



*Improving the Agile Methods
and Open Source Lab Course*

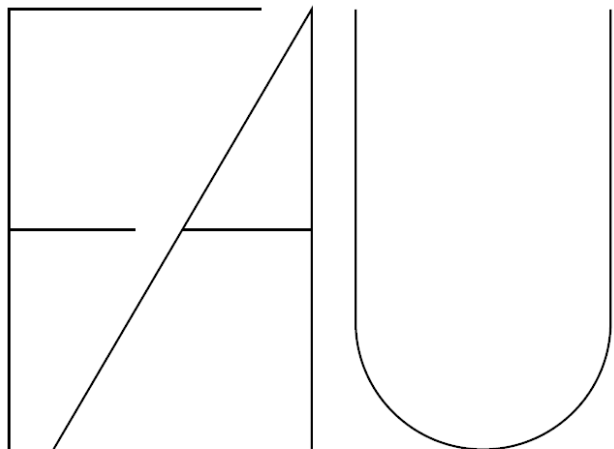
Falko Saft

Master Thesis

Open Source
Research Group

Department of
Computer Science
Faculty of Engineering

Friedrich-Alexander-
University of
Erlangen-Nuremberg



Improving the Agile Methods and Open Source Lab Course

Masterarbeit

vorgelegt von

Falko Saft

geb. 23.05.1986 in Nürnberg, Deutschland

angefertigt am 31.03.2012

**Department Informatik
Open Source Research Group
Friedrich-Alexander-Universität Erlangen-Nürnberg
(Prof. Dr. Dirk Riehle)**

Betreuer: Prof. Dr. Dirk Riehle

Beginn der Arbeit: 25.10.2011
Abgabe der Arbeit: 25.04.2012

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Der Universität Erlangen-Nürnberg, vertreten durch die Forschungsgruppe Open Source, wird für die Zwecke der Forschung und Lehre ein einfaches, kostenloses, zeitlich und örtlich unbeschränktes Nutzungsrecht an den Arbeitsergebnissen der Masterarbeit einschließlich etwaiger Schutzrechte und Urheberrechte eingeräumt.

Nürnberg, den 31.03.2012

Abstract

Although Scrum is relatively new to the academic world, there is a rising awareness to apply agile methods, not only in the professional software industry but also as part of the software engineering curriculum. The agile practices of Scrum were, in the context of this thesis, adopted at the programming course Agile methods and Open Source (AMOS), at the University of Erlangen-Nürnberg. The AMOS course that teaches students modern software engineering was in 2011 designed to develop a first product prototype of Free Seas Ahoy!, the social network for sailors. Computer Science and Information System students were applying the agile practices of Scrum during the semester-long course by taking on the roles of Team Members and Product Owners. This thesis analyzes the AMOS project of 2011 in detail and compares it with the prior and first AMOS course of 2010 by a post-facto analysis. In addition, this research also reviews similar development courses that make use of agile methods and draws a comparison between the AMOS project and the findings of related case studies. Several lessons learned, which result from the analysis, include suggestions how to improve future instances of the AMOS project and comparable academic courses. These proven recommendations embrace changes on the adaptation of agile practices in university courses and the academic course design.

Table of Contents

Abstract.....	I
Table of Contents	II
List of Figures.....	V
List of Tables	VI
List of Abbreviations.....	VII
1 Introduction	8
1.1 Motivation.....	8
1.2 Context and Contributions.....	8
1.3 Structure	9
2 Fundamental principles	10
2.1 Plan driven development.....	10
2.1.1 Waterfall model.....	10
2.1.2 V-model	11
2.2 Agile methodology.....	12
2.2.1 Extreme Programming	12
2.2.2 Scrum	15
2.3 Open Source	21
2.3.1 Open Source Software.....	21
2.3.2 Open Source development	22
3 Scrum in academic courses	23
3.1 Related work	23
3.1.1 Android applications.....	23
3.1.2 Computer games	25
3.1.3 Web service	26
3.2 AMOS project.....	27
3.2.1 Objectives	28
3.2.2 Roles.....	28
3.2.3 Process.....	30

3.3	Academic courses in comparison.....	34
3.3.1	Roles.....	34
3.3.2	Practices	36
3.3.3	Artifacts.....	38
4	AMOS project of 2011	40
4.1	Domain model.....	40
4.1.1	Product Summary	40
4.1.2	Target Audience.....	40
4.1.3	Features.....	41
4.2	Weekly sprints.....	45
4.2.1	Kick-off.....	45
4.2.2	Sprint 1	46
4.2.3	Sprint 2	46
4.2.4	Sprint 3	47
4.2.5	Sprint 4	47
4.2.6	Sprint 5	48
4.2.7	Sprint 6	48
4.2.8	Sprint 7	49
4.2.9	Sprint 8	50
4.2.10	Sprint 9	51
4.2.11	Sprint 10	52
4.3	Development Speed.....	52
4.3.1	Velocity	53
4.3.2	Sprint Forecasting.....	56
4.3.3	Burndown bar.....	58
5	Mydosis and FSAhoy.....	60
5.1	Course setting.....	61
5.1.1	Components.....	61
5.1.2	Domain	61
5.1.3	Participants and Roles	62
5.1.4	Technology	62
5.1.5	Timeframe.....	64

5.2	Project Performance	64
5.2.1	Velocity	64
5.2.2	Lines of Code.....	72
6	Discussion	75
6.1	Academic course.....	76
6.2	Scrum methodology	78
6.3	Velocity	83
7	Conclusions.....	87
7.1	Limitations.....	87
7.2	Recommendations	88
7.3	Future research.....	90
	References	i

List of Figures

Figure 1: Waterfall model according to Winston Royce [10].....	11
Figure 2: V-model according to John O. Clark [12].....	11
Figure 3: Twelve practices of Extreme Programming [19].....	14
Figure 4: Roles in Scrum [11].....	16
Figure 5: Product Backlog, Sprint Backlog and Feature Archive in Scrum.....	18
Figure 6: Release Plan in Scrum [11].....	18
Figure 7: Course schedule of Computer games [29].....	25
Figure 8: Course schedule of Web service [6].....	26
Figure 9: Feature responsibilities in the AMOS project of 2011.....	37
Figure 10: Daily Scrum on the mailing list in the AMOS project of 2011.....	38
Figure 11: Menu bar in FSAhoy.....	41
Figure 12: User profile in FSAhoy.....	42
Figure 13: Trip planning in FSAhoy.....	42
Figure 14: Daily logbook entry in FSAhoy.....	43
Figure 15: OSM integration in FSAhoy.....	44
Figure 16: Planned and completed story points in the AMOS project of 2011.....	55
Figure 17: Burndown bar chart in the AMOS project of 2011.....	58
Figure 18: Programming languages of Mydosis and FSAhoy [38].....	63
Figure 19: Completed story points of Mydosis and FSAhoy.....	65
Figure 20: Completed story points of Mydosis teams and FSAhoy.....	66
Figure 21: Story point curve of Mydosis and FSAhoy.....	69
Figure 22: Story point distribution of Mydosis and FSAhoy.....	70
Figure 23: Completed features of Mydosis and FSAhoy.....	71
Figure 24: Code analysis of Mydosis and FSAhoy [38].....	72
Figure 25: Commits of Mydosis and FSAhoy [38].....	73
Figure 26: LOC of Mydosis and FSAhoy [38].....	74

List of Tables

Table 1: Roles in the AMOS project of 2011	29
Table 2: Structure of the class day in the AMOS project of 2011	32
Table 3: Adaptation of Scrum roles, practices and artifacts in academic courses	35
Table 4: Story points and their meaning in Scrum.....	53
Table 5: Planned and completed story points in the AMOS project of 2011	54
Table 6: One-Sample Statistics for completed story points	57
Table 7: One-Sample Test for completed story points	57
Table 8: Key facts for the AMOS courses of 2010 and 2011.....	61
Table 9: Velocity per developer of Mydosis and FSAhoy	67
Table 10: Statistics for the story point distribution of Mydosis and FSAhoy	70
Table 11: Suggestions for future AMOS projects	75

List of Abbreviations

AMOS	Agile Methods and Open Source
App	application
FSAhoy	Free Seas Ahoy!
IP address	Internet protocol address
LOC	Lines of Code
MPO	Main Product Owner
Mydosis	electronic database for dosage information
OAuth	Open standard for authorization
OSM	Open Sea Map
PO	Product Owner
SM	Scrum Master
SPO	Student Product Owner
TM	Team Member
XP	extreme Programming

1 Introduction

The Agile Methods and Open Source (AMOS) lab course is a semester-long programming course offered at the University of Erlangen-Nuremberg. The basic idea of this course is to teach students the aspects of modern software engineering and to develop a prototype using agile methods [1].

1.1 Motivation

Today, agile methods are not only about to become the state-of-the-art in the software development industry [2], they are also gaining increasing attention in academic courses and are more frequently used in software development projects at universities [3]. Early studies in 2002 showed that lightweight agile processes better fit student projects than heavyweight plan driven approaches. A survey of 49 capstone projects found that agile practices better match the culture and skills of students [4]. In addition, recent case studies document that students are highly positive about courses that apply Scrum in the development process [5], [6].

Although that quite a lot of universities are teaching agile practices, this methodology is often only referred to as another development method next to plan driven models in theoretical lectures on development processes [7]. Courses that primarily address and apply agile practices in a software development project are a relatively new phenomenon [8]. Thus, the adoption and tailoring of agile practices that they match the setting of a university course is still followed by a trial-and-error process that leaves room for improvement.

1.2 Context and Contributions

The annual AMOS project is an academic programming course that uses agile methods and is adjusted to meet the requirements of the academic setting at the University of Erlangen-Nuremberg. Since this project just took place for the second time in 2011, it is still in a learning phase where impediments and challenges are faced. Hence, the idea and concept of the AMOS course is not carved in stone. It is

repeatedly being revised to overcome any arising difficulties and to respond to the feedback that is given by participating students.

The goal of this paper is to suggest and validate improvements that set the stage for future instances of the AMOS project. The obtained experiences from the AMOS project of 2010 and 2011 are in this regard valuable insights that provide lessons learned and give recommendations for a consistent further development of the AMOS concept.

1.3 Structure

This thesis is structured as it follows: Chapter 2 introduces the fundamental principles of plan driven development and explains the basic ideas of the agile methodology and Open Source. These foundations are provided in order to understand the underlying concepts of this thesis. Chapter 3 describes in more detailed the adoption of the agile method Scrum in academic courses. Chapter 4 then covers a thorough analysis of the university course Free Seas Ahoy! (FSAhoy). FSAhoy is the AMOS project of 2011 and the key element of this thesis. The review of FSAhoy is followed by a comparison between FSAhoy and Mydosis in Chapter 5. Mydosis is the previous and first AMOS project of 2010. From the analysis resultant ideas how to improve the AMOS project are framed as hypotheses in Chapter 6 and evaluated by post-facto analyses and case study research. Chapter 7 provides the conclusions of this research by laying out the limitations, summarizing the suggestions how to improve future AMOS projects and outlining the areas which require further research.

2 Fundamental principles

Today, agile methods such as Scrum or Extreme Programming (XP) are becoming more and more state-of-the-art development practice. A survey by Forrester Research found that already 35% of responding organizations were using agile practices as part of their software development process by 2009. Scrum was thereby the most adopted agile method with a utilization rate of 10.9% [2]. In addition, a more recent study reveals that the number of organizations working with Scrum increased from 25% in 2009 to 33% in 2010 when 100 project managers were asked by Danish researchers [9]. This current trend and widespread use of agile methods is raising the questions, “Why is Scrum getting so popular?” and “Why are agile methods getting ahead traditional and plan-driven approaches?”.

In order to understand this observed phenomenon, the main features and differences between plan-driven processes, agile methods and open source are explained in the following.

2.1 Plan driven development

Plan-driven development models are regarded as the traditional way to develop software [6]. They are characterized by linear and phase-oriented process models that try to minimize risk by up-front planning. These linear models are often designed with equal phases that contain specified activities.

2.1.1 Waterfall model

The most noted plan-driven development process is the waterfall model. This model was first described by Winston Royce in 1970 and is illustrated in Figure 1. The idea of the waterfall model is to start from the top and follow the flow successive down to the bottom [10]. That means the same time, one can only move to the next stage in this model if the preceding phase is already completed. The stages that are described by the waterfall model are often defined as: Analysis, Design and Implementation [11].

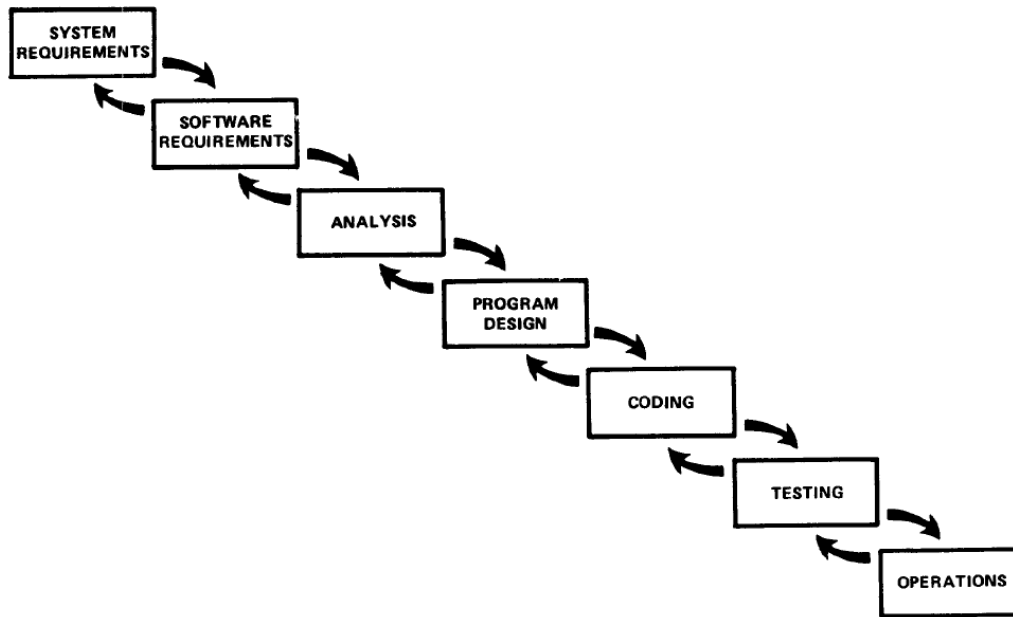


Figure 1: Waterfall model according to Winston Royce [10]

Iterations and feedback loops are, however, restricted and only allowed within consecutive phases.

2.1.2 V-model

A more advanced plan-driven process model is presented in Figure 2 by the V-model.

