Marta Olszewska | Mikołaj Olszewski | Sergey Ostroumov
Gohar Shah | Haider Rizvi | Bilal Altaf

Experimenting with Event-B and Scrum on Student Project Course

Turku Centre *for* Computer Science

TUCS Technical Report
No 1176, March 2017

Experimenting with Event-B and Scrum on Student Project Course

Marta Olszewska

PhD, Åbo Akademi University, Department of Information Technologies

Mikołaj Olszewski

PhD, Vaadin Expert at Vaadin

Sergey Ostroumov

PhD, Software Engineer at Softability; Åbo Akademi University, Department of Information Technologies

Gohar Shah

Åbo Akademi University, Department of Information Technologies

Syed Haider Abbas Rizvi

Åbo Akademi University, Department of Information Technologies

Bilal Atlaf

Åbo Akademi University, Department of Information Technologies

TUCS Technical Report

No 1176, March 2017

Abstract

Agile methodologies and frameworks are present in IT field for over 15 years now. Coming from industry, they reached research and teaching at the academia, to finally be utilised in the student projects. Formal methods, on the other hand, exist for over 40 years and constantly iterate between the needs of industry and the resources provided by university research. Nonetheless, they seem to remain difficult to be taught and learnt. Thus, usually, they are placed as a separate learning module, not re-used between courses.

Student projects can be thought of as small ecosystems combining the learning methods required by the given project with the application of knowledge and skills gathered so far. This paper presents a student project, which was executed within a Project Course throughout 7 months. The project was combining the Vaadin framework (UI), Event-B formal method (proving system properties) and Scrum (development process) in order to create a web-application. The course was mimicking the real-world environment, where a Team of developers is having an industrial customer to whom a functional system needs to be delivered.

Our contribution is two-fold and encompasses observations and recommendations regarding (i) the use of Scrum in student projects and (ii) the application of formal methods in "traditional" software development in the student context.

Keywords: Event-B, Scrum, Student Project Course, Experimentation

RITES Laboratory
Distributed Systems laboratory
Integrated Design of Quality Systems Group (InDeQS)

1 Introduction

Agile methods have been present on the IT stage for over 15 years now [1]. The application spectrum varies in many respects: from small-scale companies like start-ups [2] to large-scale industrial players [3]. In addition, the application domains are diverse: web-applications [4] and services [5] on one hand and the safety-critical systems [6] on the other.

Regardless of the application, there is a need for validation of the feasibility of the used methods. Experimentation is one of the techniques, which allows one to investigate the practicality of the proposed novel approaches, as well as highlights its advantages and drawbacks [7].

Agile methods, although primarily present in the industry and research, are also a part of the academic curriculum, usually as an element of a software engineering course that tackles the software development processes [8]. Students learn the basis of agile philosophy, its values, principles and practices, together with certain agile processes (e.g. XP [9] or Scrum [10]). However, they lack the possibility of the hands-on experience on how agile approaches work in practice. Therefore, it is difficult to find substantiation that agile methods, in particular Scrum, actually facilitate students' work when they are involved in software engineering team-projects [11].

There is even less evidence on the use of formal methods by the students, although formal methods have been applied for over 40 years and are present in most of academic teaching curricula [12] [13]. In particular, the empirical work on group assignments involving students who apply formal methods is scarce [14]. Formal methods, such as Event-B, are especially beneficial when applied to safety-critical or mixed criticality systems [15]. They assure that the created system preserves some required properties, i.e., it functions as desired. They can be used to model and prove the most critical part of the system, while the rest of the system is implemented in the "traditional way" [16].

To the best of our knowledge, there have been no experience reports on combining formal methods and agile approaches in software engineering student projects involving teams, so far. This paper describes a student project executed within the master-level course *Project Course* within Åbo Akademi University [17], where a group of students was working on the development of a scheduling system utilising an agile development process (Scrum). The uniqueness of this particular project is two-fold: (i) the use of an agile method in the student project, (ii) engaging formal methods in the development. Regarding the former, the environment for the project was made as real as possible, i.e., involving the industrial customer, providing loose requirements document, as well as requiring certain methodologies and development frameworks – all to be arranged within Scrum. As for the formal methods, these were used to model the key properties of the scheduler. No project within the Project Course has been using any kind of formalisms, so far.

As a major contribution of the paper, we consider the lessons learnt from the student project, where students were embracing an agile method (Scrum), learning and using a formal method (Event-B), as well as utilising other methods and frameworks required by the customer. Furthermore, we provide our insight on how to successfully conduct a project course running in an agile setting, reflect on the possible risks and problems, as well as give recommendations on how to sustain the progress of the project. We base our advice on the semi-structured interviews and collection of statements we gathered from the developers and the customer during and after the project. We believe that some of our observations can be transferred to industry, especially in the cases when a new team is being formed, formal methods are employed, or a new staff member not familiar with the technologies used is added to the project.

The paper is structured as follows. First, we describe the Project Course at Åbo Akademi University that sets up the scene for the development. Then we follow with a short description of a scheduler system developed within that course. Next, we present the methodologies used in the project. Afterwards, we give the context of our investigation. Next, we describe the development itself, focusing on two aspects: the formal modelling and the system implementation parts, as well as the aspect of quality assessment. Then we provide the observations on the development from the perspective of the developers and the customer. Consequently, we build the lessons learnt and recommendations. We conclude with the discussion on the threats to validity in our investigation and final remarks.

2 Project Course

Project Course is a course organised yearly, from October to March by Åbo Akademi University. It can be taken by Master-level students of Computer Engineering, Computer Science and Information Systems. It is worth 10 study points, which corresponds roughly to 240 work-hours. A group of 4-6 persons works collaboratively to develop a fully functional version of the planned ICT solution, i.e., carry out a development project from concept, through design and implementation to testing and deployment. Furthermore, there should be an exploitation plan detailing why and how the solution is useful and/or can be turned into a viable business solution.

The team is supposed to use the theoretical skills from previous courses and apply them in practice. It is also required to provide documentation and business plan for the solution, as well as present the project, product and progress to stakeholders, lecturers, colleagues and the general public. Moreover, it is essential for the team to interact with the stakeholder, be it a company, a lecturer or other external party, in order to learn how to communicate the requirements and progress, as well as to develop a product that reflects the vision of the stakeholder. Finally, the team needs to react to the changes in the project while the project is being executed, which means modifying and tailoring the project as it advances. Each team has a mentor assigned to it, who deals with issues regarding project management, team and motivation. A mentor can be of help also in technical matters.

Project course with its setting is structured to mimic the challenges entailed by a real life IT project. In this setting, it is advised to use, e.g., a task management system, version control tools (repository) and any other tools that facilitate the development. It is also important to track the effort of the development team members, as well as estimate the effort necessary to complete the tasks. Moreover, the team should be able to evaluate the risks in the project (person on a sick leave, a person leaving the team, technological difficulties, etc.).

A similar format of the course is given in parallel by Turku University of Applied Sciences and Turku University. The final products are demonstrated by the teams at the annual ICT Showroom. The Showroom is an exhibition and competition, where students of the course present their project work done during the past year. A jury consisting of lecturers and representatives of IT companies decides who will be granted the prize of the best project in terms of commercial potential, technical feasibility and presentation (poster and demo). Moreover, students and visitors can vote for their favourite project.

2.1 Meeting Scheduler

The idea for the project has been given by the stakeholder and specified in a document presented in Appendix A. The purpose of the project was to develop a M*eeting Scheduler* implemented as a web-application. The Meeting scheduler had to support its users with organising events by scheduling them in a calendar with the best suitable date for every invited participant. A second separate part of the system was responsible for aggregating calendars from other services and managing them within the Meeting Scheduler. Since there were numerous properties that the system should maintain, the stakeholder had emphasised an idea of formally modelling these properties. Therefore, the development of the Meeting Scheduler differed quite much from a usual project development in a sense that, apart from regular coding, it comprised formal modelling.

The system required a Client-Server Architecture: a front-end web application and a back-end service, which is presented in Figure 1.

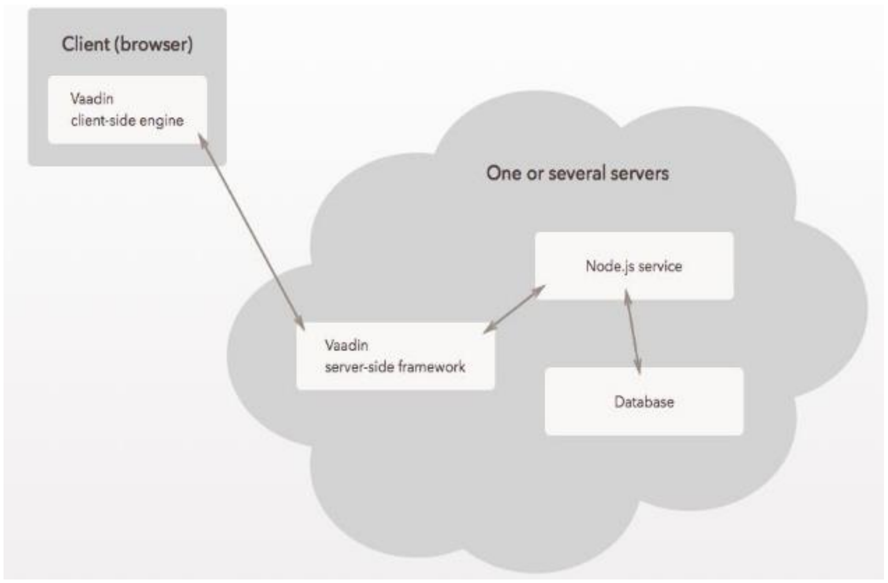

Figure 1 High-level view of architecture of the Meeting Scheduler

Scrum [10] had been chosen as the agile development process. Most of the technologies to be utilised were predefined by the stakeholder:

- Vaadin [18] as a framework for UI,
- Event-B and Rodin tool [19] for formal modelling,
- CalDav protocol [20] for the calendar aggregation,
- PostgreSQL [21] for the database.

