBL to planning meeting In our case the Scrum Master brought the PBL. It could be roles were blurred at times or it was just a matter of education which can take time to sink in.
Whole team participates in planning meeting Yes
Planning meeting results in a sprint plan Yes
Whole team believes plan created in planning meeting is achievable Yes
PO satisfied with priorities Yes
Time boxed iterations Yes, we did have one Sprint overrun due to other obligations.
Iteration length 4 weeks or less Iterations were one week
Iterations always end on time One of the Sprints was extended.
Team not disrupted or controlled by outsiders Yes
Team usually delivers what they committed to No, we were initially very poor with this and due to technical issues it took longer than expected to complete some work. Towards the end of the project we become much better at delivering what we had committed to.
Team members sit together One of our members was in a separate lab.
Max 9 people per team 4 members

7.2 Agile Manifesto

Principle 1
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

As discussed we used a Product Owner who was facilitated by using scenarios. We delivered working software from sprint 3.

Principle 2
Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

The document risked pushing us down a particular stream of thinking and at times we had discussions that started with “that is what it says in the document”. We resorted to retrospectively updating the document in order to make changes we felt necessary. I felt the real problem here is being forced to think about a problem too early which is what the requirements document did.

Principle 3
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

We had working software after Sprint 3. We could have placed a greater emphasis on the product being actively used as part of the Product Owners day to day life.

Principle 4
Business people and developers must work together daily throughout the project.

The Product Owner was our business person however a more ideal situation would have been having Samsung driving the product. It is worth noting when we had our product unveiling one of the representatives from Samsung said there were many similar products like ours in the market. This is an example of real customer feedback. It would have been better to have this feedback earlier on in the project.

Principle 5
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

We enjoyed full support from Samsung. They provided the hardware and point man to help resolve any technical issues we may encounter.

Our time was roughly split between this project and other academic commitments. We were mostly a very highly motivated team. Commitments to other areas meant we could not always give the project a full work week’s effort.

Principle 6
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Three members were collocated while another mem-
ber worked in a lab a couple of doors down. We held stand-up meetings and used KaKaoTalk for general group related discussions. There were a number of instances were being fully collocated would have helped resolve issues considerably faster.

**Principle 7**
Working software is the primary measure of progress.

We had working software from Sprint 3. There was an issue with the meaning of Done and Accepted which directly impact the meaning of working software. This was picked up and corrected mid project.

**Principle 8**
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

This project was a time-boxed 3 month project. I did feel that after the final sprint we were all spent and happy to be finished. Given the context of this project it was fine however I do not believe we would have been able to maintain this pace indefinitely.

**Principle 9**
Continuous attention to technical excellence and good design enhances agility.

Technical debt was aggressively dealt with however again given the projects short life-time towards the end of the project we were more lenient on technical perfection.

As a whole the project is sound and the technical debt is manageable and part of the emergence of design within the software.

**Principle 10**
Simplicity—the art of maximizing the amount of work not done—is essential.

We did not build things we did not need however we did do a considerable amount of analysis of features we never ended up building. We additionally struggled to get OAuth to work on the Tizen framework and therefore we were forced to build full authentication functionality. This took up considerable time and sadly reduced the amount of functionality we could build.

**Principle 11**
The best architectures, requirements, and designs emerge from self-organizing teams.

The architecture for the system emerged through test driven development and simple design approaches.

The requirements we collaborated on even though a Product Owner drives the vision these requirements are discussed in an open platform. We were fully self-organizing.

**Principle 12**
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

We held daily stand-up meetings and weekly retrospectives.

As is typical in a stand-up meeting we discussed what we worked on the previous day, plans for that day and any issues we were facing.

We used retrospectives to discuss how we could be better.

**8. Lessons Learned**

SCRUM by its nature forces people to address issues as they evolve the way they work. We faced many difficulties throughout the project. They ranged from changing timelines, inexperience with SCRUM, irregular time logging in VersionOne. These are actually quite common even in large organizations.

We did not have a large team however trying to get everyone to code was not in our best interests. This became most apparent in our lack of Product Owner which resulted in us steering the product based on our vision of what it should technically be. It is vital that a Product Owner drives what is built and the priority it is built in for a SCRUM project. We waited until we started having arguments about what best to build before it became apparent we had a problem. Introducing this role allowed one person to drive the vision of the product from a Business point of view instead of from the depth of a technical team.

It is a utopian ideal that everyone codes however teams need independent roles that work in unison towards a common goal.

**9. Conclusion**

This research contributes to the Agile, SCRUM, Requirements Engineering and BDD community. There are a number of experimental techniques that were used within the SCRUM process and documentation and reporting on the outcome of these
helps us to better understand SCRUM empirically.

This project allowed us to manipulate and control variables. Projects that do not allow as much flexibility may experience different outcomes. Future research would benefit from using a traditional style upfront requirements document and creating silos between operational units. This would mimic an organization transitioning to a SCRUM process which uses operational silos. Furthermore, it would be useful to identify traditional Requirements Engineering practices that complement the SCRUM process. Analysts still find themselves in SCRUM teams but the practices that benefit a SCRUM team are different to that of a traditional approach.

To conclude, we managed to run our project using SCRUM embedded within a document driven process. The creation of an executable specification and providing a scenario were key artefacts for allowing us to efficiently operate. What we learnt was that it is possible to run SCRUM within a Document Driven process however one needs to be creative and adaptive in the usage. In many ways all environments provide challenges from a lack of support to an aversion to Agile. We were fortunate because we encompassed all aspects of build, test and analysis.

References


Jonathan Sawyer
2003년 National Diploma in Information Technology from Cape Peninsula University of Technology. 2007년 Bachelor of Technology from UNISA 10 Years of Industry experience working as an Agile Consultant and C# Developer. Research interests include Artificial Intelligence, Topic Modelling and Agile Software Development.

이석원