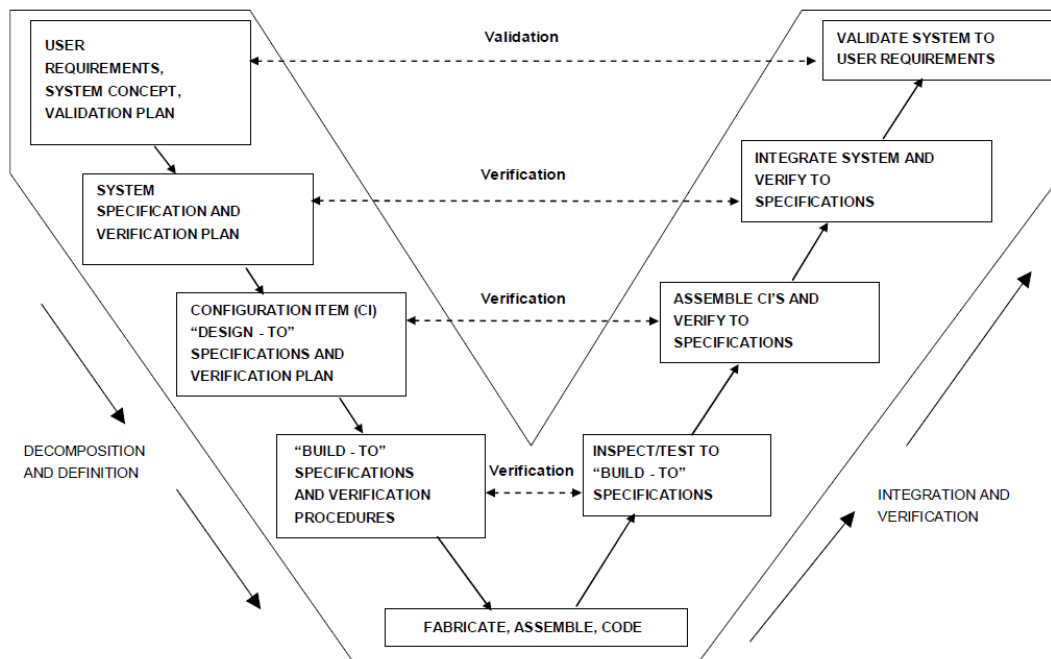


Figure 2: V-model according to John O. Clark [12]

The V-model is basically an enhanced waterfall model that puts an increased emphasis on quality assurance. There are in particular two new dimensions added to the waterfall model: verification and validation. Both terms try to answer the questions, “Are we doing things right?” and “Are we doing right things?”. These questions and the right side of the “V” make it now possible to check and test the software upon desired quality and intended operability.

The latest V-Model is commonly known as the V-Model XT. It describes a sophisticated software development process that became the standard for software projects of the German government. It provides about 20 modules that can be tailored. The V-Model XT is thereby easier to implement as any V-model before [13].

2.2 Agile methodology

Agile methods describe a variety of process models such as Scrum, XP, Crystal, or Feature Driven Development to name a few. These lightweight processes all share the Agile Manifesto that was published by a group of software practitioners [14]:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

All agile methodologies have in common that they rely on short processes and linear development models which are based on iterations. Short iterations and frequent user feedback are important to minimize risk and to adapt quickly to changing requirements [15].

2.2.1 Extreme Programming

Kent Beck, one of the founders of the Agile Manifesto, explains the idea of Extreme Programming. XP is in his words a “style of software development focusing on excellent application of programming techniques, clear communication, and teamwork.” [16].

Values

Beck is addressing some important principles in his definition of the lightweight discipline. The main values are communication, feedback, simplicity and courage. These values are mentioned very explicit in XP to affirm the impact and significance of them.

Communication is all about the interaction with the customer and between developers. The intention of a transparent communication is to remove any barriers that can exist between the customer and the developer. Moreover, the communication takes the given roles in responsibility and allows for adjustments and changes.

Customer feedback is not the only valuable source. Also builds of the software code can provide useful information. Frequent release cycles help the developer team to observe what is working and what not. These feedback loops indicate the progress and pitfalls of a development project.

The development team is also advised to find the simplest solution that is possible. It is important to internalize the feedback that is given by means of communication to build a simple and working project. Making it complex neglects the XP value of a simple solution.

The values of XP also encourage developers to admit if things are going wrong or getting too complex. It is Sometimes a painful decision to fix software code or get rid of features. However, meeting these challenges and communicating them frankly is important in XP [17], [18].

Best Practices

XP provides a set of best practices that can ideally be used and combined. The original twelve practices and their dependencies are highlighted in Figure 3. Today these best practices are extended by further ones. Nevertheless, the focus of this thesis is limited to the three following practices that were also applied in the AMOS project of 2011.

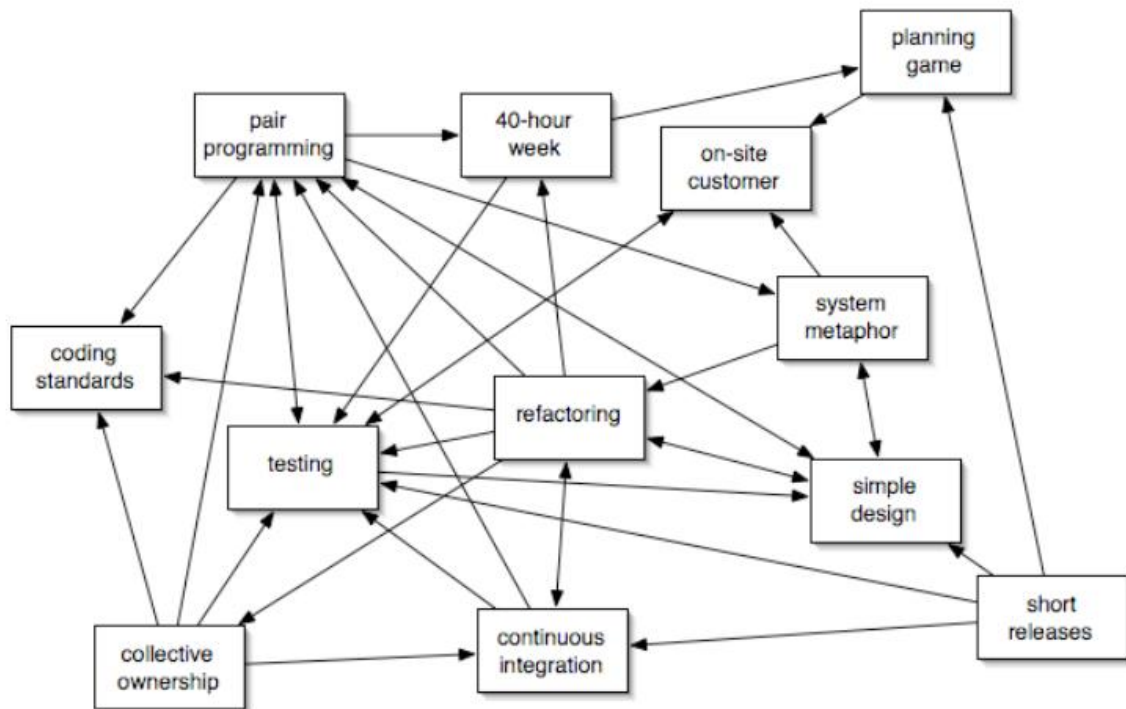


Figure 3: Twelve practices of Extreme Programming [19]

- **Pair programming**

Pair programming or pilot and navigator is probably the most known and applied XP practice. The concept of pair programming is described as two developers sitting in close proximity and working at one workstation. One of them is implementing the code and the other one is reviewing and commenting the written code. This setting encourages the sharing of ideas and contributes to a less buggy and more conceived software code [20].

- **Continuous integration**

A fully integrated system is proposed by the continuous integration practice. Daily builds or builds that are made a few times per day help to maintain working code keep the system running [18]. Thereby, situations where the system is going broke due to implementation problems are reduced. This practice is especially important if you are working with a development method that is based on short iterations and release cycles.

- **Short releases**

The practice of short releases is takes place at the end of each iteration. The next step after passing all acceptance tests, is to release the software. Regular releases and releases within short iterations allow the customer to get an idea of the progress and state of a software product. The customer can use this information to provide feedback and prioritize new features or bug fixes [17].

Organization

Although that there might be a lot of different names for the roles in XP, it consists mainly of two: the customer and the developer. It can be added a management role for lager teams, but XP is in general not paying as such attention to roles as other agile methods do.

XP is principally organized by user stories and the planning game. Both elements require the roles of customer and developer. The customer is writing down the user stories on an index card and prioritizes them according to their value. The stories are rather a description of what the customer wants to see in the final product than an entire requirements document. These user stories are being evaluated then upon their effort by the developer in the planning game in order to estimate the implementation costs. Customer and developer are in the following both discussing the value of user stories for the next release. Some of the stories can be accepted easily if they are small enough to implement and low on risk, others might be too big and complex and have to be broken down for later implementation. The implementation of user stories will subsequently take place in releases which can be further divided in iterations to ensure a better control [21].

All the practices that are mentioned beforehand can be used to develop a software solution that follows the values and principles of XP.

2.2.2 Scrum

The term Scrum originates from a formation of players in the rugby sport. In the context of Computer Science, it describes next to XP one of the most common agile

methodologies. Nowadays, Scrum is also about to become the predominant development practice [9]. The concept of Scrum became popular in the nineties and was first applied by Jeff Sutherland at Easel Corporation and introduced at Advanced Development Methods by Ken Schwaber [22].

As many other agile methods, Scrum describes roles, artifacts and practices that will be explained in the following.

Roles

Scrum basically manages the agile development with three roles as it is shown in Figure 4: Team Member, Product Owner and Scrum Master.

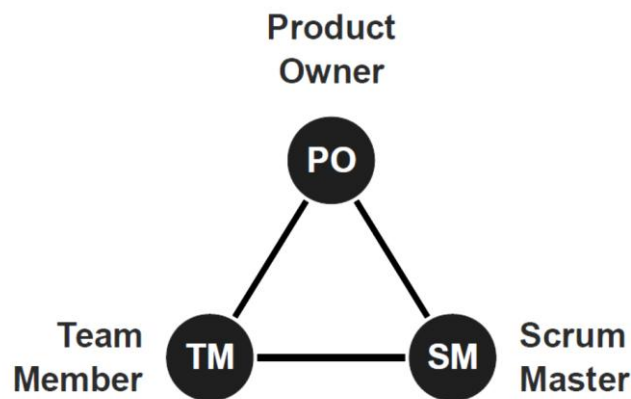


Figure 4: Roles in Scrum [11]

- **Team Member (TM)** are standing for developers in the Scrum model. They are delivering the features that are described in the Sprint Backlog. It is their area of responsibility to code and test the software product upon the agreed requirements. A senior developer or lead architect can complement this team.
- Another role is described by the **Product Owner (PO)** who is in charge of the product backlog and release plan. The functionality, scope and priority of features in the product backlog are defined by the PO. Besides, the PO communicates the vision of the project, interacts with the customer and answers the questions of the development team.

- The **Scrum Master (SM)** is holding the overall responsibility for the Scrum process. The practices and techniques that are used in Scrum are best known by the SM. Any impediments during the Scrum should be removed by the SM who is also responsible to solve conflicts between TMs and PO [22].

To summarize these roles, it can be said that the PO determinates what is to do, the TM decides how to do it, and the SM ensure the framework to do so.

Artifacts

There are several documents that support the development process in Scrum. The Product Backlog, Sprint Backlog, Feature Archive, Release Plan and Burndown Chart are the most important artifacts in Scrum. The Product Backlog, Sprint Backlog and Feature Archive are closely linked. Figure 5 illustrates the relationship between those artifacts.

The **Product Backlog** is the main document that contains the majority of functional and non-functional features at the project start [23]. All these features are available as user stories in the Product Backlog. User Stories should follow the INVEST criteria if they are created. INVEST is the guideline how to write good user stories and stands for independent, negotiable, valuable, estimable, small and testable. This standard ensures that features are worded in a proper way so that they can be understood by the TMs. Since the Product Backlog is not definite, User Stories can always be added, changed or deleted. Most important for the Scrum process is the maintaining and prioritization of User Stories on the Product Backlog. High prioritized features are determined for the next or future sprints.

User Stories which are getting accepted during the Planning Poker are in the following moved to the **Sprint Backlog**. Features that are not agreed on for implementation remain in the Product Backlog. The Sprint Backlog contains the User Stories and the estimated effort in story points to put these features into operation in the concurrent sprint. In the event that implemented features are not accepted by the PO, they will be put back to the Product Backlog and waiting to get prioritized again.

If features are signed-off by the PO, they will be moved to the **Feature Archive**. The Feature Archive is an historical document that tracks already implemented features on a list. Attributes like release date or actual and planned effort can also be added for a comprehensive documentation.

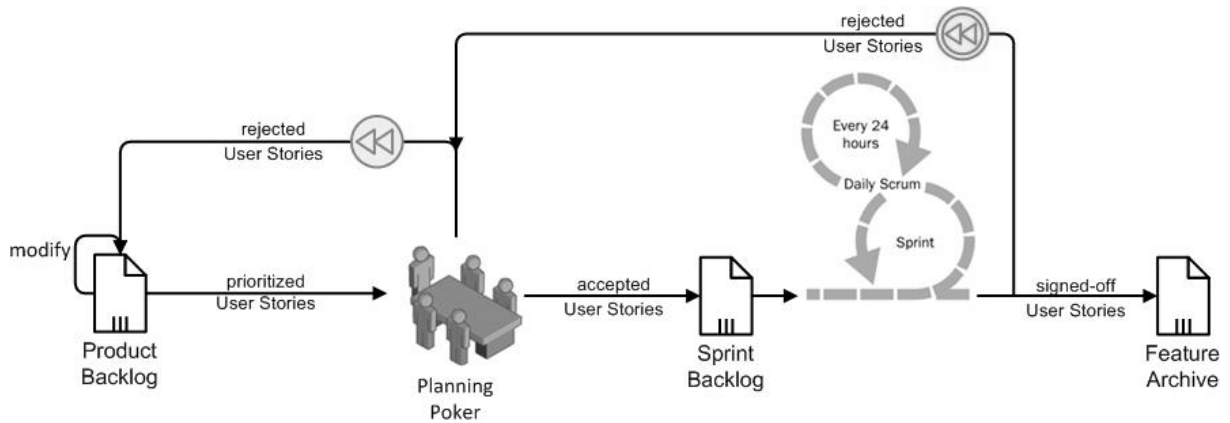


Figure 5: Product Backlog, Sprint Backlog and Feature Archive in Scrum

The Release Plan and Burndown chart are documents that facilitate the release planning and progress tracking.

Figure 6 shows an exemplary **Release Plan**. This plan combines the two dimensions of sprints and features in one document and allows for predicting possible releases.

Sprint #	1	2	3	...	14
Start/End	21.04.10 - 28.04.10	29.04.10 - 04.05.10	05.05.10 - 12.05.10	...	13.07.10 - 20.07.10
Goal	Set-up web server	Bring-up UI, basic forms			
Features	1, 2, 6	9, 12, 14, 18	3, 4, 10, 15, 13, 21		TBD
Est. Effort	13	15	17		TBD
Comment	Form team	Get going	Speed up		Final release

Figure 6: Release Plan in Scrum [11]

The Release Plan includes the starting and end dates of sprints, the description of goals, features and their estimated effort for every sprint. This information can be used to measure the progress of a team and to detect slowdowns. The Release Plan makes, just as the product backlog, changes on the plan possible.

The **Burndown Chart** is, as described later on (see Chapter 4.3.3), as well an effective tool to forecast the release and gauge the progress of a project [22]. The sprint number and the estimated effort in hours or story points of a Scrum project are displayed in the burndown chart. By adding a trend line, it is also possible to illustrate the number of sprints that remain for the final release. The projected trend line also serves as an indicator for the target-performance comparison. It can be made a statement on the achievement of sprint goals by comparing the average effort per sprint with the actual effort per sprint.

Practices

Scrum is, just like XP, an agile method that relies on short iterations. Iterations are called Sprints in Scrum and are supposed to be equal in length and size. The duration of a sprint can, however, vary across projects. A sprint can last one up to four weeks.

Every sprint consists of the three consecutive phases:

- Sprint Planning
- Sprint Execution
- Sprint Review, Release, Retrospective

A more detailed description for each single sprint phase is given in the following.