The back-end technology, Node.js [22], was chosen according to the preferences of the developers. It is responsible for the application logic: scheduling of meetings, registration of users etc. It retrieves and processes the information through an SQL database server.

Potential users of the meeting planner are anyone who needs to meet with other people at irregular times. A plethora of people uses different calendars in various contexts – private ones or assigned by the organisation they are affiliated with. This application is to help with handling this complexity: the aggregation of different calendars to determine availability provides the relevant information for scheduling a meeting in one place.

A simplified functionality of the Meeting Scheduler is as follows. Users registered in the application can host meetings, suggest some alternative dates and times and invite users. Invited users are notified by e-mail with a link to the meeting. The link allows a user to vote for the available times even without being registered in the application. A registered logged-in user can add URLs to other calendars supporting the CalDav protocol. This way the application knows at which times the user is busy, based on other calendars. This allows the application to give the user only the relevant time suggestions for meetings to which he or she is invited.

The application would only approve the actual time for a meeting when a time suitable for all invitees has been found. Possibly some lesser constraints can be implemented as well. Users can be assigned roles in meetings and can comment on and discuss a meeting they have access to.

3 Supporting Methodologies, Processes and Tools

Project course, by definition, groups people into teams of various expertise and backgrounds. In this manner, their skills and knowledge acquired prior the course are combined to accomplish the best possible outcome – a fully working, innovative and viable product that potentially compete on the current stage of ICT solutions. Therefore, numerous methodologies, processes and tools are combined to realise this undertaking. This section describes the practices and means that were utilised in the development of the Meeting Scheduler.

3.1 Scrum

In this work, we chose Scrum [23] as an iterative agile development framework. Scrum is based on frequent releases and short development cycles. As one of the processes within agile philosophy, it supports process improvement. Scrum provides a well-defined platform for interactions between the developers and the stakeholders, as well as enables good control over the development. Since some of the characteristics conceptually overlap with the ones of Event-B (e.g., iterations and refinement steps), we consider the integration of the two to be seamless.

There were several reasons for which we chose Scrum as the development process. First of all, we identified it as the most suitable for the Event-B developments (see FormAgi framework [24]). Scrum has a clear definition of the time frames for iterations (organisation of sprints) and the set of meetings to be held during the development process [25], which provides certain degree of control over the progress of the development.

Second of all, the Project Course with its setting serves not only as a platform for developing a viable ICT solution, but also as a material for our research. We consider it as a continuation of our exploratory work on the synergy of agile and formal methods. Previously we have experimented with Scrum in the academic setting with experienced Event-B developers [26]. Now we want to study the opportunities and challenges encountered by having a project run with Scrum development process, where the developers are new to Event-B. Moreover, the Event-B development is considered only as a part of the project, where the vital system properties need a model with proof.

An overview of a sprint within Scrum is shown in Figure 2. During each sprint, the development team takes a set of features from the product backlog (listed in the issue tracking system) which holds a set of high-level requirements. Then the stakeholder informs the developers of the features that should be completed in this iteration. From this subset, the team selects the features that are realistic to be implemented. Next, when the goals for the sprint are determined, they are shifted to the sprint backlog, which remains fixed at the time of the sprint. The sprint lasts for a specific amount of time (2-4 weeks) and at the end, it produces a potentially shippable product increment, which is presented by the team to the stakeholder. Any of the unimplemented features are returned to the product backlog.

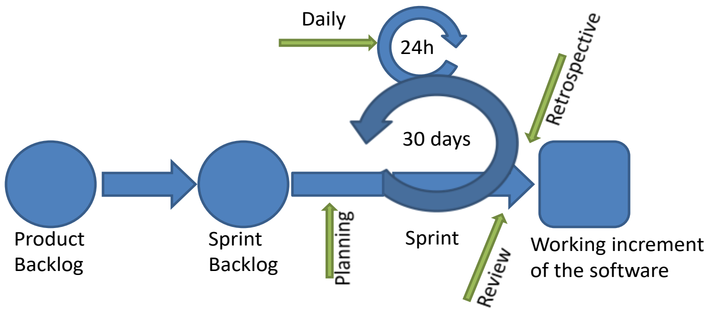

Figure 2. The overview of a Scrum sprint

Communication, one of the cornerstones of the agile approaches, is also important within Scrum. Various meetings are held throughout the development: before the sprint starts (Planning), during the sprint (Daily Scrum), after the sprint (Review and Retrospective). The retrospective is a process improvement-oriented meeting, while the other meetings concern the development itself.

There is a strong involvement of the representative of the end user (or the stakeholder) in Scrum, so that the directions for further development can be indicated at the end of each sprint. The issue of how much functionality will be implemented during each iteration is controlled and decided by the development team. The contents of a sprint do not change during its duration. Thus, the moments at which functionality changes can occur are limited and well-defined.

The goal of this development methodology is to increase the relative effectiveness of development practices for the improvement purposes. At the same time, the delivery of a framework for the development of complex products has to be accomplished [10].

3.2 Event-B Modelling

Application of formal methods ensures correctness of the system (or part of it) that is being developed already at the early stages by formally modelling and proving system properties. It is particularly important in the safety-critical domains such as medical, military, transportation or airspace [13]. Learning formal methods by students prepares them for the work in industry, where the formal modelling skills may be required. Finally, the skills can be useful when tackling the system requirements, regardless of the criticality of the application domain.

Event-B [16] is a formal method and modelling language for stepwise system-level modelling and analysis, based on the Action Systems formalism [27] [28] [29]. It is derived from the B-Method [30], with which it has several commonalities, e.g., the set theory and refinement. Event-B is dedicated to model complete (reactive) systems, including hardware, software and environment [19] and has gained appreciation in industrial settings [31].

An Event-B specification uses a pseudo-programming notation – Abstract Machine Notation – and consists of dynamic and static parts – a *machine* and a *context* respectively. The formal development starts with the specification of an abstract machine from a set of requirements and proceeds by its refinement in a number of steps. A machine consists of its unique name and has the following constructs:

- a list of distinct variables that give the state space of the system;
- invariants which state the vital properties of the system required to be preserved;
- a collection of events that pose operations on the variables, where "initialisation" is the event that initialises the system.

A more abstract machine can be refined by another, more concrete one. The refinement chain and the modelling process can be easily tracked and controlled.

The static part of the specification is also extended with respect to the development of the machine. It encapsulates the sets and constants of the model with their properties given by axioms and theorems.

An Event-B machine can refer to a context through the "sees" relationship. The relation between machines and contexts, as well as the refinement relation for these, is presented in Figure 3.

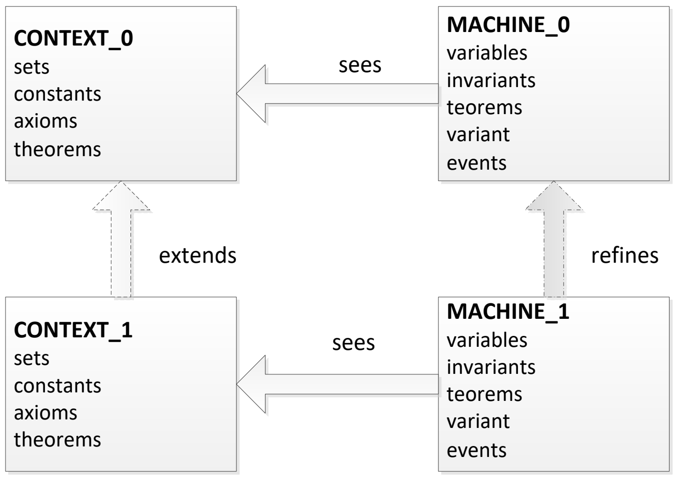

Figure 3 Refinement in Event-B (following [16])

Event-B utilises refinement (described later) to represent systems at different abstraction levels. Refinement enables the developers to gradually introduce details into the constructed system and to represent new levels of a system with more functionality. Mathematical proofs are used to verify consistency between the refinement levels. This is supported by Event-B and provides rigour to the specification and design phases of the development of critical systems. It is effectively supported via the *Rodin platform* [32]. Rodin is an Eclipse-based tool (an open source "rich client platform") extendable with plug-ins. For instance, there are plug-ins that provide the model simulation and animation, as well as its visualisation. Finally, the model can be used for the source code generation into various programming languages.

Refinement [33] [34] [35] [27] is a stepwise approach to the system development, which allows developers to iteratively create the system following certain rules called *refinement rules* (also referred to as proof obligations) [36] [37]. The *stepwise refinement* is a top-down approach [34] which aids handling all the implementation matters and *complexity* by splitting up the problems to be specified and gradually introducing details of the system to the specification. In the refinement process (presented in Figure 4), an abstract specification, that is usually non-deterministic and capturing system level properties, is created from the requirements. It is then transformed into a more concrete and deterministic system that preserves the functionality of its specification in consecutive refinement steps. Each refinement step is supported by an invariant that states the properties of the system. Hence, the invariant is also created in an iterative manner.

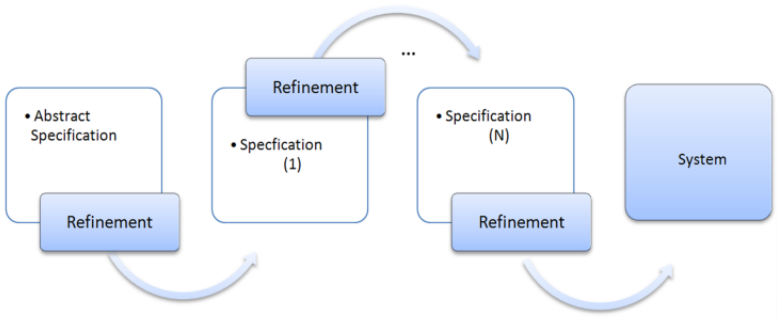

Figure 4. Refinement process

This approach to the system development results in a system that is correct by construction [33]. The correctness of each refinement step is ensured by the mathematical proof of the model consistency and feasibility. These are ensured by the fact that each refinement model preserves the invariant. Even if proving is tool-supported, there are still some proofs that cannot be automatically discharged, but will require human interaction. The amount of involvement needed heavily depends on the chosen modelling strategy and is a subject of our current work.