Development projects that apply Scrum start typically before the Sprint Planning phase. The Product Backlog, the artifact containing a feature-list of the final product, has to be created ahead of the first sprint. This can be done jointly by the PO and TMs. Both parties are working out user stories that are based on the input of the customer and the product vision. Afterwards, these features are getting prioritized by the PO and are made available for actual sprint planning.

- **Sprint Planning**

The PO determines features in the sprint planning that generate a high value and should be implemented within the next sprint. The job of the PO is to also present those prioritized features in front of the development team during the Planning Poker. The developers are then asked to determine the effort for each feature. TMs are doing this by evaluating the features by story points (see Chapter 4.3). The agreed on features are in the following moved to the sprint backlog as long as the developers think they are able to complete them within the next sprint (see Figure 5). The planning poker ends if the developers cannot take on any more features.

- **Sprint Execution**

The attention of the sprint execution is directed to the implementation of features. The TMs are now responsible to implement features according to the User Stories in the Sprint Backlog while the PO is working the product backlog and release plan. The PO is also acting as an intermediary between the customer and the developers in order to support the developers if any questions need clarification. The same time, the PO updates the Scrum documents according to the requirements and feedback of the customer. The SM manages the whole process to keep it self-sustaining.

A **Daily Scrum** meeting can also be part of the Sprint Execution (see Figure 5). That meeting is supposed to take place every day, possibly in the morning and should last no longer than 15 minutes. The following questions should be addressed briefly by each TM [15]:

- What did you do since the last Scrum?
- What are you doing until the next Scrum?
- What is stopping you getting on with your work?

These questions help to get an idea of the actual progress in the Sprint and if there are any impediments that may delay the implementation of features.

However, the daily scrum is not designed to solve problems. Further meetings would be necessary for troubleshooting.

- **Sprint Review, Release, Retrospective**

The end of every Sprint is composed of a review process and optional release as well retrospective.

The functionality of new features that were implemented during the Sprint is presented in the sprint review by the TMs. The features that are passing all the acceptance tests and are implemented as required can be signed-off by the PO and moved to the feature archive. Not accepted features will be taken back to the Product Backlog and get prioritized again (see Figure 5).

A release of the software product subsequent to the sprint review is also possible if features add new functionality to the product and are implemented according to their acceptance criteria. The SM can also make use of a retrospective at the end of every sprint. This session is normally brought into play to review the last sprint along with the development team and propose improvements to the process [23]. This is in particular necessary if acceptance tests fail and features are not being implemented as supposed to.

2.3 Open Source

The term Open Source classifies both, software which is provided under an Open Source license and the development practice.

2.3.1 Open Source Software

Open Source software is in lieu of proprietary software characterized by the free redistribution and modification of source code. That means, the license of Open Source software allows the usage and change of software without paying a license fee. But these usage rights involve also the requirement to distribute modified software under the same terms as the license of the original software [24].

2.3.2 Open Source development

The development model of Open Source is basically embracing the three principles of Open Collaboration [25]:

- Egalitarian
- Meritocratic
- Self-organizing

Egality stands for the way how development takes place in Open Source projects. The programming work is egalitarian so that everyone can access and contribute in a dispersed community. In addition, Open Source development follows a meritocratic process where contributions are merit-based and evaluated upon their quality. Peer review is in this regard a widely used element of Open Source development. Open Source is also always a self-organizing process where collaborators are developing and customizing their own process as they require it. The contributors are working in a dynamic virtual enterprise where they are relying on e.g. electronic communication media, virtual project management and version management to coordinate and manage dispersed projects [26].

The way Open Source works is highly efficient as it is described by Bruce Kogut and Anca Metiu. They found that Open Source avoids the inefficiencies which are apparent in companies that are relying on intellectual property rights. They also explain the success of Open Source by the implementation of the concurrent process to design and test software modules [27].

3 Scrum in academic courses

Despite the success of Scrum in the software industry [2], it is a relatively new development methodology to universities [8]. University courses that do not only teach students the theory and basics of Scrum have to make more of an effort to apply the agile practice in lab courses since Scrum was originally designed for the development in the software industry.

Scrum has a lot of practices, artifacts and roles that are designed to meet the general conditions of the business world (see Chapter 2.2.2). As a matter of fact, universities have to the customize Scrum that it fits the academic setting. The following two subchapters describe first the use of Scrum in different academic courses [6], [28], [29] and the way how Scrum was adapted in the AMOS project of 2011. Last but not least, Chapter 3.3 compares the chosen courses in terms of roles, practices and artifacts.

3.1 Related work

The agile method Scrum is a non-prescriptive practice that allows for changes and is not further restricted to a certain development environment. A few universities are using this approach along with some modifications to develop a project of their choice in the classroom. Three academic courses that are distinguishable in the project type and use of Scrum are selected for an analysis. The development of android applications, computer games and a web service are the paradigmatic university courses that are studied in this thesis. More information on the context of three different courses is given below.

3.1.1 Android applications

The first academic course that is analysis describes the development of Android applications at the Rochester Institute of Technology [28]. The elective course that was designed in order to teach students the agile software development using Scrum

was open to undergraduates. It was split into two parts to separate the preliminary course work from the development process.

Before the actual development started, preparatory work was necessary to set the stage for the forthcoming weeks. The first few weeks of the course were used by students to become familiar with the Java-based development kit – Android Development Tools. An “Android boot camp” was set up to prepare students for the development of android applications. Furthermore, students were asked to write user stories with regard to their application (App) idea. By doing so, all students were involved in the process of collaboratively defining Android Apps.

The actual application development process followed after the preliminary “Iteration Zero” sprint. Three teams with 6-7 students each were developing their own applications in the second part of the course. The following app ideas were created by students [28]:

- A training app that could be used by runners to track their distance (using GPS) and time during training runs
- An app that accessed the university’s bus system to alert the user when the next scheduled bus was approaching a bus stop
- A Black Jack card game which tutored the user in casino betting strategies

The development using the agile method Scrum took place in the following three sprints. Each sprint was thereby three calendar weeks long.

The outcome of this development project is that students profit from the use of Scrum in a few ways. Students learn about requirements engineering, project planning, tracking, testing and effective team collaboration in using Scrum. Nevertheless a more effective learning success can only be achieved if students have the possibility to complete multiple iterations and apply suggested improvements during these Sprints [28].

3.1.2 Computer games

The game design lecture that was offered for Computer Science students at the University of Duisburg-Essen in 2009 was divided into two phases [29]. The first phase was called Design phase or Pre-production and included the theory on game design, structure and production. The second phase was known as the Development phase and contained the practical development of a computer game in ABC-Sprints using the development method Scrum (see Figure 7).

Phase	Activity	Deliverables	Week	Lecture Topics
	Start		1	Organizational Matters
Design	Idea Creation		2	Genres, Innovation and Ideas
	Grouping	Game Design Document, Product Backlog,	3	Production Process
	Conception	Alpha Sprint Backlog	4	Production Tools
	Planning		5	Project Engineering
ABC-Sprints			6	Structure of Games
	Alpha Sprint	Basic Functionality, Proof of Concept, Beta Sprint Backlog	7	Gameplay and Balancing
			8	Interface Design
			9	M1 ALPHA VERSION
			10	Christmas Gaming Session
	Beta Sprint	Feature-complete Game, Final Assets, Completion Sprint Backlog	11	Interactive Storytelling
			12	Character Development
			13	M2 BETA VERSION
	Completion Sprint	Bug-free, Balanced, Polished, and Finalized Game	13	Code Review and Refactoring
			14	Games Business
			15	M3 RELEASE

Figure 7: Course schedule of Computer games [29]

Students had to generate a mind map of their computer game ideas in brainstorming sessions in the first phase. The information gathered in these meetings was used by the students to develop a high concept. These concepts were in the following discussed in groups to create a common understanding and to describe a Game Design Document. Students proactively formed teams of 4 to 5 members for this purpose. The teams were required to create and discuss the Product Backlog with the PO after the group formation process took place.

The second phase started in Week 6 and was split in an Alpha, Beta and Completion Sprint (ABC-Sprints). The Alpha sprint lasted four weeks and was planned to deliver a first prototype with basic functionality. Further features were intended to be implemented during the four-week long Beta sprint. The final release of a feature-complete game was the goal of the Completion Sprint that was designed for debugging and polishing in last two weeks of the semester. All the Sprint work was

accompanied by weekly lectures on game design (see Figure 7). Apart from that the development teams were working as self-organized Scrum teams during the second phase of the game course, solely the monitoring of the teams was assigned to the instructor of the course.

The implication from this case study is that using Scrum in game development leads to an improved productivity and more sophisticated games. However, students were claiming a very high course workload in a survey that was carried by the end of the course even though that they admitted the benefits of Scrum [29].

3.1.3 Web service

Another course that is using the agile method Scrum was held at the University of Ljubljana [6]. This undergraduate course was taking place the first time in the academic year 2008-2009. A year later, the second instance of the course was running. The goal of this development course was to develop a web service in the review period of 2009-2010. Like the two courses aforementioned, this course was also composed of two parts (see Figure 8).

Sprint 0	Weeks 1-3	Formal lectures on Scrum and user stories. Preparation of development environment. Initial Product Backlog presentation, user stories estimation, release plan development.
	Week 4	Sprint planning meeting, initial Sprint Backlog development.
Sprint 1	Weeks 4-7	Project work, Daily Scrum meetings, Sprint Backlog maintenance.
	Week 7	Sprint review meeting, Sprint retrospective meeting.
Sprint 2	Week 8	Sprint planning meeting, initial Sprint Backlog development.
	Weeks 8-11	Project work, Daily Scrum meetings, Sprint Backlog maintenance.
	Week 11	Sprint review meeting, Sprint retrospective meeting.
Sprint 3	Week 12	Sprint planning meeting, initial Sprint Backlog development.
	Weeks 12-15	Project work, Daily Scrum meetings, Sprint Backlog maintenance.
	Week 15	Sprint review meeting, Sprint retrospective meeting.
Release 1		Delivery of requested functionality, presentation of observational study results.

Figure 8: Course schedule of Web service [6]

The course started with introductory lessons on the theory of Scrum in the first three weeks of the academic term. The so called Sprint “0” involved students to become acquainted with the idea and use of user stories as well as the Product Backlog. At the same time the development environment was prepared. In addition and prior to the following sprints, 13 teams with four students each were formed.

The remaining three sprints were each four weeks long and used for the development of a Web-based student record system. The teams were asked to develop the web service by the user requirements that were the same for all the teams and already predefined by the instructor. 60 prioritized user stories were therefore introduced to student teams. The user stories were prioritized according to the categories of “must have”, “should have”, “could have” and “won’t have this time”. Although that the majority of teams implemented all 24 “must have” and 5 “should have” features, only the best participating team completed also all 4 “could have” and some of the 27 “won’t have this time” user stories.

The bottom line of this academic course is that students enjoy learning Scrum and show and increased performance from Sprint to Sprint by applying this agile method. But going in a more detailed analysis reveals that students often not entirely understand the concept of Scrum and misinterpret the definition of when a user story is “done” [6].

3.2 AMOS project

The Agile Methods and Open Source (AMOS) lab course is a semester-long programming course at the University of Erlangen-Nuremberg. This course is offered by Prof. Dr. Prof Riehle at the Open Source Research Group and takes place every summer semester.

The concept of the AMOS course was originally described in 2010 when the course took place for its first time. It was updated to match the setting of the AMOS course in 2011 after first practical experience was gained and feedback was given from participating students in 2010 [30], [5]. Since the circumstances of the university lab

course are changing every year also the AMOS concept is likely to be refined in future instances in order to respond to the lessons learned and meet the conditions of upcoming projects.

The project concept of the AMOS course and some of the adaptations for the AMOS project of 2011 are described in the next paragraphs. The AMOS concept is subdivided in the objectives (Chapter 3.2.1), the roles (Chapter 3.2.2) and the process (Chapter 3.2.3) of the university project.

3.2.1 Objectives

The general idea of the AMOS course is to combine the teaching of agile methods and Open Source with the development of a software solution. The objectives that can be derived from the idea of the AMOS project are drawn in the following [1]:

- Effective learning outcome in a real-world environment
- Sustainable project for a good cause
- Start-up foundation for interested students

The learning process of the AMOS project is taking place in a real-world environment that comprises a real customer and real product. This authenticity facilitates the knowledge transfer and the understanding for students. A sustainable product is furthermore an incentive for students to attend this course. Developed software that is still available and used after the AMOS course is finished, can as well contribute to the foundation of a start-up. Especially students that are about to graduate are offered a great chance to create a start-up company.

3.2.2 Roles

Computer Science students as well as Information Systems students are the target group of this course. University courses that make use of agile methods are often offered exclusively as capstone courses, either for undergraduate or graduate students [4], [7], [8]. This distinction was not made for the AMOS project of 2010 and 2011. Both, bachelor and master students are allowed to enroll for this course.

An overview of the specific roles for the AMOS project of 2011 is given in Table 1.

Table 1: Roles in the AMOS project of 2011

SCRUM Roles	AMOS Roles	Recruiting
Scrum Master	Lecturer	Professor
Product Owner	Main Product Owner	Senior Student
	Student Product Owner	Students
Team Member	Senior Architect	Senior Student
	Team Member	Students
Customer	Domain Expert	Extern
	Operation Manger	PhD students
	Controller	Students
	Web Designer	Students
	Tester	Students
	Business Planner	Students

The traditional Scrum roles that were described beforehand (see Chapter 2.2.2) are adapted to meet the requirements of AMOS project concept. The SM is represented by Prof. Dr. Dirk Riehle who is giving the annual AMOS lecture. The role of the PO was divided in Main Product Owner (MPO) and Student Product Owner (SPO). This was new to the AMOS product and necessary due to the vast amount of students with a background in business that were enrolled in the AMOS project of 2011. These students were now assigned the role of SPOs that were guided by MPOs. In 2011, there were two senior students taking on the role of MPOs and 14 SPOs. The author of this thesis was one of the MPOs. The TMs were composed of one senior student who was acting as the lead architect of the project and 5 Computer Science students. Domain expertise was provided from the philosophical leader of OSM.

Not all of the enrolled students could be assumed the traditional Scrum roles. There were for instance some PhD students available that ensured the operability of the project server and release system. Next to it, new roles for students were established in the AMOS project of 2011. This step involved all the students that participated in the course and were not assigned the role a SPO or able to work as a TM. The special roles that were created are displayed with a green background in Table 1.

Students were now taking on the roles of controller, web designer or tester. Moreover one team consisting of five members was responsibility to work on the business plan.

3.2.3 Process

The AMOS project consists of the four phases that are explained hereinafter [11]:

- idea creation
- preliminary work
- lab course
- start-up foundation

Idea creation

Preliminary to creating the vision of the project and the project plan, is to find external partners or sponsors and students that are willing to support and be part of the AMOS project.

If there are ideas provided by external sources, there should always be a contact person available that is at the same time an expert in that domain. Philosophical and technical leaders are in this regard very valuable to the AMOS project as they can give their input to the concept and development of the project.

The ideas that are proposed should also be of commercial interest and call for domain expertise that can be communicated easily to the students. Without these attributes, it is difficult to go for a project that is interesting for students and likely to be commercially successful. Social software solutions are in this matter neglected since they are most likely already developed by someone else or lacking domain knowledge.

One of the ideas for the AMOS project of 2011 was suggested by the stakeholders of Open Sea Map (OSM). To be precise, the philosophical leader and technical leader of OSM were proposing their ideas and offering collaboration. The contact persons of OSM were represented by the domain expert Markus Bärlocher and the technical expert Olaf Hannemann.

The idea to create a social network for sailors did not entirely meet the requirements of the AMOS concept since the final product was a category of social software. However, this social network was requiring domain expertise in development and focusing on a niche market is of commercial interest. Further information on the domain model of FSAhoy is explained in Chapter 4.1.

Preliminary work

The output of the preliminary work should be a concrete product idea that is summarized in the vision and plan of the product. To accomplish this goal, it is first of all necessary to generate ideas for a later use in the AMOS project and assess them based on their feasibility. The feasibility study can then also help to create a product strategy or first business plan. All the information that is collected and generated can be part of the product vision. It is also recommended in the concept of the AMOS project to develop a product plan next to the product vision. This plan is a valuable document as it contains also the requirements of the final product. Since the AMOS project is making use of the development method Scrum, the product plan can be realized easily by the Product Backlog.

Apart from the contextual work, it is also essential to solve all technical issues to ensure a smooth project start. The development environment and tools have to be set up by the PhD students and a senior architect should be determined to prepare the technical ground for the AMOS project.

Possible ideas for the AMOS project of 2011 were gathered in conference calls with potential customers. The concepts that were developed by the input of the telephone conferences were documented in the project wiki next to the meeting minutes. These documents presented the basis for a market research and feasibility analysis. The obtained studies revealed that the idea of a social network for sailors would be the most promising concept. This concept was then framed as a product strategy and summarized in the product vision. Beyond that, the POs were working on the product plan that was implemented in the Product Backlog whereas the senior architect was setting up and configuring the development environment.

Lab course

The lab course is in general designed for the time period of 13-14 weeks that are available during a summer semester. However, the first class is used as a kick-off and the last class as a wrap-up. In the end, there are only about 11-12 weeks remaining for the actual development of the first prototype. The number of iterations that were available for the development of FSAhoy was even reduced to 10 sprints in 2011 due to the arranged absence of the SM in one instance.

The AMOS course itself starts in the first week of the semester and contains a theoretical and practical part. The weekly lecture on the theory of agile methods and Open Source are held in the morning. The hours after the lunch break are reserved for the Scrum process that includes the Sprint Review, Sprint Planning and next Sprint Preparation (see Table 2).

Table 2: Structure of the class day in the AMOS project of 2011

Type	Start	End
Lecture	10:15 a.m.	11:45 a.m.
Lunch break		
Sprint Review	12:15 p.m.	01:00 p.m.
Sprint Planning	01:00 p.m.	02:00 p.m.
Sprint Preparation	02:00 p.m.	02:45 p.m.

These time slots were not set for all practical purposes. The Sprint Review could be also prolonged. This decision was always depending on the team performance during the sprint, the release decision after the Sprint Review and the necessity of a retrospective.

The development and software project management is taking place during the week of a sprint. The POs are hereby working on the Product Backlog and the TMs are responsibly to implement and test the accepted features. Once in a week is the AMOS class day, where the completed user-stories are presented by the review manager during the Sprint Review. The Sprint Review can contain the release of the software product if new features are implemented and no greater challenges are

faced. Otherwise, a retrospective can be held to clarify occurring issues. Subsequent to the Sprint Review is the Sprint Preparation for the next week. The MPOs are using this session to brief the SPOs how to write good user stories and what user stories they are responsible for. SPOs are the same time also instructed how to lead the upcoming Sprint Planning and Sprint Review.

Start-up foundation

The AMOS lab course is embedded in a start-up incubation process and thereby encouraging students to create a start-up after the end of the semester-long course. There are normally quite a few stakeholders involved in the AMOS project that might also be interested in the idea of a start-up foundation.

The possible members of a start-up team could be:

- *Students* in the role of Product Owner or Team Member
- *Customer or Sponsor* of the project
- *Professor* as the Scrum Master

If the product is running and marketable by the end of the AMOS course, it is very likely that students will be motivated to create or join a start-up. The MPO and the senior architect that are involved in the AMOS project from the outset are predestined to continue working on the project. Students in the role of SPO or TM are as well qualified to work in the start-up as they have learned a lot about the product during the sprints. A sponsor or customer can also be part of a start-up team as they are providing the domain expertise throughout the development project. The professor of the AMOS course cannot be a founding member in a narrow sense, but he has the ability and network to support the start-up.

A possible start-up team can refer to the first product prototype that was developed during the semester and fall back on the SCRUM artifacts that are available. In 2011, there was even a group of students working on the business plan of the AMOS project. This document can serve as the basis for the start-up to apply for a sponsorship or venture capital.

3.3 Academic courses in comparison


Chapter 2.2.2 describes the roles, practices and artifacts in Scrum. These properties of the agile development method are serving now the foundations for the comparison between the three academic projects aforementioned and the AMOS project of 2011.

3.3.1 Roles

Table 3 clearly indicates that all evaluated Scrum teams are composed of a minimum of 4 up to a maximum of 7 students. Although the total number of participating students is different, there is not much variation in the number of students per team. The average number of approximately 5 students per team describes probably best that working in too small or too large groups is less productive. Small teams lack the manpower and knowledge to work effectively and large teams face issues in matters of organization and communication that are related to the size of a team [31].

The POs were represented by students in two out of four agile projects while the other half assigned the course instructor for the role of the PO. These significant differences in roles are mostly a result of the unlike project philosophies. The AMOS project of 2011 was for instance designed to offer the role of PO to Information System students and to provide the role of TM for Computer Science students. Since there were a lot of students participating in the AMOS course 2011 that could not be assumed the role of a developer, there were way more POs than TMs available. The development course of Android applications was also composed of students as POs that were the same time TMs. Opposed to these approaches, the development of a Web service and Computer games were relying on the instructor as a PO. The instructor had in these courses greater possibilities to provide directions for development. In the role of PO, the instructor could reduce the workload and complexity in the game development course and was able to answer questions regarding the Product Backlog in the Web service project.

Table 3: Adaptation of Scrum roles, practices and artifacts in academic courses

SCRUM		AMOS 2011	Android applications	Computer games	Web service
Roles	Team Member	1 Team: 5 Students, 1 Senior Student	3 Teams: 6-7 Students each	3 Teams: 4-5 Students each	13 Teams: 4 Students each
	Product Owner	14 Students, 2 Senior Students	1 Student per team	Instructor	Instructor
	Scrum Master	Instructor	Instructor, Students	1 Student per team	Instructor
Practices	Sprint Planning	Planning poker	Planning Poker (tasks in hours)	Planning Poker (manageable tasks)	Planning Poker (tasks in hours)
	Sprint Execution	Kick-off meeting 10 sprints (1 week each) Daily Scrum (mailing list)	Iteration Zero sprint 3 development sprints (3 weeks each) Stand-up meeting (once per sprint)	Design and ABC-Sprints (2-4 weeks each) Daily Scrum (twice a week) Scrum of Scrums (once a week)	Sprint 0 (3 weeks) Sprint 1,2,3 (4 weeks each) Daily Scrum (twice a week)
	Sprint Review	Review, Release, Retrospective	Review, Release, Retrospective	Presentation, Release	Review, Release, Retrospective
Artifacts	Documents	Product Backlog, Sprint Backlog, Feature Archive	Product Backlog, Sprint Backlog, Burndown Chart	Product Backlog, Sprint Backlog, Burndown Chart	Product Backlog, Sprint Backlog, Release Plan
	Tool support	Google Docs	Board		Agilo for Scrum

The SM originally describes the role of someone who has experience in applying Scrum and a broad knowledge on that methodology (see Chapter 2.2.2). It is all the more astonishing that this role was the job of a student in the Computer games course whereas all remaining courses were at least starting with the instructor as the SM. However, the disposition of the SM in the development of Computer games was different. This role was seen “as interface between the Scrum Team and the Product Owner” and therefore responsible to mediate between the TM and PO [29]. The android development course started with the lecturer as SM. This role was later on transferred to students as they gained more experience in working with the agile method Scrum.

3.3.2 Practices

The sprint planning is very similar in the surveyed projects. Each development team was playing planning poker in order to estimate the effort of user stories that were accepted for implementation in the upcoming sprints. All but one development project were breaking user stories down into more specific and smaller development tasks. Those development tasks were estimated in hours in two projects (Android applications and Web service) defined as manageable tasks in one project (Computer games). On the contrary, the AMOS project of 2011 did not decompose the user stories further in development tasks. There was just the approach, late in the project, to assign features to different developers (see Figure 9). Only one user story was decomposed in the user story itself and related acceptance as well as JUnit tests. But no estimates were given for the expected development time in hours.

All development projects have in common that the first weeks of the semester were used to get acquainted with Scrum. Next to it, two projects were creating the concept and user stories of their projects (Android applications and Computer games) in the first sprint. Others were setting up the development environment (Web service) or getting familiar with the development tools (Android applications). The AMOS project was an exception in this regard as the product vision and majority of user stories was already in place at the project start. Also the development environment was for the most part set up and the developers experienced in the programming language of the

project. By having this preliminary completed by the semester start, the attention of the first week was directed on the team formation and getting to know each other. Especially the large group of participants made it necessary to start the AMOS project with team building exercises (see Chapter 4.2.1).

Amos Sprint: function responsibility : AMOS Sprints					
Feature Name	Andreas	Christoph	Lars	Simon	Tino
Create Daily Logbook Entry					
acceptance tests			X		
JUnit/Validation/Message Handling		X		X	
Login via OSM	X	X			
Authorize access on OSM	X	X			
Security with CAPTCHAs			X		X
User Profile				X	X
Update Profile				X	X
View & Update Trip				X	
Change Password			X		
Update Info different colour	X				
Disclaimer Page					X
Privacy and Policy Page					X
Welcome Introduction Text					X
Coordinates OSM in Logbook		X			
Bug Fixing					X
Rücksprache Falko					X

Figure 9: Feature responsibilities in the AMOS project of 2011

The duration of sprints was different for the academic courses. The sprints of the AMOS project lasted only one week whereas the sprints for the development of the Web service were scheduled for 4 weeks. In the end, the total amount of weeks was roughly the same, allocating for 10 sprints in the AMOS project and one preparation plus three development sprints for the remaining projects.

The Daily Scrum practice was adapted in various ways to comply with the limitations of academic courses. Since students were not working for the development projects on a daily base, adjustments had to be made to the original concept of the Daily Scrum. A meeting twice a week was feasible for students in the computer games and web service course. The development course for android applications established stand-up meetings once a week. Again the AMOS project was an exception. This project was also asking three questions according to the Daily Scrum rules (see Chapter 2.2.2) but used a mailing list for the weekly consultation. Figure 10 shows the Daily Scrum practice as it was introduced on the mailing list of FSAhoy early in the project by a senior developer.

[Freeseasahoy-devel] Status?

From: Frank Denninger <frank@de...> - 2011-06-05 14:55

Hallo,

wie siehts bei euch aus?

Ich halte es für das beste, jeder von euch schreibt eine kleine EMail nach den "DailyScrum"-Regeln.

Also Beantwortung der drei Fragen:

"Was habe ich seit letzten Mittwoch gemacht"

"Was hab ich bis nächsten Mittwoch vor"

"Was hält mich von meiner Arbeit ab"

mfg

Frank

PS: "nichts" ist natürlich auch eine Antwort

Figure 10: Daily Scrum on the mailing list in the AMOS project of 2011

In addition to the Daily Scrum, there was also established a Scrum of Scrums that covered the exchange of communication and technical issues between the SMs across the teams in the computer games development project. The SMs were expected to provide the solutions that were discussed in the Scrums of Scrum for every development team.

Review, Release and Retrospective were as part of the Sprint Review implemented equally in development projects. The Sprint Review in the Computer games course involved a presentation of the students at the end of each sprint. This presentation by the development teams was used to showcase the development progress, outline problems and discuss improvements as it is common for the review and retrospective. Releases were also often determined by the preset course structure as it is shown by Figure 7 and Figure 8 (Computer Games and Web service).

3.3.3 Artifacts

Artifacts, such as the Product Backlog and Sprint Backlog are a widely used practice in the development projects. All evaluated projects contained as their main documents in the Scrum process a Product Backlog with user stories and a Sprint Backlog for the sprint execution. The Feature Archive that was not mentioned in the

surveyed case studies was essential for the AMOS project to measure the project progress and velocity (see Chapter 4.3.1) since there was no burndown chart or release plan available. The Release plan was in fact only an integral component for the Web service project. One of the early activities of the TMs in the development project of a Web Service was to create the Release Plan. The Burndown chart (see Chapter 2.2.2) is an artifact that was used of in half of the projects (Android and Game development). It was widely used to compare and monitor different teams in the Game development course. The Android applications course used the burn down chart extensively to track the scheduled and remaining sprints hours against the background of a limited budget of hours [28].

The tool support for the Scrum artifacts is above all relatively low. The course for android applications maintained its Scrum documents manually on a Board which highlighted the user stories and burndown chart. The AMOS project was setting up the Product Backlog, Sprint Backlog and Feature Archive on Google Docs to facilitate the communication and access via shared spreadsheets. A professional project management tool was only applied in the course designed for the development of a Web service. The commercial software Agilo for Scrum was used to collect and compare data about the planned and achieved story points [32].

4 AMOS project of 2011

After looking at the basics of agile methods and considering Scrum in an academic setting, it is getting time to shed more light on the AMOS project of 2011. The general set-up, the roles, practices and artifacts are explained in Chapter 3.2. Missing is the description of the domain model and the actual development process. Chapter 4.1 will outline the idea and domain model of FSAhoy before a review of the single sprints and development process is given in Chapter 4.2.

4.1 Domain model

In order to provide an understanding of the AMOS domain in 2011, the big picture of FSAhoy is described first by the product summary. This product summary which is based on the product vision will outline the general idea of the social network for sailors. The target group and features of the social network are further details of the AMOS domain model that are presented afterwards.

4.1.1 Product Summary

The AMOS project of 2011 describes the new generation of an open social networking portal for sailors and anyone interested in sharing nautical information and experience. Interested parties can sign up directly, via OSM or with Facebook Connect. Users of the social network can share information about their past, present, and future sailing trips with their friends. This includes among other things: trip dates, nautical information, photos and travel reports. A key component of FSAhoy is the use of OSM, the free sea map of the world. Using OSM data, sailors can plan, describe and annotate their trips.

4.1.2 Target Audience

Initially, the social network is designed for a variety of non-commercial sailors who want to connect to like-minded sailors for the purpose of learning more about the sailing sport and present themselves with their sailing hobby. Moreover, sailors can make use of this portal to stay in touch with their family while they are on sailing trips.

Amateur as well as experienced sailors are invited to share their knowledge about sailing, to look for crews and plan boat trips. Boaters striving for competitive sail events are addressed in the same way as pleasure boaters who are organizing private cruises. In addition, event managers are encouraged to participate in the social network to create and spread their nautical events. Stakeholder with a commercial interest such as sailing schools, port operators or travel agencies will be attracted by the community of the social network.

4.1.3 Features

Users of the social network can access all the features FSAhoy is offering as soon as they are logged in successfully. The menu bar of FSAhoy in Figure 11 displays the main features of the Web Service that are visible after the login. The user can navigate through the menu bar to discover the functionality of the social network, the trip planning feature and the integration of OSM.



Figure 11: Menu bar in FSAhoy

Social Network

For the reason that FSAhoy is a network which is connecting sailors, it requires a lot of social features that match the demands and expectations of sailors. Basic profile information is in this manner enriched by sailing skills and interests (see Figure 12).

Sailors can not only portray their personality in FSAhoy, they can also tell about their sailing profession as well as their favorite travel destinations. Providing this data enables users of the social network to search and find like-minded people. The social component is also extended by the profile picture. Users can upload or change their profile picture. By attaching a photo to their profile they can present themselves visually and make their profile more attractive for other users of the social network.