The complexity of proofs depends not only on the problem and the complexity of the system to be modelled, but also on the refinement strategy utilised and, e.g., on the decomposition mechanisms [38]. Therefore, assisting the modelling activity by facilitating the development process would help to deal with the complexity issues.

3.3 Vaadin

Vaadin is an open-source framework for building web applications in Java. It requires a Java servlet container (such as Tomcat or Jetty) to execute the applications in. Once an application is deployed, the framework takes care of capturing events and user actions that happen in a web browser and forwarding them to the Java application code.

The framework contains also a number of graphical user interface components and means of their composition and extension to enable creating responsive and accessible applications. Furthermore, a significant number of components can interact with any data source that is compatible with the Vaadin Data Model. Several implementations of the model are available in the framework by default, so that it is possible to interact with SQL databases or Java Persistence API implementations.

3.4 Other Tools and Technologies

According to the guidelines provided by the Project Course Mentors, version control (SVN [39]) and issue tracking system (Trac [40]), were in use. All features were supposed to be specified as entries in Trac, with priorities and associated milestones etc., as agreed with the customer. Submissions to the main branch of the code repository were to take place once a code was tested and functional. The team was expected to create branches for the development on the as-needed basis.

The programming environment for the front-end was Java 8. The developers were free to choose a development environment they felt most comfortable with.

4 Experimental Setup

The requirements for the project and its execution were given at one of the first lectures of the Project Course. Then, a three-hour Crash Course was organised specifically for the developers, who joined the project, where the introduction to the methods and tools was given. Moreover, the requirements document for the scheduler was presented, as well as the requirements and technicalities were discussed (30th September 2015).

The Crash Course tackled:

- The project and its requirements – customer's perspective
- Methods and tools needed in the project
- Introduction to Agile / Scrum
- Overview of Trac and Vaadin
- Event-B modelling in Rodin (including demonstration)

There was also a time slot dedicated to possible questions and comments.

The project was not only executed for the purpose of the project course (student perspective), but also generated data for an academic experiment (research perspective). Thus, it had to be explained what data will be collected and what type of observations are sought. Finally, the role of the researcher in the project needed to be clarified for ethical reasons.

4.1 Roles

Scrum quite clearly identifies the roles in the project, which enabled us to divide the responsibilities in an unambiguous way. The *customer* (stakeholder) was a *product owner* at the same time. He was the initiator of the project, as he was the one providing the idea for the development.

The development *Team* was international and initially consisted of 5 Master-level students with various experiences in programming. The division of responsibilities in the project was left to the Team. However, it was emphasised that the Team should be cross-functional, with shared responsibility and should have one goal in mind – to produce a shippable system.

Initially, the responsibilities within the project were divided into four sub-teams, each responsible for the front- and back-end development, formal modelling and databases. The distribution of work given in the following listing shows that the Team organised the work in a careful manner, having in mind that there should not be a task left without the contingency plan, i.e., the members typically had a "double responsibility". It is justified by the various real-life scenarios, such as the one when one of the team members is not capable of working due to, e.g., sickness. The allocation of tasks was as follows:

- Team Lead: back-end and the business and visibility matters (e.g., poster)
- Team member 1: database development and formal modelling
- Team member 2: front-end and formal modelling
- Team member 3: front-end
- Team member 4: back-end and database development

During the project course, two team members quit the project. One member resigned from the project due to some other commitments after three months of the project duration, whereas another one occurred to be inactive and eventually was not considered as a Team member. The personnel changes resulted in leaving the front-end angle of the project without a developer. Moreover, the formal modelling part of the project was significantly weakened, since had a strong background in the formal method to be used. In the end, one member of the remaining team members got assigned to the front-end (previously meant to assist in database and back-end development); formal modelling was left with one team member assigned, so was the database development. Consequently, the team roles were as follows:

- Team Lead: back-end and the business and visibility matters (e.g. poster)
- Team member A: database development and formal modelling
- Team member B: front-end

The development tasks had to be partially sequential, e.g., database design was planned first to provide the infrastructure for the project; only then the focus was put on learning how to formally model the system and proving its properties. This sequence of actions was also caused by delaying the formal development by the team member initially assigned to it (who later resigned). The management and visibility of the project were executed throughout the project, as well as the development of the front- and back-ends, which had to be developed in parallel.

4.2 Process - Schedule, Meetings and Communication

The Scrum process was adapted to the development specifics. The development team worked in a distributed manner. The students were meeting face-to-face mostly for the purpose of the lectures at the Project Course or the appointments with the stakeholder. The team members were working at their own pace, at various hours of the day, depending on their schedule and time preferences.

The Scrum process tailored to the needs of the project is shown in Figure 5. The figure illustrates the product and spring backlog together with the scrum meetings – planning, daily, review and retrospective.

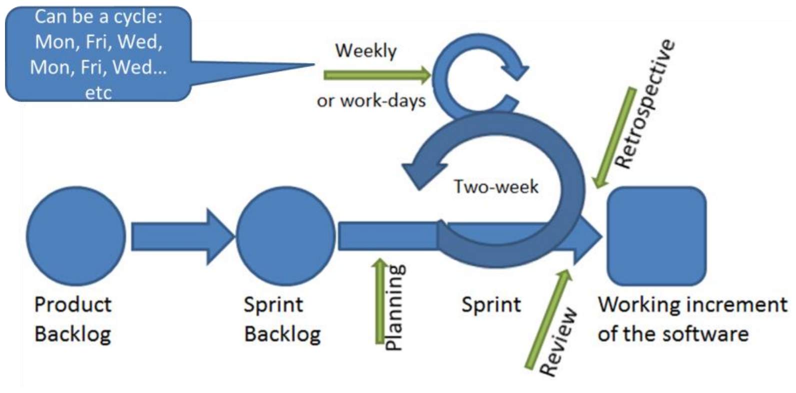

Figure 5 Suggested development process – tailored Scrum

To keep certain level of control of the project, the distribution of the work-load had to be balanced out by the Scrum meetings. It was suggested to keep the "dailies" as a cycle of weekday meetings occurring every second or third day. The sprints were kept as two week periods, at the end of which a meeting with stakeholder was organised. At that point, the current status was checked and the features from the backlog, which were not implemented within the current sprint, were moved to the next one. Furthermore, the planning of next sprint took place and the features to be implemented in the following sprint were agreed on with the stakeholder. Unlike in a regular sprint, the retrospective was not done within the Team. The possible improvements to the current process and the challenges and difficulties that appeared in the previous sprint were enquired by the researcher.

The project took place between 1st of October, 2015 and 1st of April, 2016. A Crash Course Workshop was organised before the official start of the project. During the progress of the project, there were 5 milestones set, including the "preliminary design" milestone, the purpose of which was to obtain the basic idea on how the software system works and how it is done. At that point, all programming languages and methodologies were supposed to be chosen and the repository structure was set up. Long iterations were set for two and three week periods. However, in the beginning and at the end of the project the iterations were weekly. It also meant that some milestones had a few iterations.

Various types of meetings, as suggested by scrum methodology, were scheduled. These helped to manage, control and improve the development. The planning and the review were held when meeting with the stakeholder, whereas the "daily" stand up meetings were organised within the Team, with frequency of 3 times a week (initial setting, then less frequent – controlled by the team lead), often remotely due to the distributed location of the developers.

Table 1 Project schedule with milestones, meetings and special events.

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
41			M				
42					M, D0		
43							
44							D1
45			M				
46			M				
47							
48			M		Event-B		
49							D2
50			M				
51							
52							
53							
1							D3
2			M				
3							
4			M				
5							D4
6			M				
7							
8			M				D5
9			M				
10	M			ICT			
11							
12							
13							
14			Post				

In**Error! Reference source not found.** Table 1 we present the project schedule, where the columns represent the days of the week, the rows represent the numbers of weeks in a year, the cells marked "M" show the days of meetings with the customer, whereas the "D" stands for the milestones (deliverables). In addition, we included some special events in the figure, such as the Event-B workshop (denoted as Event-B)), where formal modelling “learning-by-doing” was realised, and previously mentioned ICT Showroom event. Finally, a post mortem meeting took place after the project ended (denoted as "Post"). The goal was to summarise the project, ICT Showroom experience and the collaboration, as well as collect suggestions for improvement regarding the quality and functionality of the delivered project.

The communication between the team members was mainly based on internet messaging, phone and e-mails. Face to face discussions were difficult due to the distributed location of the members whose main opportunity to meet in person was during the planning and review meetings. However, during the holiday break and the time preceding the ICT Showroom event, the communication was more intensive and direct.

During the retrospectives the possible process improvements were discussed. They mainly concerned the working times of the Team, which vastly varied between the developers. Some Team members worked in the middle of the night, whereas others preferred to work over the day, which made it difficult to synchronise the efforts and organise dailies. Also, the schedule of the Team members differed with respect to their obligations towards attended courses and other university activities.

5 Scheduler Development

The development can be conceptually divided into two parts, which are interleaving, but in practice were executed separately: the formal modelling with Event-B and implementation in Java and database-related techniques. Both are set up within the Scrum development process (see Figure 6). In the following sections, we will describe both, focusing more on the formal modelling part.