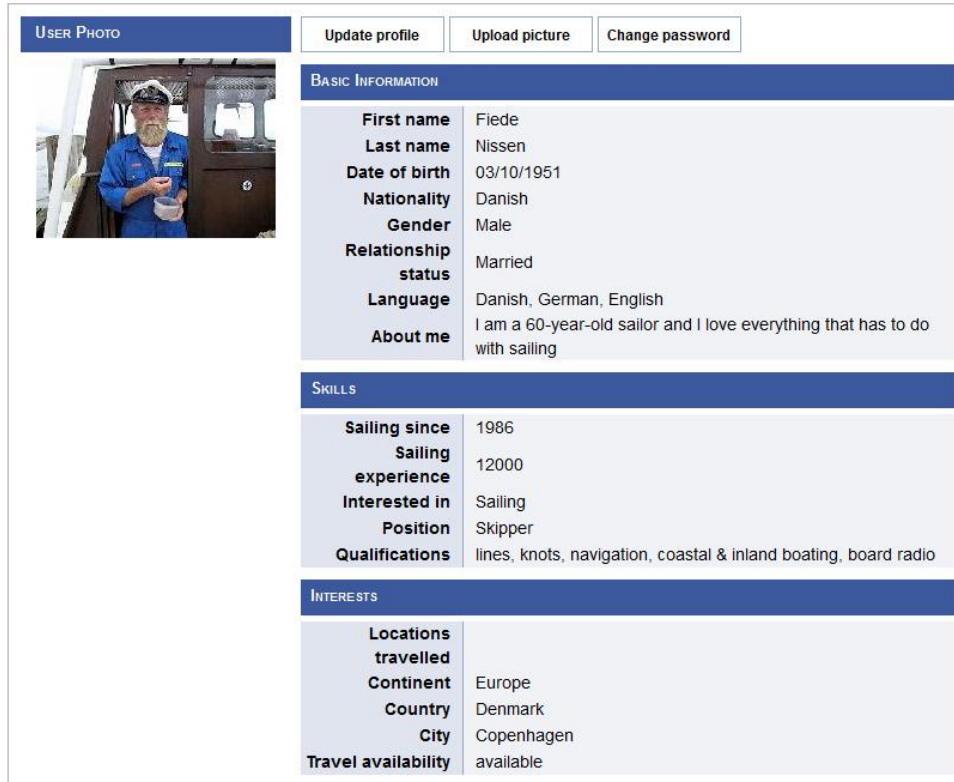


Figure 12: User profile in FSAhoy

Trip planning

Sailing trips can be planned directly in FSAhoy. The Trip section allows users of the social network to create new trips by naming trips, providing a start and end date as well as an optional description. Every trip can have multiple logbook entries that can be attached to the trip. Figure 13 shows an overview of the trip “Nordcup” that is consisting of three logbook entries as it can be found in FSAhoy. All information can be viewed, updated or deleted. In addition, it is on the user’s hands to decide on the level of visibility. The trip can be made available to the public or remain private.

Create trip		My trips	
Trip name	Timeframe	Visibility	Operations
▼ Nordcup	11/11/2011 - 12/18/2011	PRIVATE	Map / Add logbook entry / View & Update / Delete
1 Ralswiek - Cuxhaven	11/11/2011 - 11/15/2011	PRIVATE	View & Update / Delete
2 Cuxhaven - Oostende	11/16/2011 - 11/22/2011	PRIVATE	View & Update / Delete
3 Oostende - Brest	11/25/2011 - 12/18/2011	PRIVATE	View & Update / Delete

Figure 13: Trip planning in FSAhoy

A logbook entry as it can be added to any trip is displayed in Figure 14. The logbook, which is a mandatory document for sailors, addresses the course of a sailing trip in depth. Departure as well as arrival places and trip dates are next to the coordinates of a trip the most important and required logbook entries. The electronic logbook of FSAhoy comes also with nautical information such as water level, wind direction and speed, sea state, weather and air temperature. It embraces beyond these data also the possibility to write down notes or add crew members to the logbook.

<p>Departure (place) *</p> <input style="width: 90%;" type="text" value="Ralswiek"/>	<p>Date / Timezone *</p> <input style="width: 80%;" type="text" value="11/11/2011 12:"/> <input style="width: 10%; border: none;" type="button" value="(GMT +1:00)"/>	<p>Water level (?)</p> <input style="width: 90%;" type="text" value="15 meter"/>
<p>Stopover (place)</p> <input style="width: 90%;" type="text"/>	<p>Date / Timezone</p> <input style="width: 80%;" type="text"/> <input style="width: 10%; border: none;" type="button" value="(GMT +1:00)"/>	<p>Coordinates (?) *</p> <p>Longitude: <input style="width: 40px;" type="text" value="13"/> ° <input style="width: 80px;" type="text" value="8920045"/> ' <input style="width: 30px;" type="text" value="E"/></p> <p>Latitude: <input style="width: 40px;" type="text" value="54"/> ° <input style="width: 80px;" type="text" value="31.0346"/> ' <input style="width: 30px;" type="text" value="N"/></p>
<p>Arrival (place) *</p> <input style="width: 90%;" type="text" value="Cuxhaven"/>	<p>Date / Timezone *</p> <input style="width: 80%;" type="text" value="11/15/2011 12:"/> <input style="width: 10%; border: none;" type="button" value="(GMT +1:00)"/>	<p>Wind direction (?)</p> <input style="width: 90%;" type="text" value="North"/>
<p>Notes</p> <div style="border: 1px solid #ccc; padding: 5px; min-height: 60px;"> <p>The first stop of our Trip is in Cuxhaven. After a rough weather the last few days, we are resting for a night in Cuxhaven. Tomorrow we will continue our sailing trip towards Oostende</p> </div>	<p>Crew/Guests</p> <div style="border: 1px solid #ccc; padding: 5px; min-height: 60px;"> <p>Fiede Nissen Michaela Burg</p> </div>	<p>Wind speed (?)</p> <input style="width: 40px;" type="text" value="76"/> <input style="width: 100px;" type="text" value="Kilometers per hour (km/h)"/>
<input type="button" value="Create logbook entry"/>		<p>Sea state (?)</p> <input style="width: 90%;" type="text" value="rough"/>
		<p>Weather (?)</p> <input style="width: 90%;" type="text" value="stormy"/>
		<p>Air temp (?)</p> <input style="width: 40px;" type="text" value="5"/> ° <input style="width: 100px;" type="text" value="Celsius"/>
		<p>Motoring time (?)</p> <input style="width: 90%;" type="text" value="19:37"/>
		<p>Fuel status (?)</p> <input style="width: 90%;" type="text" value="250 liter"/>

(*) required fields

Figure 14: Daily logbook entry in FSAhoy

OSM

The integrated map of OSM is a major feature of FSAhoy. It is accessible after a new trip is created. The link to the map is available in the trip overview (see Figure 13) and allows the user of the social network to plan a sailing route on the open layer of OSM. The user can to be more precise, interactively mark waypoints on the map of OSM (see Figure 15). These recorded coordinates, which are listed in a pop-up window, can be transferred to a new logbook entry as it is shown in Figure 14.

Besides, the new window also presents the total trip distance and the difference between waypoints in nautical miles.

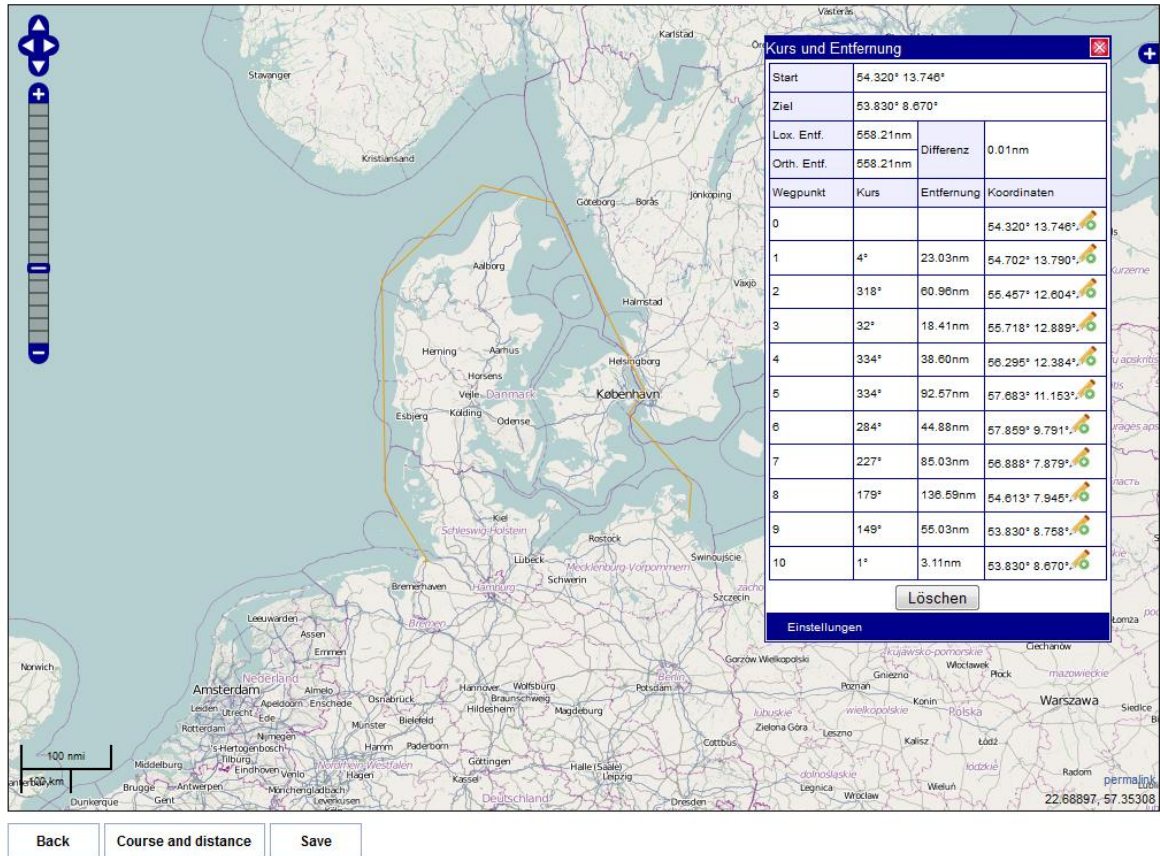


Figure 15: OSM integration in FSAhoy

The traditional web service was also planned to be accompanied by an android component that includes the same functionality as the network portal and additional features. The possibility to use the compass and GPS of a mobile device allows for embedding a mobile navigation module and uploading of real-time geographic information. The geo-data can be used to track the sailing trip, change the sailing route or to give feedback to OSM. Local businesses and points of interest at the harbor site can be located easily by the GPS module and OSM data. In addition, information about the fresh water, grey water, fuel, and battery level of the vessel is retrieved by the mobile application.

However, the android application was not realized in the AMOS project of 2011.

4.2 Weekly sprints

To obtain a better understanding of the applied agile methods in the AMOS project of 2011, a more detailed description of the weekly sprints is given in the following. Not only the idea and domain model of FSAhoy will be evaluated, but also insights on the circumstances and noteworthy events will be provided. The presented information is gathered from the meeting minutes of the weekly Scrum sprints, discussions throughout the AMOS course, Emails with stakeholders, Skype-logs and the developer's mailing list.

4.2.1 Kick-off

The kick-off of the AMOS project of 2011 was taking place on May 4, 2011. That makes a week before the first sprint was scheduled. Prof. Dr. Dirk Riehle made use of the first AMOS session to outline the course organization and to give an understanding of the development process. The domain model of FSAhoy was introduced by Markus Bärlocher, the customer of the project. After briefing all attending students, two team building exercises took place. A ball game was played and a pipeline was constructed jointly by the participants of the AMOS project in order to become acquainted with the names and hobbies of the fellow students. Both exercises were in manner of speaking icebreakers that facilitated the process of getting to know each other and set the same time the stage for a successful collaboration in the upcoming weeks. Afterwards, the SM, PO and customer were meeting up in order to define the domain model more precisely and to talk about further actions.

Since more students than initially expected were enrolling and attending the first lecture, there were a lot of organizational matters to be discussed within the first days of the project. Students with a background mainly in Business studies were either assigned the role of a SPO, part of the business plan team or responsible for a special job (see Chapter 3.2.2). Computer science students or Business students with experience in programming were assumed the developer's role. The development team was complemented by the senior developer and architect Frank Denninger, who was also taking on an active part in the AMOS project of 2010.

4.2.2 Sprint 1

The TMs played planning poker for the first time in the AMOS project of 2011 in the second week of the semester. Right after the features were described by Prof. Dr. Dirk Riehle, the developers had to estimate the effort to implement the features within a weekly sprint. In consequence the developers mutually agreed to implement the two features *Register new user* and *Login on FSAhoy*.

The POs had to clarify a few questions concerning the localization and password setting of FSAhoy during the first sprint week before the user stories could be implemented by the development team. Prior to the first release, the implemented features were presented by the week's review manager. The features were in this respect tested upon their acceptance criteria and signed off by the lecturer, who was the PO for the first week. The release process itself was explained and carried out by the senior developer of the AMOS project. He was assisted by a PhD student during the release preparation.

4.2.3 Sprint 2

For the second sprint, the developers accepted the user-stories *Verify user by email* and *Recover the password* to enhance the registration and login process. As the senior developer Frank inspired and convinced the POs to work with the Spring framework in matters of security, it was discussed with the TMs. Lacking an expert for Spring security, it became important to acquire knowledge on that topic. Accordingly, a new feature called *Explore Spring security* was created to afford an opportunity to study Spring security.

In addition, a few minor bugs concerning already implemented and approved features were documented during the second sprint. A major issue was thereby the blocking of the Internet Protocol address (IP address) in the event of an unauthorized login attempt. Not only the IP address blocking was difficult to implement, but also the intended use of this security feature was not applicable. It turned out that securing the login would work best with the CAPTCHA procedure. CAPTCHA is a challenge-response test which is designed only to be solved by a human. The

implementation of CAPTCHA was however postponed. For the time being, it was more convenient to fix the IP blocking bug.

4.2.4 Sprint 3

Since the Spring framework was well received by the exploration of one TM, corollary was to re-implement the login process with the help of Spring security to ensure a higher level of security. This framework was also proposed to facilitate future modifications or extensions to the source code. Besides, Spring is supposed to work well with the open standard for authorization (OAuth). OAuth is a protocol which was in this case a requirement to enable access on OSM user accounts. By using request and access tokens, the user allows FSAhoy to access the OSM account information. This procedure makes it possible to login on FSAhoy via the user account that was created for OSM.

Contrary to previous assumptions, implementing Spring security caused by far more issues as it turned out at the end of the third sprint. What made it worse, was the dependency of the user stories concerning Spring Security and OAuth. In order to use OAuth, the Spring framework had to be running first. Since there was a major problem in applying Spring Security, the OAuth features could not be realized. The development team was consequently stuck in implementing new features and mainly focused on the finding and fixing of bugs throughout the third sprint.

4.2.5 Sprint 4

The sprint number 4 was unlike other sprints before and afterwards. There were certain changes to the SCRUM process taking place. First and as aforementioned students could take part in the AMOS project of 2011 as SPO. Therefore students filling the role of a SPO were briefed, how to write user-stories and define acceptance criteria. Secondly, it was the first time in this project that major issues arose that kept the developers from putting new features in place. The SPO, who was responsible for this sprint, was advised to take the feature *Implement Login with Spring Security* back to the Product Backlog after it was rejected and to present it again to the TMs.

Thirdly, the sprint number 4 lasted two weeks instead of one week. Compared to the usual interval, this sprint was extended by one week due to the absence of the SM.

Having all these facts in mind, the developers decided to implement features that were accounting for 17 story points. The Implementation of Spring Security was evaluated with the remaining effort of 5 story points, which makes a total effort of 8 story points. Besides, features for the creation of a logbook and trip were accepted. In addition, a developer responsible for the layout of the webpage was asked to implement the new GUI of FSAhoy after it was designed by a student specialized in graphical design.

4.2.6 Sprint 5

After the biweekly sprint, Spring Security was finally properly implemented. The new layout of the webpage was also available but still lacking implementation and compatibility in some parts, while new ideas about graphical elements were arising. IP blocking, as it turned out, would not work with Spring Security in an acceptable manner. Seeing that a lot of time and effort were invested in finding a solution for blocking unauthorized access by IP addresses resulted in the decision, to dismiss this idea and focus on more significant features. So features such as to create trips, logbooks and daily logbook entries on the social network for sailors were first priority in that sprint. A few of them could not be signed-off due to missing test cases in the previous sprint. These features had to remain in the sprint backlog and wait upon the fulfillment of the acceptance criteria.

4.2.7 Sprint 6

Subsequent to reaching clarification in the cardinality of trips, logbooks and logbook entries, the database model was refined to meet all predefined criteria. It had also to be determined in this context, which attributes or data types are allowed in the data model and whose of them are mandatory. This extra effort and the requirement to specify user instructions at the logbook entry level made it necessary to estimate another 5 story points.

Apart from the logbook feature, the cooperation with OSM demanded to implement the access and login via OSM. In contrast to the third sprint when the linkage on OSM failed due to issues with Spring Security, all preconditions were established now. In consequence, there was no drawback in implementing the connection to OSM.

The security issue was not solved with the work and discussion on Spring Security. There was still missing an effective security mechanism. Therefore the idea of the challenge-response test was reactivated and about to be implemented with the help of CAPTCHAs.

4.2.8 Sprint 7

The seventh sprint was mainly about the integration of features that were not working as supposed to. The login via OSM was unsuccessful alike the integration of CAPTCHA's. These issues had to be fixed first. Important for this sprint week was also, to make basic functions available at the user profile. This means, the user should be able to see his profile page after the login process and be allowed to change his profile information or add more and specific information to the profile page.

A significant role for the further course of development played the telephone conversation with the customer. He suggested a few changes to certain features of the FSAhoy domain model. First, the logbook ought to provide the functionality to upload GPS tracks. This information can be used to facilitate the trip planning. Hence, sailors are merely required to upload their GPS information to create a trip instead of entering trip coordinates. Second, he was explaining the key user attributes that are meaningful for sailors in a social network. A photo of the sailor, the sailing skills and experience as well as the travel interests are the most valuable information for sailors according to Mr. Bärlocher. Last but not least, he drew the big picture of FSAhoy. In his opinion, OSM ranked foremost as the central element of FSAhoy. All other features should be provided as add-ons to an interactive map of OSM. He proposed to integrate a FSAhoy button within the menu bar of OSM.

The MPOs of the AMOS project were analyzing the feedback of the customer and drawing consequences for the FSAhoy project. The feedback given by Mr. Bärlocher encouraged the MPOs to put more effort on the logbook. Since the logbook was already implemented, the next step was to connect the logbook to the map of OSM. The use GPS information was described in a feature that was kept in mind for a later stage. For the moment, it was essential to integrate the map of OSM in the social network. The features concerning the user profile were adapted and extended by the information of the customer. One of the SPOs was informed to change the user-stories according to the input of the customer. The new versions of the user-stories *User profile* and *Update user profile* were in this way directly updated in the Sprint Backlog.

4.2.9 Sprint 8

The sprint number eight adapted a few ideas from the customer and responded to a major bug. Some users of FSAhoy claimed that they never could login successfully on FSAhoy. However, the login and a smooth registration process are vital to a social network. Tracking and analyzing this bug showed that the registration email required rework. It had to be made sure that emails send for registration purposes are not filtered as spam and that a verification of an email is required in order to activate the user account on FSAhoy. The professed goal to design a spam-free registration email was consequently a new user-story for the weekly sprint.

The connection to OSM was now about to get implemented as it was requested. The use of Open Layers was a way to embed the world map of OSM in FSAhoy. The map itself was supposed to be dynamic so that users can navigate through the map by zooming in and out. In addition, users of the social network for sailors should be able to plan sailing trips on the map. For that purpose, it was necessary to implement a route planer that allows for marking the starting point, stopovers and destination of a trip on the embedded map. It was the developer's job to integrate an existing trip planner in a way that user-created routes on the nautical chart are visible in FSAhoy and waypoints are saved in a database.

Apart from those features, further work was needed at the user profile. The upload of a user photo was still missing. For the moment there was just a dummy picture as placeholder available. Hence, a new user story was written to enable the upload and display of uploaded pictures at the dashboard of the user.

4.2.10 Sprint 9

The last two sprints aimed at making FSAhoy self-contained. That means there were less new features compared to the sprints before. Although the total number of features was increasing, the estimated effort per feature was declining. A simple explanation for this event is the fact that there were just minor changes needed in order to improve the usability and to work on the bug fixing. Features like *Update info different color* or *Usability Issue with Registration* were in this regard designed in order to enhance the user-friendliness of FSAhoy. A few features were also proposed to make the social network more interesting and well-known. The philosophy and legal obligations of the portal should be explained by the user stories *Welcome Introduction Text*, *Disclaimer* and *Policy*.

New functionality was in addition added to the trip section. Next to saving and storing trips, it was also essential to the update and to view the trip. In the context of the daily logbook, the user was also enabled to transfer coordinates from the OSM map directly into a new daily logbook entry form. When the user was marking a trip on the map, there should be a button that allows for taking the trip coordinates to a new logbook entry. Implemented was also the feature how to *Change password*. Up to sprint number 9, it was only possible to send a new and automatically generated password via Email. Now it was made possible to change the password directly on the website of FSAhoy.

Another way to optimize the look and feel of the Webpage was to review the design of the webpage by documenting evident bugs. This review process was taking place in cooperation with a MPO and TM specialized in the implementation of the graphical design. Existing bugs were in the following tracked in the bug tracker and prioritized by the MPOs.

4.2.11 Sprint 10

Sprint number 10 was the last sprint in the AMOS project of 2011. Almost the same procedure as last sprint was applied to clean up the code and prepare for the release of the first product prototype. No risk was taken when the features were presented by the SPOs. By all means, it should be avoided to work on features that cannot be implemented entirely within the last sprint.

This time there were even features evaluated with half a story point such as *Message stating activation required* and *Profile picture on the profile page*. Half a story point stands for something in between no effort and minimal effort (see Chapter 4.3). Nonetheless, there was also one feature with an effort of five story points. This might seem risky at first sight but it was accepted and successfully implemented as all other features of the last sprint.

The logbook and trip planning were not completed before the entire functionality to create, view, edit and delete these elements was integrated. The related user-stories *Delete trip*, *View & edit logbook* and *Delete logbook entry* accounted for nine story points altogether whereas five story points were given to the feature *view & edit logbook*.

Remaining and highly prioritized bugs were fixed at the same. The presentation of the user name on the dashboard and the validation of date entries were for instance minor improvements that were marked as bug fixes in order to optimize the usability. Since all features of the last sprint could be signed off by the POs, the major version 1.0 was released.

4.3 Development Speed

Scrum uses a metric to track and forecast the progress of a team or project. This relative measurement is known as the velocity and based on story points. Story points are the expected effort that is needed to complete a user story within a given sprint. The total number of story points for each user story is determined by the team

during the Planning Poker and can follow the Fibonacci numbers. The AMOS project of 2011 is relying on the Fibonacci numbers 0, 1, 2, 3, 5, 8 and 13 (see Table 4).

Table 4: Story points and their meaning in Scrum

Story Points	Meaning
0	No effort
1	Minimal effort
2	Small effort
3	Medium effort
5	Large effort
8	Very large effort
13	Too large effort

The story points are ranging on a scale from “no effort” up to “too large effort”. In consequence, a story point of 0 means no effort is required whereas a story point of 13 indicates a too large effort is being required. A user story that was evaluated by 13 story points can be split up in order to get smaller user stories, which can be completed in one sprint. Opposed to absolute values such as hours or person days, story points represent a relative measure. The relative value of story points is, according to Dr. Jeff Sutherland, even more accurate and less variable than those of absolute values [33]. The co-founder of the Scrum development process exemplifies in his blog that the story point estimation method at a telecom company was many times faster and at least as precise as the waterfall estimation practice.

4.3.1 Velocity

The velocity is an indicator of the development speed and an asset for the release planning in a project. To be more precise, the velocity v is calculated as the completed story points s within a sprint t .

$$v = \frac{s}{t}$$

It is important to note that only story points of completed features are counted in means of the velocity. If user stories cannot be signed off due to missing test cases or failing acceptance tests, their story points will not be taken into account at the time

the velocity is calculated. The seventh principle behind the Agile Manifesto postulates “working software is the primary measure of progress” [14] and thereby emphasizes the role of working and completed features, respectively. By estimating the development speed using completed work measured by story points, a more reliable point can be made than predicting the progress by hours.

However, the velocity is often volatile and may vary during the development of a project. It often takes a few sprints until the velocity becomes stabilized. Cohn argues, new teams or new-product development projects may need up to three sprints to attain a more steady velocity [34]. Nevertheless, the information presented by the velocity, even though it might be biased, is crucial to forecast what a development team can achieve during the next sprints and vital for a realistic release planning.

It is also interesting to analyze the difference between the story points that were estimated during the sprints and those that were actually achieved. Table 5 highlights the planned story points, the completed story points and the difference between both of them for the AMOS project of 2011.

Table 5: Planned and completed story points in the AMOS project of 2011

Sprint	Planned story points	Completed story points	Difference in story points
1	6	6	
2	11	11	
3	16	3	-13
4	14	8	-6
5	15	7	-8
6	18	8	-10
7	14	14	
8	18	18	
9	18	18	
10	16	16	
Sum	146	109	-37

Figure 16 depicts more clearly the different between the planned and completed number of story points. However, in the first instance, both story point series will be examined independently.

Taking a closer look at the dark blue graph that describes the story points which were estimated by the development team, it can be stated that it takes about three sprints until the estimation becomes more stable. From the third sprint on, the accepted story points are within the range of 14 to 18. Taking the standard deviation of planned user stories as a basis, the hypothesis of a more stabilized estimation from the third sprint on can be validated. The standard deviation is declining from 3.75 to 1.73, if the first two sprints are neglected. This phenomenon was, as aforementioned, also described by Cohn for the development of the velocity. A reason for the decreasing variation in the estimation of story points after the second sprint might be due to the fact that the development team had to get to know each other first and become familiar with the agile practices of Scrum.

The light blue graph which is displaying the completed story points in Figure 16 is characterized by a greater fluctuation than the graph of planned story points. 5.32 is the standard deviation of completed story points and thereby surpassing the standard deviation of planned story points. A reason therefore can be found in the analysis of the actual project.

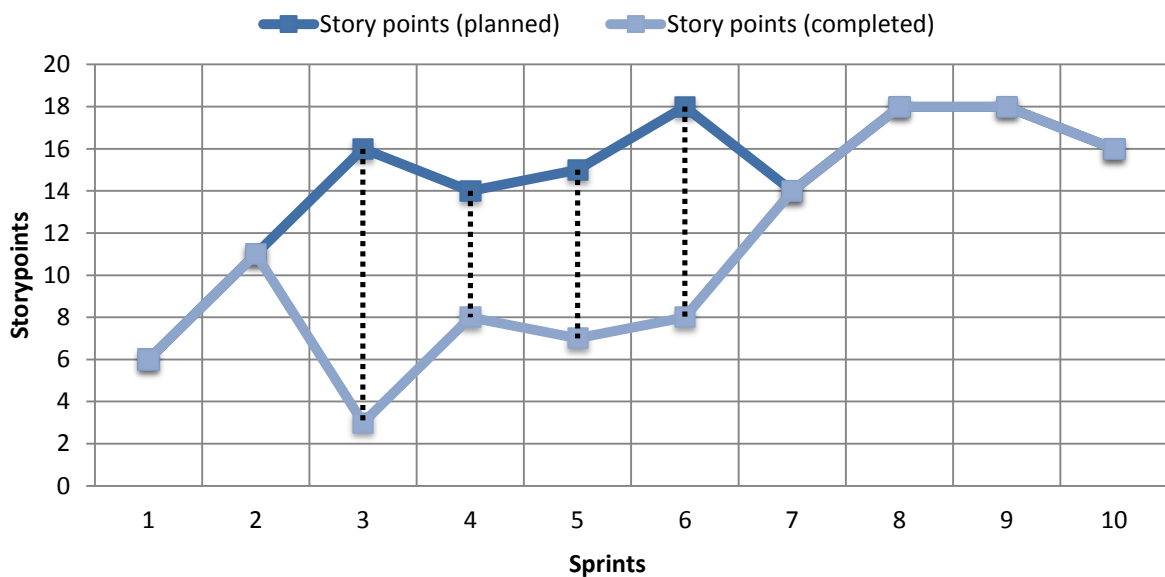


Figure 16: Planned and completed story points in the AMOS project of 2011

It is no surprise that the average velocity of 10.9 for the completed story points is less than the average velocity of 14.6 for the planned story points. In every project there are issues or impediments that can slow down or almost stop the development progress.

The same situation emerged during the development of FSAhoy in the third sprint. Since the Spring framework could not be implemented within this sprint, there was also no progress on the dependent features *Authorize access on OSM* and *Login via OSM*. The fixed IP blocking bug was the only positive aspect of the third sprint. Thus, the low point of three story points is on the one hand the negative result of the issues related to Spring security and on the other hand the positive outcome of the fixed bug that accounted for three story points. The following sprints 4, 5 and 6 resemble as well a remarkable difference between planned and completed story points. Missing test cases were the reason to reject a few smaller features that concerned the implementation of the design and the creation of trips and logbooks in the fourth sprint. There was only one feature that was not accepted in the Sprint 5. Four out of five features were completed. However, a feature evaluated with a very large effort of 8 story points still needed some rework. This feature *Daily logbook entry* turned out to be more complex than expected. Only 8 out of 18 planned story points were completed during the sixth sprint. The feature to login via OSM just as the security feature using CAPTCHAs continued to cause problems and could not be signed off. At the same time, additional effort was required to work on the user profile, as one of the acceptance criteria could not be fulfilled. The graph of completed story points aligns with the graph of planned story points after the plummet in the sprints 3 to 6.

4.3.2 Sprint Forecasting

The data about completed story points obtained from Table 5 can be valuable for a number of purposes. Besides comparing planned and completed story points, the completed story points are also useful for comparing the velocity of teams in a project or predicting the amount of work that will be completed in upcoming sprints. Since there was only one development team available in the AMOS project of 2011 and the

comparison between FSAhoy and other agile projects might be skewed by the nature of the these projects, the release planning is the focus of the analysis.

Given the SPSS output in Table 6, the sample size (N), velocity (Mean/Velocity) and standard deviation (Std. deviation) for completed story points can be observed. The velocity of the team and the standard deviation can be used to compare planned and completed story points as described before.

What draws more attention is the second row of the table. This row explains the influence of the third sprint on the aggregated statistic. Neglecting the third sprint (N=9), a higher velocity of 11.78 and lower standard deviation of 4.816 are obtained. This adjusted data set makes a more realistic output opposed to what was analyzed before (see Chapter 4.3.1). At the same time, a higher validity can be retrieved in the release planning if the statistics of adjusted story points are used.