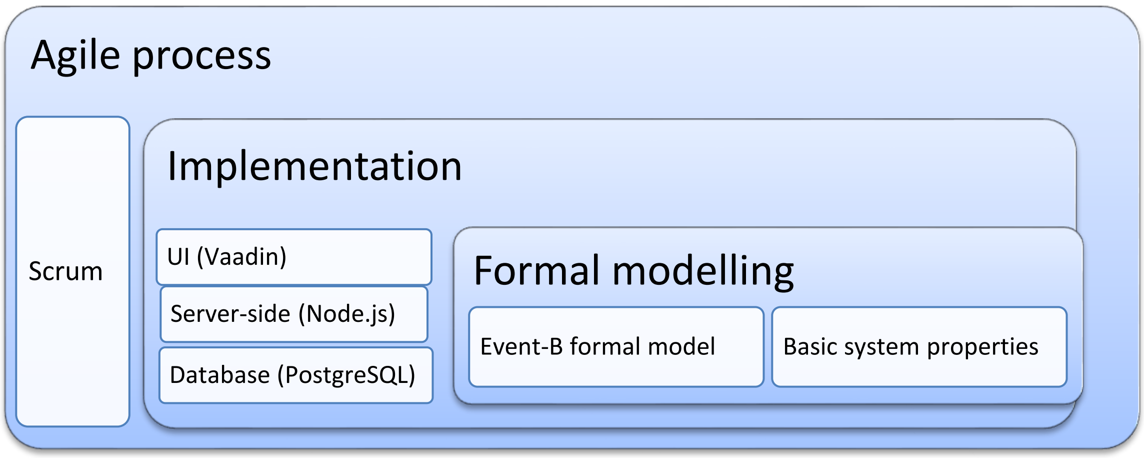

Figure 6 Relation of the methodologies used

5.1 Formal modelling

The formal development was started when the database structures were established, and the basic concepts on the front-end and back-end were implemented. The delay was caused by the need of being able to present the working software to the customer. Although formal modelling was a vital part of the development, it was not integrated with the java implementation (no code was generated from the model, since it was built to prove the basic logic behind the scheduler).

The Team member responsible for modelling and proving took a 2-month course (Specification Methods) prior to the Project Course in order to gain some familiarity with formal modelling. The course provided mathematical ways to solve given problems and notations used to build the respective models. The course was utilising the B-Method, a formalism similar to Event-B. Event-B is derived from B-Method, thus both notations share many concepts, however, they differ in terms of e.g. operator precedences (on the differences between the formalisms, see Rodin Handbook [41]). These differences can be quite misleading, especially for the beginners. The modeller was not familiar with the Event-B formalism and was not aware how to create a model using Rodin. It took roughly 2-3 weeks to learn how to develop a simple model in Rodin.

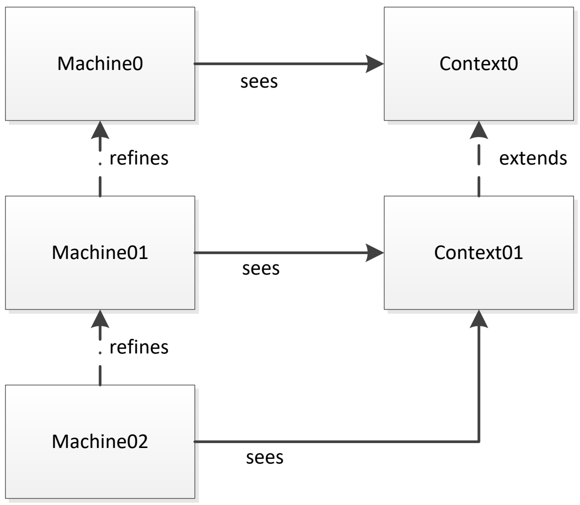

Figure 7 Event-B model – relations between dynamic (machines) and static (contexts) parts

The formal model consisted of 3 machines and 2 contexts. The high level structure of the model is given in Figure 7. Functionality of Machine0 is basically to manage the users of the meeting scheduler system, i.e., to add users to the application, as well as to log them in and out. In the first refinement step, Machine01, the management of the meetings takes place, for instance adding e-mails to the contact list, sending invitation and cancelling meeting. In the second refinement step, Machine02, the voting is modelled. Note that the created model is kept on the high and nondeterministic level, which makes the code generation impossible. Having the model fully developed would require far more effort than it was foreseen for the project course and expected from the students.

All 20 proof obligations were discharged automatically. Most of them were related to the abstract machine (Machine0) – 4 and its refinement (Machine01) – 14. The proof statistics data were taken directly from Rodin tool.

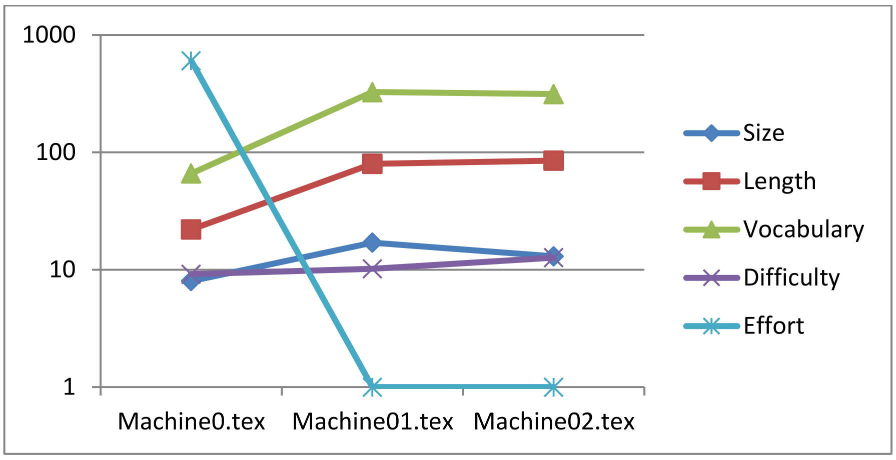

Figure 8 Measurements on dynamic part of the model (machines)

The model was evaluated with metrics based on the syntax of Event-B models [42]. From the model measurements that were collected, we computed the size, complexity and effort required to construct the model. In Figure 8 we can observe that the biggest effort was made to build the abstract machine, whereas the largest size of the model and the biggest diversity of operands were observable in Machine01. The difficulty of building the model was quite steady throughout the development.

Unfortunately, the detailed data concerning the developer's effort and time spent for particular activities when learning the method and constructing the model were not available. The developer in charge of the Event-B modelling, focused mainly on designing and implementing the system. Abandoning or limiting the documentation and reporting were associated with the time pressure (project schedule) and the idea of the working code being a priority.

In post-mortem analysis, the developer stated that the time spent on building the model was roughly 3-4 weeks. Interestingly enough, refactoring of the model, which was advised by the formal methods expert, took one day. It implies that the intensive work with Event-B and Rodin platform together with the acquired domain knowledge, flattens the learning curve and it is much easier and efficient to implement the requested changes.

5.2 Vaadin Implementation

The implementation of the user interface (UI) was done in Vaadin framework. We ran a STAN code analysis tool [43] on the Java code produced by the Team. STAN supports a set of carefully selected metrics, suitable to cover the most important aspects of structural quality. Special focus has been set on visual dependency analysis, a key to structure analysis. The tool helps to identify the design issues and reports on the design flaws.

The UI for Meeting Scheduler consists of one library and one package. The analysis tool listed several issues regarding the size of the implemented methods, as well as the depth of inheritance tree. Moreover, the complexity metric represented as FAT values, shows how big and cluttered are the individual classes and/or packages. It is also reported as a violation. In Figure 9 we present the "pollution chart", which summarises the biggest concerns with respect to the quality of the design according to the STAN report.

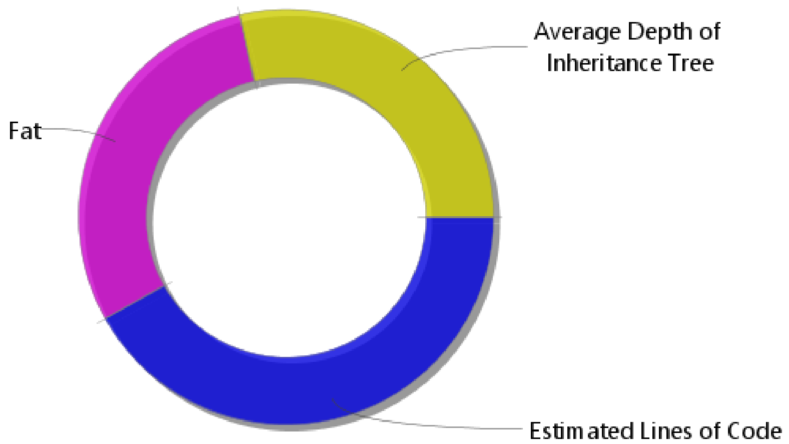

Figure 9 Pollution chart for the UI of Meeting Scheduler

The stable abstractions principle [44], also calculated by the STAN tool, points out a relation between two package measures: the abstractness of a package, which expresses the portion of contained abstract types, and its stability, which indicates whether the package is mainly used by other artifacts (stable) or if it mainly depends on other artifacts (instable). A package should be as abstract as it is stable, meaning that one should avoid packages which are used heavily by the rest of the application and which, at the same time, have a low degree of abstraction. Such packages are a constant source of trouble, since they are hard to change or extend [45]. The goodness of the dependencies of packages can be displayed on the chart as a distance metric, where being on or close to the line of "main sequence" is desired. The line falling diagonally from the top left to the bottom right signifies that packages with a low degree of instability should have a high degree of abstractness and vice versa.

The only package in the UI development is placed in the upper-left corner of the distance chart, which shows that the package is concrete and instable. This in consequence means that in case of any changes in the user interface, everything will need to be changed in the very package, which may involve complications (ripple effect).

The implications of this are the following. The UI code will be difficult to modify or extend because of its monolithic structure, lengthy code in methods, as well as the complexity of relations between methods. In case of further development of the Meeting Scheduler, as was foreseen by the Team, the UI will need to be re-engineered to be manageable and could be used as a basis for the product evolution.

6 Observations on the development

Based on the diary from the development and post-mortem unstructured interviews we identified the following remarks. We divided them according to the perspective of the development Team and the stakeholder. We separately analyse the impact of Scrum on the project, as we believe it vastly influenced not only the development itself, but also its social aspect.

The development Team found it exceptionally useful and important to have the preparatory activities for the project. In particular, the Crash Course covering the basics of agile (Scrum) process and other methodologies and techniques occurred to be beneficial. Since formal modelling was one of the requirements for the project and was considered as one of the most difficult parts, due to the learning curve, it also required an introductory example-based lecture. It was followed by a two-hour hands-on workshop, when the actual modelling was to start. Afterwards, it was followed by "on-demand" iterations, whenever there were some major difficulties encountered.