Table 6: One-Sample Statistics for completed story points

Story points	N	Mean/Velocity	Std. Deviation	Std. Error Mean
Completed	10	10.90	5.322	1.683
Completed (adjusted)	9	11.78	4.816	1.605

Table 7 presents a One-Sample Test that is more detailed and provides a deeper analysis of completed story points. Complying with Mike Cohn’s approach [35], the 90% confidence interval is computed to forecast the velocity for the upcoming sprints.

Table 7: One-Sample Test for completed story points

Story points	Test Value = 0					
	t	df	Sig. (2-tailed)	Mean/Velocity	90% Confidence Interval of the Difference	
					Lower	Upper
Completed	6.477	9	.000	10.900	7.82	13.98
Completed (adjusted)	7.337	8	.000	11.778	8.79	14.76

From Table 7 it can be concluded, with 90% confidence, between ~8 (7.82) and ~14 (13.98) story points will be completed in the next sprint. If the adjusted data of the second row is used instead, it can be argued that there will be between ~9 (8.79) and

~15 (14.76) story points achieved in the next sprint in 9 out of 10 times. At this point, it is possible to continue the estimation of the range of story points which could be completed in the upcoming sprints. This can be done by multiplying the number of sprints by the values of the confidence interval [35]. However, that is not the objective of this analysis.

4.3.3 Burndown bar

The burndown chart is an effective tool to measure the progress of a project and to predict the release of a product. Roman Pichler considers the factors “time” and “remaining effort” as the two main elements of a release burndown chart [36]. The time is usually displayed by order as the number of sprints while the remaining effort resembles any artifact that can determine the effort of work. The feature quantity is used in this thesis as a measurement for the remaining effort. Story points or hours of work would be possible indicators as well. The burndown bar chart, as it is illustrated by Figure 17, describes the progress of FSAhoy.

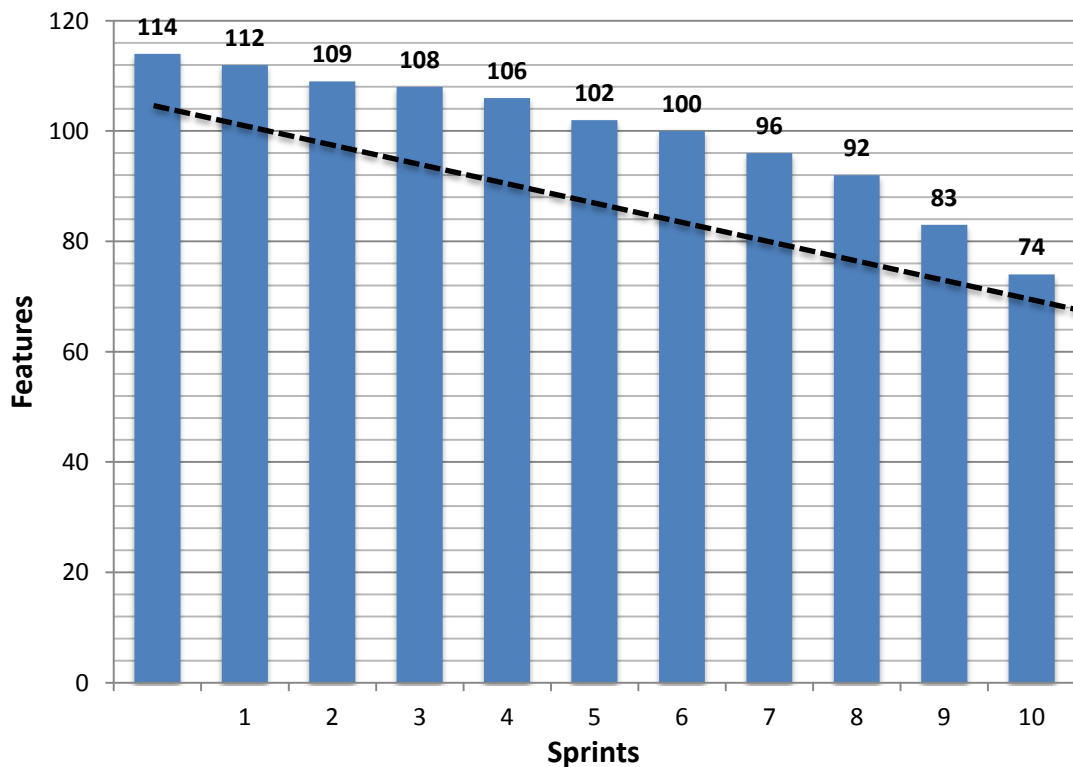


Figure 17: Burndown bar chart in the AMOS project of 2011

The burndown bar starts with a planning value of 114 features before the actual development begins to take place in the first sprint. The following few sprints are characterized by a slow decline in the number of completed features. Figure 17 conveys the impression that the team formation process took time until the sixth sprint. As mentioned earlier, another reason for a slow project start can be found in Spring security impediment. Especially the decrease in numbers of achieved features during sprint 2 and 3 are expressing the struggle with the Spring framework. From the seventh sprint on, the project progress is getting more and more constant. There are at least four features completed in each of the following iterations.

The dashed line which portrays the burndown trend helps to forecast the release. Following the burndown trend, it is possible to guess the final release date after the last recorded sprint. If the number of remaining features (74) is divided by the average number of completed features per sprint (4) a value of 18.5 is given. This number represents the timeframe in sprints that is required to complete all the remaining 74 features. As a result, approximately 19 additional sprints are needed to release the final product. However, this assumption is subject to the condition that there is no variance in the development speed and there are no additional or dropped features or any other factors that may have an impact on the project's progress. Burndown charts are after all primarily used as an instrument for the release planning since the development speed is better described by the velocity of a development project [37].

5 Mydosis and FSAhoy

The AMOS course takes place each summer semester at the University of Erlangen-Nürnberg. It is a semester-long course that consists of two parts. The first part is a lecture that teaches the theory of agile methods and the concept of Open Source software development [1]. The second and more practical part of the AMOS module describes the development of a software solution. Opposed to the theoretical part of the AMOS course, which faces merely incremental changes, the domain of programming project is different for every year.

The first AMOS project started in 2010 with the development of an electronic database for dosage information (Mydosis). The web service that was developed using the agile method Scrum provides dosage information for pediatricians, doctors, nurses and students [5]. Mydosis was designed in collaboration with Dr. Bernitzki from the University Clinic of Cologne and is accessible from every web browser and by the correspondent smart phone App on Android devices.

It was about one year later, in May 2011, when the kick-off for the second instance of the AMOS course took place (see Chapter 4.2). The idea of the AMOS project of 2011 was to develop a social network for sailors where nautical experiences and information is shared. Further details and the domain model of FSAhoy were described in Chapter 4.1.

Both projects Mydosis and FSAhoy, bear a lot of similarities but to some extent also substantial differences. In order to learn from the pitfalls and success stories of these agile projects the peculiarities of each project have to be understood first (see Chapter 5.1). As soon as more knowledge is acquired, both projects can be compared on a defined set of criteria (see Chapter 5.2). The results of this analysis can help to suggest improvements for the AMOS course in future instances (see Chapter 6).

5.1 Course setting

The first approach is now to identify the general setting of the two courses. Table 8 outlines the key facts of the AMOS project of 2010 and 2011. A more comprehensive analysis of the factors of the AMOS projects will outline in the following that almost all of these observed fields draw the picture of two distinct setups.

Table 8: Key facts for the AMOS courses of 2010 and 2011

Factors	AMOS 2010	AMOS 2011
Components	Web service and Mobile App	Web service
Domain	Healthcare	Sailing Sport
Participants	Computer science students	Computer science and Business students
Roles	Scrum Master, Product Owner, Product Owner Proxy, Team Member	Scrum Master, Main Product Owner, Student Product Owner, Team Member
Technology	Java, Apache Tomcat, db4objects, PostgreSQL, Spring	Java, Apache Tomcat, Open layers, Spring, OAuth
Timeframe	11 weekly Sprints	10 weekly Sprints

5.1.1 Components

Referring to the components of the AMOS projects, Mydosis was made available as a Web service and mobile App while FSAhoy was only delivered as a Web solution. Although it was intended to develop an App of FSAhoy for the android market, there were not as such enough time and developers available to realize this plan. In the end, only Mydosis was released for both of platforms, the web and the android operating system.

5.1.2 Domain

The application domain of Mydosis is pointing at the healthcare industry. That is an industry which is originally more profit-oriented than recreational sports such as sailing. The main target groups of Mydosis are pediatricians or other doctors with a

presumed business interest [5]. FSAhoy is primarily addressing non-commercial sailors and pleasure boaters on an amateur or professional level (see Chapter 4.1.2). The difference in interest groups could scarcely be greater.

5.1.3 Participants and Roles

The active participants in the AMOS projects are varying through 2010 and 2011. In the first course, there were only Computer Science students involved in the software development process. Even though that Business students or students with a major background in Business studies were enrolled in this course, they just attended the lecture and were not supposed to take over an active part in the development of Mydosis. Next to these passive observers, the familiar Scrum roles such as SM, PO and TM were assigned. New was the role of the Product Owner Proxy. Markus Stipp was assumed that role in order to represent Dr. Bernitzki who was the PO of the AMOS project of 2010 [5].

A notable change in the composition of the Scrum roles was taking effect with the AMOS project of 2011. In response to the feedback from the student evaluation at the end of the AMOS project of 2010 and the increasing number of Information System students, new Scrum roles were established. Subsequently, master students of Information Systems with little or no experience in development were now granted the possibility to take on the role of a SPO. A SPO that falls within this definition is a student who is taking over the role of PO for one sprint. The SPO is thereby instructed and guided by a MPO. The MPO can be a senior student or someone who has gained previous experience in Scrum. The MPOs in the AMOS project of 2011 were composed of a senior Information Systems student and the author of this thesis.

5.1.4 Technology

Technology-wise there were fewer differences ascertain. Both projects were developed with the help of the web-based source code repository SourceForge on an Apache Tomcat. The Figure 18 demonstrates that the central programming languages were Java and XML whereas slightly more code was written in XML at

Mydosis. Java was next to a few other minor languages predominant in the development of FSAhoy.

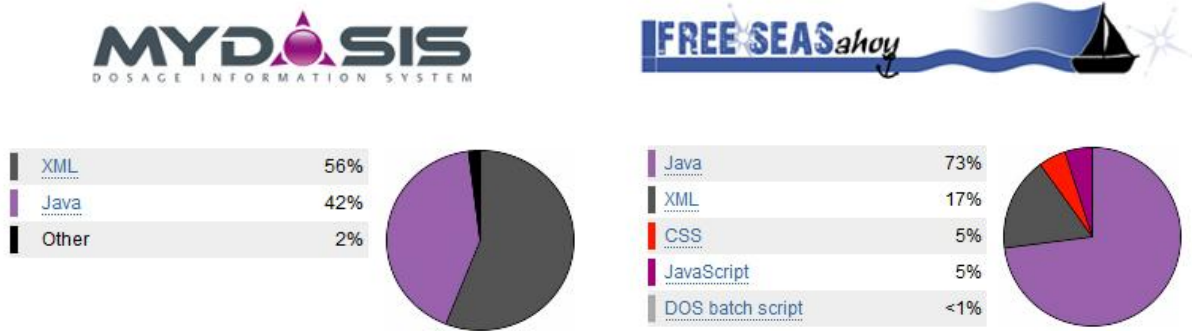


Figure 18: Programming languages of Mydosis and FSAhoy [38]

Figure 18 is composed of two analyses generated by the software directory ohloh for Open Source. It analyzes the project source code and determines the programming languages of each project. Excluded are comments and blanks [38].

To better understand the proportions of the pie charts, a more profound analysis of the software architecture is made. The key element of the software architecture in Mydosis is a database where all the dosage information is stored. PostgreSQL was deployed for this reason on the server application side of Mydosis. The mobile Android client was using the database db4objects. The communication with each of the databases and the structure of the documents was mostly described in XML. That is probably why XML is playing a superior role in Mydosis.

FSAhoy itself was not based on a database to such a great extent. Certainly, there was also a database that was used to store and manage customer information. More important was though the implementation of OAuth and OpenLayers. For these interfaces it was necessary to make use of Java and Java Script. Java was also prevailing due to the fact that a lot of new functionality had to be built. There were quite many diverse features implemented in FSAhoy. The registration process, user profile and trip planning are just a few examples. CSS was also attaining further attention in FSAhoy as one of the TMs was specialized in the layout and style of the social network.

Last but not least, the Spring framework became a big issue in both projects albeit it was implemented for different purposes. Spring was implemented in Mydosis to handle the html interface [5]. In FSAhoy, Spring was used for security purposes. The technology that should ensure a high level of security was causing a lot of problems as mentioned before (see Chapter 4.2.4).

5.1.5 Timeframe

Depending on the timetable of the summer semester, there are about 13 to 14 weeks of classes. The first class in the last two instances of the AMOS project was used as an introduction [39]. The last class in the semester is a class wrap-up. In consequence, there remain 11-12 weeks for the actual development process in the case that there are no further limitations.

The first AMOS project started in 2010 in the third week of lectures and came up to 11 sprints by the end of the semester. In 2011, the Scrum development process was started in the second week of the semester. However, the development period of FSAhoy was reduced to 10 sprints due to the single absence of the SM. One sprint was subsequently planned for a biweekly iteration. The difference between 10 and 11 sprints is in particular important when the velocity and completed features of both AMOS projects are compared as follows.

5.2 Project Performance

Second to getting a general idea about the setup of the AMOS projects, is a deeper analysis of the performance and progress of both projects. The velocity and Lines of Code (LOC) are two instruments which can be used to get a better feeling for the performance of a team or project.

5.2.1 Velocity

So far it was analyzed, how FSAhoy was doing in terms of velocity and completed features. It is only in comparison with similar projects, that this obtained data can prove itself valuable. Therefore, the next step is to evaluate how FSAhoy performed in comparison with Mydosis and to draw a conclusion from that analysis. While

comparing both projects, the limitations and constraints that are concurrent due to different project settings have always to be kept in mind (see Chapter 5.1).

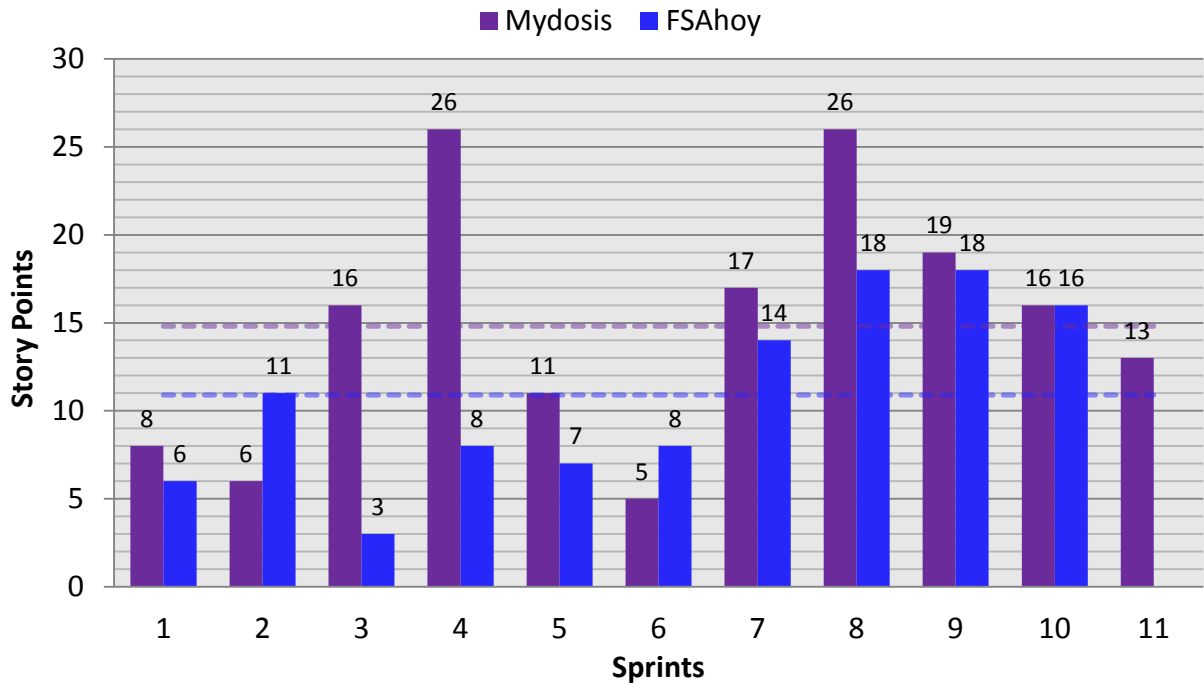


Figure 19: Completed story points of Mydosis and FSAhoy

Figure 19 visualizes the progress of the two AMOS projects by completed story points per sprint. The average velocity of 14.8 for Mydosis and 10.9 for FSAhoy are signaled by a purple and blue dashed line. Before the ups and downs are studied more detailed, some general statements can be made.