The team member, who was assigned to the formal modelling, reported problems with the availability of learning materials, which was one of the reasons of the need of consultancy with the formal methods expert. Moreover, some of the materials were dedicated to a specific operating system, which was quite limiting. One of the main comments about the formal modelling part of the development was that the learning time needed to obtain the basic working-idea of the method was significant, in case of this project roughly 3 weeks of intensive learning, which confirmed the estimates of the researcher. Finally, the Handbook of Rodin, which is a tutorial on the use of the Event-B method within the Rodin platform, was considered difficult to follow, specifically for a person who does not have strong mathematical background and is only beginning with the formal methods as such.

The developer assigned to modelling pointed out that the formal modelling done prior to the implementation would benefit the project. He reasoned that the modelling helps to understand how the system should work and what kind of properties and functions need to be considered later in the implementation stage. This in turn, makes the developers aware and early warned about possible problems and requirement inconsistencies which may appear in the development phase. He also emphasised that this approach is feasible for those who already have some experience in working with Event-B and the Rodin platform.

Yet another claim made by the Team was that the tools used in the project were not working well together and also with the process. Instead of supporting developers' work, they required an extra effort. The developers stated that the use of the Git version control instead of SVN could have smoothened the development. Different philosophy of having a local copy and then pushing it to the main branch of development could have helped with the steady pace of the development. Furthermore, Trac was hardly ever employed, i.e., it was used at the beginning of the project and when the code was requested by the stakeholder. Later, especially in the final two months of the project, the Team members considered the documenting the features to be implemented as an activity of lowest priority and concentrated on the implementation activities.

One of the missing elements of the Crash Course, and Project Course in general, that was considered by the Team was a short introductory lecture on how to handle the requirements given by the stakeholder, i.e., how to organise them, so that they are grouped in backlog. Since the requirements for the project were given in a non-structured form, they needed to be processed by the Team, discussed with the stakeholder, refined and specified in more detail. This was regarded as quite a challenging task, in particular when considering the formal modelling, when the properties to be modelled have to be clearly defined. Agile process enabled the Team to better work on the requirements, i.e., timely identify and resolve the inconsistencies and uncertainties.

There were many risks and variables that could not be controlled during the project. The academic context of the development was a source of additional risks that might have never occurred in the industrial setting. Some of the risks, such as resignation of one of the team members, were identified at the beginning of the project. The possibility of team members quitting the project is much higher in the academic setting, since there are no real consequences for such actions (except of not receiving the points from the course; however, the course can be taken in the following edition). Nonetheless, when there are changes in personnel, adjustments are needed on an individual and managerial level. For instance, when the team members leave the group, it not only hampers the schedule, but also forces the team to divide their resources anew. It also has a de-motivational factor and largely hinders the possibility of a successful outcome.

There were no specific testing strategies, code reviews or code quality evaluations implemented. This made the evaluation of the project with respect to its quality, while it progressed, impossible. The team employed in vivo (black box) testing, which, by nature, does not capture most of the defects. The stakeholder suggested having unit tests as the *quality assurance* mechanism; however, they were not employed. The only solid quality assurance technique used was formal modelling in order to prove some of the system properties. However, it was done after the logic behind the system was implemented.

Agile development process played a vital role in the student project. We analyse it according to the terms of the Manifesto for Agile Software Development: (i) Individuals and interactions, (ii) Customer collaboration, (iii) Working software and (iv) Responding to change.

Individuals and interactions

The idea of self-organising teams could not be fully implemented in the student projects. There was a lack of high-level vision of the project and it seemed difficult to divide and prioritise tasks. It was also related to the fact that the project course was not the only item in the students' schedule. For instance, as observed by the developer assigned to formal modelling, it would be much easier to first model the system and its properties and only then get to the implementation stage. This, however, could not be fulfilled by the Team.

Moreover, the schedules and availability of the Team members differed and it was difficult to synchronise the meetings, which in consequence hampered the communication and information flow. Different work-times, e.g., working overnight or not working on the weekends, lead to further problems with coordination. The development very much resembled the global system development, where the Team is spread with respect to location (here within one city, though) and work-times, as well as diverse when it comes to cultural and behavioural characteristics (international team). The obstacles that are known to be brought by the global development were, in this case, highlighted even more than in the real life, industrial setting.

The frequent meetings embedded in the agile process enabled the Team and the stakeholder to observe the progress of the project. This, in consequence, helped with sustaining the motivation of the Team. Moreover, it kept customer informed, since he has been continuously updated with how the project evolved by seeing the working version of the software in each sprint.

Customer collaboration

Scrum was strengthening the interaction not only within the Team, but also with the stakeholder. Moreover, it was enforcing the deadlines and putting the pressure on continuous progress and sustainable pace of the development. Since the stakeholder was coming from industry, the format of the meetings, the way the technicalities were discussed, the schedule, the time pressure and the demand level resembled the real-life environment. The meetings were more frequent than the ones scheduled for the Team by the schedule of the lectures alone.

The collaboration with the customer also helped with the elicitation of requirements. This has ultimately contributed to the implementation of the software with the desired functionality, taking the schedule and team capabilities into account.

Working Software

As mentioned from the Team perspective, the *documentation and reporting* were given the lowest priority, although it was emphasised at the Crash Course that the agile development philosophy does not discard documentation. The documentation was limited to the minimum required by the lecturers of the Project Course (project plan, progress reports, business pitch, user documentation, post-mortem presentation, etc.), whereas the documentation within the code was reported for the server-side.

The stakeholder was highlighting the need of providing the documentation, especially, that the developers were expressing the will to continue after the course was finished. In this case it would be put the system into to production and later maintain it.

The issue tracking system was managed and updated not when the request of a feature or a bug to be fixed was reported. Rather, it was used after the feature has been implemented, thus making it impossible to track and manage the project. Furthermore, this course of action excluded the stakeholder from the development, as he was unable to observe its progress. Updating the issues was considered as infeasible for the development and, as noted by the Team, required additional effort that did not explicitly lead to working software. Thus, using it was considered as a "waste" in the agile sense.

There was an issue with *scheduling* the actions and features to be implemented in sprints. Since the Team was new to many techniques and just started working together, it was very difficult to make good estimates for the development. As a result, it was observed that the first 3 sprints were undemanding and could be filled additionally with some other tasks. Note that the Team was still intact at that time and consisted of 5 developers. The difficulties appeared when the number of active team members reduced and there was a need to re-assign developers to other tasks, which required from them to use their time on learning. This was not anticipated when prioritising and structuring the requirements with respect to schedule. It led to project delays and an immensely filled in the Team members' timetable at the end of each sprint. It was estimated that roughly 80% of planned features were implemented in each sprint. As observed by the Team, the haste would not have happened if the persons involved in the development were skilled and experienced, since they could put all their effort into the development (as opposed to learning new methods).

The technical problems related to the use of new technologies and methods became extremely apparent when the system was integrated. This surfaced each time the working version of the system was to be presented to the stakeholder, as well as during the final preparations for the ICT-Showroom. The integration problems were also related to the lack of the documentation, e.g., for the protocol connecting various databases with the server, as well as and learning new technologies. Agile facilitated the integration in terms of dealing with the issues regularly, which prevented the big-bang integration and the related problems.

Responding to Change

Due to the fact that the students were learning the new technologies (Event-B, Vaadin, some communication protocols) while developing the system, it was difficult to tackle the changes in the requirements, regardless if they came from the stakeholder or were a result of misinterpretation of the requirements document. The agile process did not contribute to managing the changes in a more efficient manner. It was mainly due to the academic context of the study where the students were gaining experience and skills at the same time as the project was running.

Moreover, the need of distributing the effort and focus not only on the project (development and learning), but also on the study plan meant that the Team was hastily jumping between the tasks. Learning new methods and tools in an industrial setting would be much more straightforward, since it would most likely be scheduled within the responsibilities and/or work description. Usually, the industrial setting allows the developers to be dedicated to specific responsibilities. The developers, in this case, are driven by the idea of sustaining the job (financial motivation), self-development (acquiring new skills), contributing to the common progress and result of a Team, and thus empowering the Organisation. On other hand, students have obligations on various courses which typically have a different focus. Therefore, the time and resources of students are scattered over diverse topics, sometimes preventing students from executing certain tasks in an excellent manner due to distraction.

7 Lessons learnt and recommendations

In this section we collected the guidelines for the students taking part in group projects, which use Scrum and/or formal methods. The advices are based on the experiences of the Team members and the views of the stakeholder. The recommendations were collected at the post-mortem interviews.

7.1 Team Perspective

Crash Course occurred to be very constructive to give proper foundation and background for the project. It was also considered useful by the Team members. In particular, the introduction to formal methods and Event-B supported by practical example in Rodin gave some level of comprehension of why the rigorous part is required in the project and what formal modelling actually involves. Furthermore, the overview of agile principles and practices, with the special focus on Scrum was regarded as valuable. It laid a common understanding of the development process to be used. Before the Crash Course, some of the Team members were not familiar with agile philosophy, while others were uncertain about specifics of Scrum.

To support the development, a toolchain is typically used. It should fit the needs of the team, as well as the purpose of the project and align with the expectations of the stakeholder. In this project the tool used for code revision was considered not feasible and there was no motivation within the Team to use issue management system. Finally, the Rodin tool did not fit well in the toolchain, which was apparent already at the point when the Team decided to postpone the formal modelling until some work on the implementation of the database, user interface and back-end was done. The system properties were modelled once they have been dealt with on the implementation level.

Modelling using formal methods within a group project could be improved in terms of efficiency. There are two scenarios to consider:

(i) if there is a developer familiar with the formal methods, they should be the one starting the work early (to precede the implementation executed by the other Team members) and focus solely on processing the requirements needed for the model development and modelling itself;

(ii) if none of the Team members is familiar with the formal modelling, all of the Team members should be involved in requirements processing and their formulation in such way that they can be modelled formally.

In the latter case the Team member, who feels most suited and skilled to formally model the system focuses exclusively on this activity, while the others start the implementation. Naturally, the Team members need to consult and integrate their results, so that the properties modelled are the same as the ones being implemented. Code generation from formal models, which is enabled by one of the plugins to Rodin platform, could be a solution in this case. However, it was not utilised in this project (the model created was too abstract to be used for code generation).

Quality assurance mechanisms are certainly needed in such an undertaking. Although formal modelling was included in the development, it only covered a part of the whole system (the logic behind the scheduler). To assure the quality of the produced code, the source code requires unit testing and analysis. The unit tests and code analysis tools should be placed within the scrum process.

In order to manage better with the system *requirements*, the requirements document should, according to the Team, have a more structured format, enriched by visualisations. The tabular format, listing features to be implemented with their properties, would be particularly useful for the development and, in particular, modelling. The scenarios that were present in the requirements document given by the stakeholder were useful for the testing stage.

7.2 Stakeholder perspective

There were a few problems typical for industrial software projects identified by the customer. Focusing on remedies to these problems, while teaching agile, might give a better understanding of the development process by the students. Therefore, it can help to better prepare them for the challenges in their professional career.

The discrepancy about the decision making process was found to be the most problematic. The Team was at times reluctant to contact the customer about their doubts, instead choosing to implement the solution they saw as optimal. While it was true with respect to the implementation details, it did not hold for business-related matters – and the Team clearly had problems differentiating between these two. This behaviour was probably caused by two factors. The first factor is the lack of experience with software development that lead to incorrect separation between what is a business-logic decision and what is not. The other factor is the wrongly understood agile philosophy, in which – by common misconception – a plan and vision for the developed software are not needed.

The other significant problem that directly affects the software and any of its future development is the lack of documentation and design decisions. This is especially evident in the parts developed in the final weeks of the course, as the Team opted to focus fully on the implementation of the remaining functionality. This issue could have been solved in many ways – of which emphasising the need for documented code during the course appears to be the most useful.

Agile methods, in particular scrum, need to be adjusted to the specifics of the academic context, where students cannot work full-time on the project and are not dedicated solely to the project development. Furthermore, when learning new methods, techniques and frameworks during the project, it is difficult to manage the progress of the project and learning process at the same time. In this context, either the students should have a strong background and skills of the methodologies to be employed in the project, or the learning process should be scheduled at the beginning of the project (as first and/or second sprint) and the development should be shifted for later sprints. Moreover, the definition of working software or consumable (as in the DAD [46] taxonomy) being the measure of the progress should be redefined.

The academic context of the student projects brings additional risks other than the ones in the industrial setting. The agile way of working seems not to be suitable when the student project is not the sole activity of the Team. Since the schedules of the students differ, so does their availability. Moreover, the coordination of tasks can become an issue (continuous system integration). All of these factors combined with, e.g., one of the Team member leaving the team, impact the commitment and motivation issues much stronger than in the industrial setting. Thus, the agile process should be adjusted in terms of interaction and effort distribution (i.e., be project-specific).

8 Conclusions

We are aware that the investigation presented in this paper is quite specific for the context that it is set in. However, we believe that the observations and recommendations that we made, will be applicable and useful to other student projects on a master-level of IT education. In this section, we emphasise the motivation for this study by presenting the related work. Furthermore, we describe the threats to validity in our enquiry. Finally, we conclude with some general remarks.

8.1 Related work

Since the agile philosophy, values, principles and methods have been present for 15 years now, they have also put their mark on education. In this section, we present the works that are preceding our investigation and are related to the observations in our setting to a various degree. It should be noted that one of the papers we came across was using at the same time Scrum and formal methods within student project course.

The impact of using the agile practices for student software projects in computer science education was investigated in [47], where students were using a Pair Programming method. An enhancement was noticed, especially in terms of programming skills and final course grades in general. Although we describe the student project course in our work, where the Scrum method is utilised, our observations corroborate the improvement of skills within the Team. Additionally, we are oriented towards managing the development process. Thus, we do not put focus on the aspect of the upgrade in the course final results.

In [48], the authors proposed a hybrid methodology based on Extreme Programming method and Throwaway Prototyping method. The methodology emphasised the need of users or customers being an integral part of the development team. The results showed that the agile methods can be easily tailored to fit the needs of students at the courses. In our work, the customer was very much involved in the development without being considered as a part of the Team. Nevertheless, both publications, [48] and our work, confirm that the flexibility of the agile methods makes them feasible to be successfully incorporated in a student project.

The application of Scrum in student projects when developing three applications is described in [49]. The authors perceive Scrum as a framework for managing projects and introducing good software engineering practices. In contrast, our observations consider one large-scale project executed within 7 months, where the project has an external industrial customer. Nonetheless, our observations overlap in the underlying motivation for using Scrum, as well as in terms of the need for Scrum to be fine-tuned when it comes to the assignment of roles in project and duration of sprints. We fully agree that the students benefit from the flexibility of Scrum. They also gain skills and knowledge when utilising the agile practices related to requirements engineering, project planning and tracking, testing and effective team collaboration.

The use of agile methods in software engineering education has also been described in [50], where the three teams were developing competing electronic websites for the period of 13 weeks. While the students were managing well with technical aspects of used methodologies and agile methods, the social aspects (teamwork and customer collaboration) needed more attention. Our work focuses on one Team working on a scheduler, where the Team members are familiarising themselves with new methodologies and frameworks (Event-B and Vaadin), and at the same time learning how to work applying agile values and practices (Scrum). Furthermore, the teams in [50] based their work on general principles, practices and values of agile, only sometimes presenting elements of Scrum, while we focus solely on Scrum and its adaptation in the student project.

The experiences from student projects using agile practices in software development are given in [8]. There, the student teams developed a project over 12 weeks, were required to use the agile practices and be exposed to an agile spirit (they did not need to get a fully-developed system). Our work describes the development over roughly 26 weeks, where as a result students have to produce a working product (which gives additional pressure to the Team). The objectives of collecting experiences differ quite much between [8] and our work. Namely, the former focused on familiarising students with agile philosophy, agile values and principles, and observing adaptation of agile in the academic setting. The latter focuses on how feasible an agile method, in particular Scrum, can be for the Project Course, where the development environment is mimicking the real industrial conditions, by default accompanied by the project risks and time pressure.

Finally, to the best of our knowledge, there have been no publications describing the use of formal methods and formal modelling within student projects, where students worked in a team, that would also involve agile development methods. This makes our study unique in terms of the experimentation that already exists within software engineering.

8.2 Threats to validity

We are aware about the limitations of our work, e.g., the sample size in the experimentation, human-factors, lack of quantitative data for effort or quality collected during the development, etc. This section describes the threats to validity according to the scheme presented in [7].

Since there was only one team being involved in the study, the significance of the study is limited and the results cannot be generalised outside the scope of our study. The recommendations, by no means exhaustive, are based on a one student project, where initially five students were involved. Therefore, to collect a set of principles and tactics for teachers and students on using agile methods and formal modelling in academic setting, more in-depth investigations are necessary.

Given that we are dealing with the students (as opposed to experienced developers), there are additional factors that have to be taken into account during the experimentation and when summing up the observations. Moreover, all benefits and drawbacks of the used approaches are more apparent, i.e., they are visible as big successes and failures. We reason that less experienced students are more likely to struggle with new technologies and methods. Thus, they have to spent more time and effort on the development for the successful outcome of their project.

In academia, and particularly in this investigation, we were quite limited to the choice of participants of the study and the sample size (one team). Naturally, having multiple teams performing the same development with the same technological means would provide us with the data for comparison and would aid in generalising the results. In such case, human factors (skills, prior knowledge, commitment, etc.) would be the only ones posing a threat to validity. It should be noted, though, that the environment of the study was as planned (academic setting) and the timing did not affect the results (regardless of the time of the year, students would have other academic commitments that impact their availability for the project course).

Since the crucial factor for the experiment to bring valid results is people, we needed to be objective with our observations. We documented special events in the investigation (Team member leaving the team, Event-B workshop, meetings with the customer), which could impact the project. Furthermore, there was a "mortality" aspect, which could affect our investigation. In the described experimentation it was when two Team members dropped out from the project. The situation was well handled by the Team by reorganising the work-tasks. It has, however, impacted the collected data (effort distribution, time-pressure factor, etc.), which is well documented in this paper.

Finally, there is the factor of performance of people being observed, which is also called evaluation apprehension. It means that people have the tendency to perform better when they are observed and evaluated. In the case of this project, students were trying to perform as good as possible in order to fulfil the expectations of the course and earn the highest possible grade. They were aware of being observed and what is the purpose of the study (ethical reasons), but they did not seem to act differently. It may be explained by the fact that the environment of the experiment resembled the real-life IT project (including stakeholder, who had no prejudice or expectations to the experiment whatsoever).

8.3 Final remarks

We believe that learning and refining new methods and approaches are the key drivers in progress of IT in general. Furthermore, we consider learning by doing as an effective learning method. These factors provided the motivation for this work to equip students with the background on the up-to-date techniques and enable their application, all in a controlled environment mimicking the actual software development. Generally, agile methods fit well in this context, as they provide a tailorable development process. However, some reflection is needed to fine-tune agile processes (here Scrum) to the specifics of the academic setting (different schedules, availability, distributed environment, resources, background and skills, etc.). Similarly, the idea of utilising formal methods (here Event-B) in the student project needs some additional attention, due to the steep learning curve and producing artefacts other than executable code or project documentation.

Acknowledgements

This work was carried out within the project ADVICeS (https://research.it.abo.fi/ADVICeS/), funded by the Academy of Finland, grant No. 266373.

References

1. Manifesto for Agile Software Development, http://agilemanifesto.org/
2. Paternoster Nicolò, Giardino Carmine, Unterkalmsteiner Michael, Gorschek Tony, Abrahamsson Pekka, *Software development in startup companies: A systematic mapping*, Information and Software Technology, 56 (2014), pp.1200-1218.
3. Laanti Maarit, *Agile Methods in Large-Scale Software Organizations - Aplicability and Model For Adoption*. PhD thesis, Oulu University (2012), pp.192.
4. Ge Xiaocheng, Paige Richard, Polack Fiona, Chivers Howard, Brooke Phillip, *Agile development of secure web applications*, Proceedings of the 6th International Conference on Web Engineering. ACM], Paolo Alto (2006)
5. Lankhorst Marc, *Agile Service Development. Combining Adaptive Methods and Flexible Solutions*. Springer Publishing Company (2012), pp.204.
6. Wolff Sune, *Scrum Goes Formal: Agile Methods for Safety-Critical Systems*, Formal Methods in Software Engineering: Rigorous and Agile Approaches (FormSERA). IEEE, Zurich (2012)
7. Wohlin Claes, Ruenson Per, Höst Martin, Ohlsson Magnus, Regnell Bjorn, Wesslén Anders, *Experimentation in Software Engineering*. Springer (2012), pp.248.
8. Schneider Jean-Guy, Vasa Rajesh, *Agile practices in software development - experiences from student projects*, Proceedings of Australian Software Engineering Conference. IEEE, Sydney (2006)
9. Beck Kent, *Extreme Programming Explained: Embrace Change, 2nd edition*. Addison-Wesley Professional (2004), pp.224.
10. Schwaber Ken, Sutherland Jeff, *Scrum. The Official Guide.*. Scrum.org (2010).
11. Sanders Dean, *Using Scrum to Manage Student Projects*, Journal of Computing Sciences in Colleges, 23 (1) (2015), pp.79-79.
12. Holloway Michael, *Why Engineers Should Consider Formal Methods*, AIAA/IEEE16th Digital Avionics Systems Conference. (1997)
13. Butler Ricky, *What is Formal Methods?*. In: *NASA LaRC Formal Methods Program*, (2001).
14. Sobel Kelley, Clarkson M., *Formal Methods Application: An Empirical Tale of Software Development*, IEEE Transactions on Software Engineering, 28 (3), pp.308-320, (2002).
15. Baruah Sanjoy, *The Modeling and Analysis of Mixed-Criticality Systems*, Proceedings of Formal Modeling and Analysis of Timed Systems: 12th International Conference (FORMATS). Springer, Florence (2014)
16. Abrial Jean-Raymond, *Modeling in Event-B: System and Software Engineering*. Cambridge University Press (2010).
17. Project course 2016-2017, https://abacus.abo.fi/proj.nsf/Webdocs/AFC16D84D723C90AC225800900379305, Accessed February 2017
18. Vaadin, *https://vaadin.com/home*. Accessed February 2017
19. Event-B, http://www.event-b.org/index.html, *Home of Event-B and the Rodin Platform*. (2008), Accessed February 2017
20. CalDav, *http://caldav.calconnect.org/.*, Accessed February 2017.
21. PostgreSQL, *https://www.postgresql.org/*, Accessed February 2017.
22. Node.js, *https://nodejs.org/en/*, Accessed February 2017.
23. Schwaber Ken, *Agile Project Management with Scrum*. Microsoft Press (2004).
24. Olszewska Marta, Waldén Marina, *FormAgi – A Concept for More Flexible Formal Developments*., Turku (2014)

25. Shore James, Warden Shane, *The Art of Agile Development*. O'Reilly Media, Sebastopol (2008).

26. Olszewska Marta, Ostroumov Sergey, Waldén Marina, *Synergising Event-B and Scrum - Experimentation on a Formal Development in an Agile Setting*., Turku (2016)

27. Back Ralph-Johan, *Refinement Calculus, Part II: Parallel and reactive programs. Stepwise Refinement of Distributed Systems*. In: *Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness*, Springer-Verlag (1990).

28. Back Ralph-Johan, Kurki-Suonio R., *Decentralization of process nets with centralized control*, 2nd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, (1983), pp.131-142.

29. Back Ralph-Johan, Sere Kaisa, *From modular systems to action systems*, Software - Concepts and Tools, 17 (1996), pp.26-39.

30. Abrial Jean-Raymond, *The B-Book: Assigning Programs to Meanings*. Cambridge University Press (1996).

31. Romanovsky Alexander, Thomas Martyn, *Industrial Deployment of System Engineering Methods*. Springer Heidelberg (2013).

32. RODIN, http://www.event-b.org/platform.html, *RODIN - Rigorous Open Development Environment for Complex Systems*. (2006)

33. Dijkstra Edsger, *A Constructive Approach to the Problem of Program Correctness*, BIT Nmerical Mathematics, 8(3) (1968), pp.174-186.

34. Wirth Niklaus, *Program Development by Stepwise Refinement*, Communications of the ACM, 14(4) (1971), pp.221-227.

35. Back Ralph-Johan, *On the Correctness of Refinement Steps in Program Development; PhD thesis*. University of Helsinki (1978).

36. Metayer Christophe, Abrial Jean-Raymond, Voisin Laure, *Event-B Language, RODIN Deliverable 3.2 (D7)*. (2005)

37. Waldén Marina, Sere Kaisa, *Reasoning about Action Systems using the B-Method*, Formal Methods in System Design, 13 (1998), pp.5-35.

38. Yeganefard Sanaz, Butler Michael, *Problem Decomposition and Sub-Model Reconciliation of Control Systems in Event-B*, IEEE International Workshop on Formal Methods Integration., Turku (2013)

39. Subversion, *https://subversion.apache.org/*, Accessed February 2017

40. Trac, *https://trac.edgewall.org/*, Accessed February 2017

41. Jastram Michael, Butler Michael, *Rodin User's Handbook v.2.8*. (January 2017).

42. Olszewska (Pląska) Marta, Sere Kaisa, *Specification Metrics for Event-B Developments*, 13th International Conference on Quality Engineering in Software Technology (CONQUEST 2010)., Dresden (2010)

43. STAN - Sturcture Analysis for Java, http://stan4j.com/, Accessed February 2017

44. Martin Robert, *Object-Oriented Design Quality Metrics: An Analysis of Dependencies*. (1994)

45. *STAN - Structure Analysis for Java*. Odysseus Software (2008)

46. Ambler Scott, Lines Mark, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*. 1st ed., IBM Press (2012).

47. Perera G.I.U.S., *Impact of using agile practice for student software projects in computer science education*, International Journal of Education and Development using Information and Communication Technology, 5(3), 2009.

48. Abdulwahab L., Abdalla A., Galadanci Bashir, Algudah Marshal, Murtala M., *Agile Methods for Software Engineering Students Project: A Proposed Hybrid Methodology*, Proceedings of the The Third International Conference on Digital Enterprise and Information Systems. SDIWC, Shenzhen (2015)

49. Reichlmayr Thomas, *Working Towards the Student Scrum - Developing Agile Android Applications*. American Society for Engineering Education (2011)

50. Rico David, Sayani Hasan, *Use of Agile Methods in Software Engineering Education*, AGILE, (2009), pp.174-179.

Appendix A. Requirements document (original)

HTTP-based service centre (application server):

- manages authentication for every request
- ideally, handshake occurs the first time, and some auto-timeout is given (e.g. 30 minutes)
- handles meeting scheduling
- provides CalDAV calendar aggregation
- mix of http status codes and a platform independent format (json?)

2 separate parts of the core of the software:

- Meeting scheduler
- Calendar aggregation

Both of these run available as one service. Any implementation language ok., though web framework preferred. Rails or Sinatra best choices?

Each user has his own timezone. Date and time stamps used in the software include time zone information (i.e. ISO standard with time zone). This is converted to user's local time on the application side.

Application front-ends:

- Web front-end - Responsive would be best, as it supports many platforms at the same time, Otherwise, there must be a dedicated mobile version; Ideally written with Vaadin
- Text terminal (nice to have) - Preferable command line interface; Another possibility to have interactive shell, with keypresses and the like; Written in a script language, e.g. Ruby

Each user has many read-only calendars, in CalDAV format. Scheduled meetings is one of those calendars. Its address is user-specific and includes all meetings of that user.

On the meeting page there can be a free-form discussion (comments) only by the invited people.

Scheduling a meeting:

- User A selects participants of the meeting (user A can opt himself out of the meeting, while still keeping the "meeting host" role - however, such meeting will not show up in his calendar)
- User A selects date and time suggestions
- A meeting page is created for the user to share with other invited users, where they can vote on suggestions
    - The service is used to check if a given date/time suggestion is valid for a given user

    - Already taken spots are greyed out and users are unable to select them
    - Available answers are yes, no, maybe. For the purpose of voting "maybe" means "no", but is shown differently in the date/time grid
- Once all users have given their votes, the meeting is confirmed and no more changes to it are allowed, unless a change in somebody's calendar overlaps with it. In such case, the meeting becomes editable again.
- For the purpose of confirming a meeting, "maybe" means "yes", and no answer means "no".

Nice to have features:

- By default, each meeting is stored in the application's own calendar. While voting, a user could select a calendar (for which the application has writing permissions) the meeting is going to be saved into. In such case, it does not go to the application's own calendar.
- The above option can have its default in user's profile settings.
- There should be an option to allow meeting host to force a certain date/time suggestion to a meeting participant, if the participant had agreed to that. This should also have its default in user's profile settings.
- Text terminal app is nice to have, a standalone command line tool that could be run in Linux shell (Java not the best option here, rather Ruby or other scripting language), and even nicer to have would be an interactive mode ("interactive shell")

Meeting host can provide extra privileges to each user:

- Inviting other people
- Suggesting more date/times
- Removing existing date/times
- Confirming the meeting

Those options can be given only to registered users.

A meeting host cannot be changed. Only the host can remove a meeting, though any user with a given privilege can confirm the meeting.

Voting on a meeting:

- User picks one of the date/time suggestions he is allowed to, i.e. none of it is already booked.
- As long as the meeting is not confirmed, the user can undo or change votes.
- A user does not need an account to vote.

As a user I want to schedule a one-time meeting. I open an application on my mobile, or visit a web page. I do not want to be prompted to enter a password when I am on my mobile phone, and I would like to avoid entering it through the web page, if I have used the service within last two weeks. I select dates and times from a calendar showing current week and the next 4 weeks - I do this by clicking. At any time I can move forward and backward in the calendar to find another date/time suggestion. In the mean time I can invite guests. I do this by either selecting them from a list of contacts that I have already used, or by manually typing email address. Once added, I can intuitively select role of each user or delete their participation by clicking a self-explanatory icon or short text. At any time during my editing I can notify guests about the meeting. This results in an email being sent to their address. I do not want my guests to receive more than one email, and I would like the sending being optional. Note that I can add new guests after sending an email, and they should be notified when I resend the message. Also, removing a person from a list of guests should result in (optionally) sending an email, if the invitation was sent already. At all times there should be a link to the meeting I can share with others by copying it. The link must display an error page of some sort if it is clicked by a person not invited to the meeting. Once I make a modification to the list of guests, the calendar allowing me to pick date/time gets updated - the date/times that ANY of the guests is busy at are marked as "taken". I can still select them, though. The date/times that I am busy at cannot be selected at all, unless I select an option to just organise the meeting, not to host it. In such case, if I return to being the host, unavailable dates for me count as a "no" vote.

The names of invited guests should appear in a table, with columns being date suggestions, so that I can see votes cast by them on a given date/time. The cells that correspond to a date/time that is already occupied for that guest are disabled and cannot be voted on by that user.

Once there exists a date/time suggestion that every invited guest answered "yes" to, I can confirm the meeting. I have an option to send email notification to the guests and the event is entered into their calendars.

When the agreed date/time passes, the meeting cannot be edited anymore.

As a user I want to modify a meeting I am hosting (or I have admin rights to). I open an application on my mobile, or visit a web page… etc. I then click a visible item in the menu that shows the meetings I have created or been invited to. I am shown the same screen as when creating a meeting. When the meeting is already confirmed, there should be an option to "unconfirm" it. Guests receive an email, and the meeting is withdrawn from the calendars. Voting and inviting new people is allowed then, as when creating new meeting.

As a user I want to disband a meeting. I do the same things as if I would like to edit the meeting. Then I click on the button/link "Cancel meeting" and confirm that I want to do it. If an email was sent to guests, an email is sent to inform them about cancellation. This operation is irreversible. The meeting link stops working (i.e. shows information about cancelled meeting and nothing else). The information about guests availability is updated (i.e. the meeting gets removed from the calendar).

As a non-registered user I want to register or log in. I visit the web page, and since I am not logged in, I am shown a form to log in, with email address and password. When my credentials are matching an existing record in the database, I am shown a screen for organising a new meeting. When the password is incorrect, I am shown an option to re-enter the password, or a link to reset it. Resetting is done by sending an email to my account with a link that opens a password reset page. There is an option to enter and retype new password. On successful change I am logged in and shown a screen for organising a new meeting, and an email about successful password change is sent to my address. When the email is not found, I am shown both email and password field - each of them two times (so the form has 5 fields - one of them is a checkbox for accepting terms, conditions and privacy policy), and I am asked to retype them. Once emails and passwords are ok, and terms are agreed to, I am logged in and shown a screen to create a new meeting. In the mean time I receive an email about the fact that I have registered, and that I should confirm activate my account by clicking a link. THE EMAILS MAY NOT, under any circumstance, CONTAIN THE PASSWORD.

As a non-registered user I have been invited to a meeting, and I received an email with a link. I open the web page. I am shown a page in which I can vote on a date/time suggesting, pretty much as if I would be a registered user - with one exception. There is a visible button somewhere that allows me to register. This opens a form for entering and retyping password, and agreeing to terms and conditions. The email field is not required - but it should be shown as a read-only field or label.

A user must be logged in to organise meetings, but does not have to be to accept date/times. Only registered users may be given extra rights to the meeting (inviting guests, suggesting date/times).

As a registered user, I want to update my profile information. I open a web page, and in the menu I click a button to modify my profile information. I can change password and set my name. Email address cannot be changed - but there should be a possibility of merging an account for a different email address. I can also add addresses of calendars in CalDAV format. At most one of the calendars I can mark as the default one to which meetings are written to. By default it should be app's calendar, which should be listed there as well.

A link that is sent to a guest, with the information about the meeting, should include email address. Email addresses of other guests should not be visible to other people (use names instead, or first part of the email).

As a user I want to withdraw my participation from the meeting. I open the meeting page and click "remove from meeting", and then I can specify an optional reason of leaving the meeting. This operation must be confirmed and cannot be undone. The meeting is removed from my calendar. I can still be added by the host to the meeting again, though not if the meeting has been confirmed. The host of the meeting receives an email with the information, and a link to the meeting. Note that I can leave a meeting that has already been confirmed.

Appendix B. Event-B Specification of the Scheduler

8.4 Machines

```
machine Guarantime0 sees GuarantimeContext0

variables members // Members of the Application
     login // Members can login into the system

invariants
 @inv1 members ⊆ PERSON // Members come from the set of PERSON
 @inv2 login ⊆ PERSON // User who can login into the system are alson
belongs to the set of PERSON
 @inv3 login ⊆ members // Only Member can do login in to the system

events
 event INITIALISATION
  then
   @act1 members ≔ ∅
   @act2 login ≔ ∅
 end

 event Add_Members
  any pp
  where
   @grd1 pp ∈ PERSON
   @grd2 pp ∉ members
  then
   @act1 members ≔ members ∪ {pp} // Adding New Member
 end

 event User_Login
  any mm
  where
   @grd1 mm ∈ members
   @grd2 mm ∉ login
  then
   @act1 login ≔ login ∪ {mm} // Member logging into the system
 end

 event User_Logout
```

```
      any mm
      where
       @grd1 mm ∈ login
      then
       @act1 login ≔ login \ {mm} // User Logout from the system
     end
    end
```

▪▪

```
    machine Guarantime01 refines Guarantime0 sees GuarantimeContext01

    variables members login
         email // Set of people with email addresses
         contact_list // contact list of the members
         meeting_host // member who generates the meetings
         meeting_guest // guests of the particular meetings
         suggested_date // Suggested dates of the Meetings
         date // Sets of dates of the month, in this case i have choose only
31 days of the month


    invariants
     @inv1 email ⊆ EMAIL
     @inv2 contact_list ∈ email ↔ members // the person who has email address
can be the part of contact list of more than one member
     @inv3 meeting_host ∈ INVITES ↠ members // one member can generate more
than one meeting
     @inv4 meeting_guest ∈ email ↔ INVITES // one person who has email
address can be invited to more than one meeting
     @inv5 date ⊆ 1 ‥ 31
     @inv6 suggested_date ∈ INVITES ↔ date // more than one meeting can be
hel on single date

    events
     event INITIALISATION extends INITIALISATION
      then
       @act3 email ≔ ∅
       @act4 contact_list ≔ ∅
       @act5 meeting_host ≔ ∅
       @act6 meeting_guest ≔ ∅
       @act7 date ≔ ∅
```

```
   @act8 suggested_date ≔ ∅
 end

 event Add_Members extends Add_Members
 end

 event User_Login extends User_Login
 end

 event User_Logout extends User_Logout
 end

 event Contact_List
  any cc
  where
   @grd1 cc ∈ contact_list
  then
   @act1 contact_list ≔ contact_list ∪ {cc} // Adding email address to
contact list
 end

 event Send_Invite
  any gg mm dd db dc
  where
   @grd1 mm ∈ meeting_host
   @grd2 gg ∈ meeting_guest
   @grd3 dd ∈ suggested_date
   @grd4 db ∈ suggested_date
   @grd5 dc ∈ suggested_date
  then
   @act1 meeting_host ≔ meeting_host ∪ {mm}
   @act2 meeting_guest ≔ meeting_guest ∪ {gg}
   @act3 suggested_date ≔ suggested_date ∪ {dd,db,dc}
 end

 event Cancel_Meeting
  any dd cc mm ii
  where
   @grd1 mm ∈ ran(meeting_host)
   @grd2 dd ∈ ran(suggested_date)
   @grd3 cc ∈ dom(meeting_guest)
```

```
      @grd4 ii ∈ dom(meeting_host)
     then
      @act1 meeting_host ≔ meeting_host \ {ii ↦mm} // removing pariticular
invitation/meeting which user generated
      @act2 suggested_date ≔ suggested_date \ {ii ↦ dd} // making the
particular date free for which the meeting has been canceled
      @act3 meeting_guest ≔ meeting_guest \ {cc ↦ ii} // making guests free
from the meeting which has been canceled
    end
   end
```

```
   machine Guarantime02 refines Guarantime01 sees GuarantimeContext01

   variables members login email contact_list meeting_host meeting_guest
suggested_date date
        votes // vote which invities give for a meeting


   invariants
    @inv1 votes ∈ email ↔ Votes // people who have email address can vote on
meetings in which they have been invited

   events
    event INITIALISATION extends INITIALISATION
     then
      @act9 votes ≔ ∅
    end

    event Add_Members extends Add_Members
    end

    event User_Login extends User_Login
    end

    event User_Logout extends User_Logout
    end

    event Contact_List extends Contact_List
    end
```

```
 event Send_Invite extends Send_Invite
 end

 event Cancel_Meeting extends Cancel_Meeting
 end

 event Voting
  any vv
  where
   @grd1 vv ∈ votes
  then
   @act1 votes ≔ votes ∪ {vv} // Adding vote for a meeting
 end
end
```

8.5 Contexts

```
context GuarantimeContext0

sets PERSON // Set of People who can be the user of Application

end
```

■■

```
context GuarantimeContext01 extends GuarantimeContext0

sets INVITES // It shows the set of Invitations/Meetings
   EMAIL // Set of people who have Email Addresses
   Votes // Votes can be given in 'yes' or 'no'

constants yes no

axioms
 @axm1 partition(Votes, {yes}, {no})
end
```

ISBN XXX-XX-XXXX-X
ISSN 1239-1891