It is evident, that the average development speed is greater for Mydosis. A reason therefore might be found in the deviant number of participating developers. The Mydosis project started in 2010 with 11 developers divided into two teams whereas FSAhoy had to manage the development process with 6 developers.

In order to get hold of this situation, the given graph has to be more specified. If the story points of Mydosis are allocated according to the achievements of each team, a graph results that is more suitable for a comparison (See Figure 20). Since both teams in the Mydosis project were merged after the seventh sprint, it is appropriate to regard the development period of the first to the seventh sprint. The new graph in

Figure 20 displays the completed story points per sprint and the average development speed of both Mydosis teams in relation to FSAhoy.

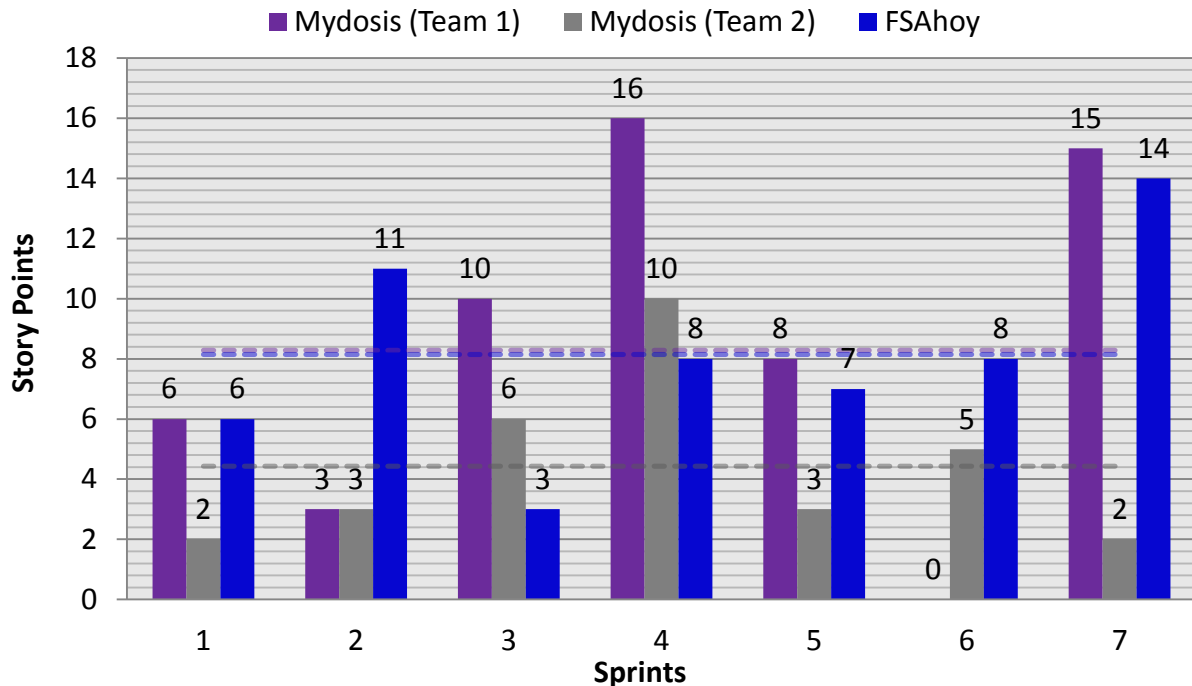


Figure 20: Completed story points of Mydosis teams and FSAhoy

Team 1 of Mydosis consisted of six developers and was responsible for the development of the Mydosis Web service. Team 2 was composed of five developers that were mainly developing the Mydosis Android App. A direct comparison of the Team 1 of Mydosis and FSAhoy is more prudential than comparing the overall Mydosis project with FSAhoy. Both, the Team 1 of Mydosis and the FSAhoy project had six developers and were mostly working on a Web service.

When the bars of Team 1 and FSAhoy are evaluated, it is recognizable that both teams are outperforming Team 2 in means of the velocity. Team 1 has an average velocity of 8.3 and FSAhoy is close behind with a value of 8.1. The average development speed of team 2 is characterized by a grey dashed line and with a value of 4.4 much lower than the ones of team 1 or FSAhoy. The lesson is clear if the analysis is limited to Figure 20: “Team 2 is not performing as well as Team 1 and there is no significant difference between Team 1 and the FSAhoy team although the

overall velocity is higher at Mydosis.” Latter result is just a consequence of the different number of participating developers in the AMOS projects. If the velocity would be broken down to the development speed per developer, the result might be a different one.

The following Table 9 indicates that FSAhoy was surpassing Mydosis if the comparison is narrowed to the velocity per developer. The average velocity per developer is at FSAhoy (1.82) greater than Mydosis (1.51). This result reverses the picture of a considerably larger overall velocity of Mydosis in Figure 19.

Table 9: Velocity per developer of Mydosis and FSAhoy

Development Speed	Mydosis	FSAhoy
overall velocity	14.8	10.9
velocity per developer (sprint 1 – 7)	1.16	1.82
velocity per developer (sprint 8 -11)	2.06	
velocity per developer (in total)	1.51	1.82

Comparing teams or projects simply by the velocity is not always convincing. There are a lot of factors that may influence the story point estimation and completion.

First, velocity can be sustainable. Planning with a velocity that remains constant during the development period might be a more successful approach than pushing the project team towards a peak velocity. A team working at a very fast pace for a long time can burn out and neglect the important qualities of a sustainable product in the end. Ralf Wirdemann describes the values of a sustainable velocity with the practices of refactoring and testing [40]. In the long run, a product is more successful if the software is working fine and developed in a way that allows extensions or changes to the codes to be implemented easily. A product, which is built with a high velocity, might compose a lot of features but lacks the quality of a well-factored and tested product.

Second, projects often come with different standards to estimate and evaluate the completed story points. Especially the definition of “done” may vary across projects and result in a divergent velocity [6]. Successfully executed acceptance tests may be a requirement for POs to accept or deny user stories [28]. That means, a feature is only implemented and the velocity is only taken into account if the user story passes all the acceptance tests. Other projects may also take story points into account if features are implemented without passing acceptance test or if only parts of the features are working.

Third, changes in the team composition can have an impact on the development speed. According to Mike Cohn, the velocity is very likely to drop when the team size changes even if the team size goes up [35]. He explains that effect with the increased communication that is necessary to get new TMs productive. Interestingly are changes in the team size had a positive effect for the Mydosis project.

Contrary to Cohn’s assumption, the velocity of the Mydosis project was even increasing in the eighth sprint after the both teams were merged a sprint earlier. Merging the two teams was preceded by the decision of two developers of team 2 to leave the AMOS project of 2010 during the seventh sprint. The figures in Table 9 surprisingly show that the average velocity per developer was lower when there were two teams and 11 developers opposed to 9 developers and only one team remaining. The increase of the velocity per developer from 1.16 up to 2.06 after the merger may probably also be associated with the underperformance of team 2 during the first few sprints or less challenging user stories after the seventh sprint.

The changes in the team size during the AMOS project of 2010 had another side effect that is visible if the ups and downs of the two projects are regarded in Figure 21. The curve shape of the Mydosis project is described by two peaks and a low point. The first drop in the sixth sprint was, according to Markus Stipp [5], a consequence of the revised domain model. Only the development team 2 responsible for the android application could work on new features in that sprint. In fact, this argument is well-reasoned if Figure 20 is regarded where only five story points were completed by the second team. The revision of the domain model may

have been frustrating and a reason for the two students of Team 2 to leave the Mydosis project shortly afterwards.

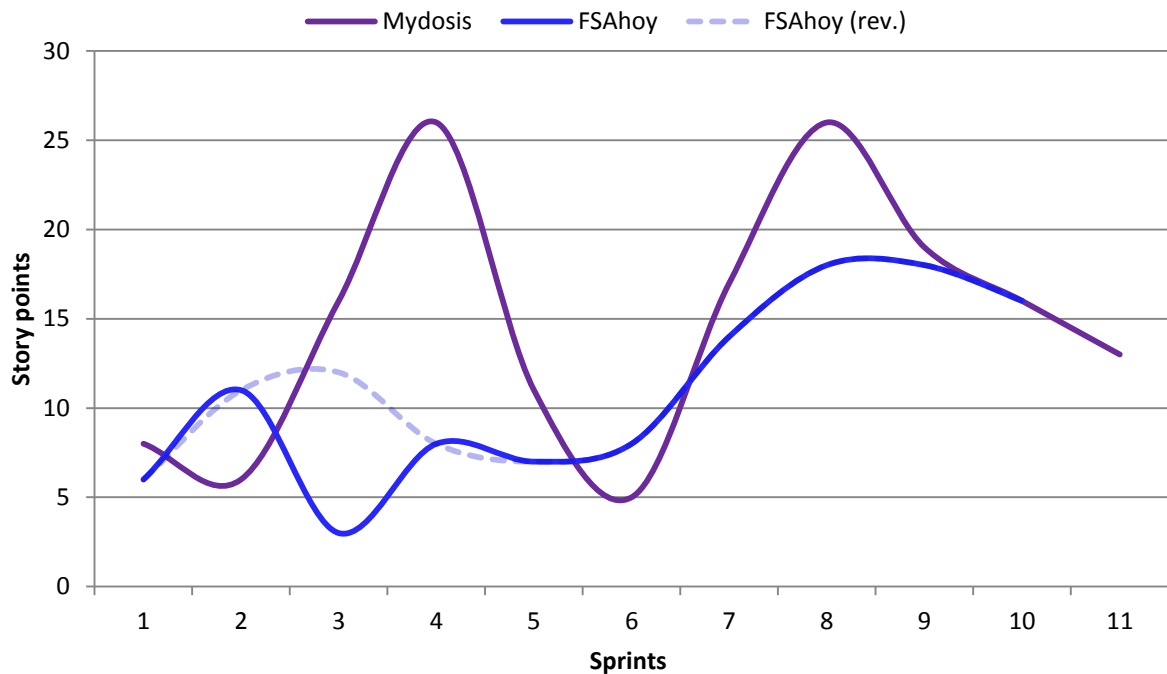


Figure 21: Story point curve of Mydosis and FSAhoy

In contrast, FSAhoy shows a more consistent and predictable development progress. There were no changes in the team size during the semester-long AMOS course in 2011. The development speed of FSAhoy, which was increasing and almost reaching at steady state throughout the semester, was not negatively influenced. There were also issues with Spring security in the third sprint and a minor delay in the fourth and fifth sprint due to the connection and authorization on OSM which contributed to a lower development progress in the sprints three to five. If the problems with spring security in the third sprint are omitted and replaced by the average velocity, a more constant picture can be drawn. This revised graph in Figure 21 is presented by dashed light blue line. It indicates that projects without key changes in the domain model or team size are more predictable and likely to follow a consistent development speed. Unfortunately the comparison of two projects is not enough for a representative statement.

Going a step further, the two projects can also be analyzed by the distribution of story points. It is not surprising that the two graphs for the AMOS projects in Figure 22 are very similar. This is usually the case when the estimates of story points per user story do not vary a lot throughout sprint iterations. If user stories meet the INVEST criteria they ought to be small enough to be estimated with a small or medium effort. User stories that are evaluated with a too large effort are often divided into smaller ones, reassessed and accepted by the TMs.

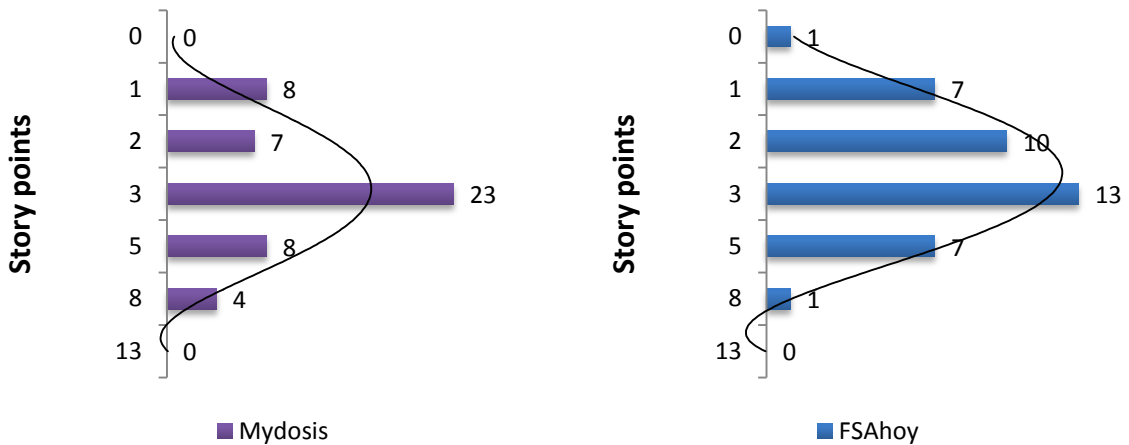


Figure 22: Story point distribution of Mydosis and FSAhoy

It can be easily concluded that both graphs in Figure 22 are characterized by a mode of 3 story points. The medium effort of 3 is therefore the most common estimate in the two projects. It is followed by the story points of 1 and 5 for minimal and large effort in Mydosis and the story point of 2 for a small effort in FSAhoy. The mean of Mydosis is 3.26 story points and compared to 2.79 story points of FSAhoy slightly higher which results in a positively skewed distribution of Mydosis and a negatively skewed distribution of FSAhoy (see Table 10). Looking at the highlighted trend line, both projects are approximately normally distributed with a median of 3 story points.

Table 10: Statistics for the story point distribution of Mydosis and FSAhoy

Statistics	Mydosis	FSAhoy
Mean	3.26	2.79
Median	3	3
Mode	3	3
Std. Deviation	1.850	1.609

In particular, the equal mode and median as well as the normally distributed graphs corroborate the hypothesis that the curve of achieved features in Figure 23 resembles the curve of story points per sprint in Figure 19. Having said this, there is still a noticeable event in the last two sprints of the FSAhoy project. The rate of completed features has more than doubled in the sprint 9 and 10 compared to the previous sprints. Instead of four features that were achieved on average per sprint, there were nine features completed (see Figure 23). An explanation for the amplitude of the last two sprint sessions can be found by investigating the context of these sprints. The majority of features that were part of the last two sprints were designed in order to improve the usability and work on the bug fixing of FSAhoy.

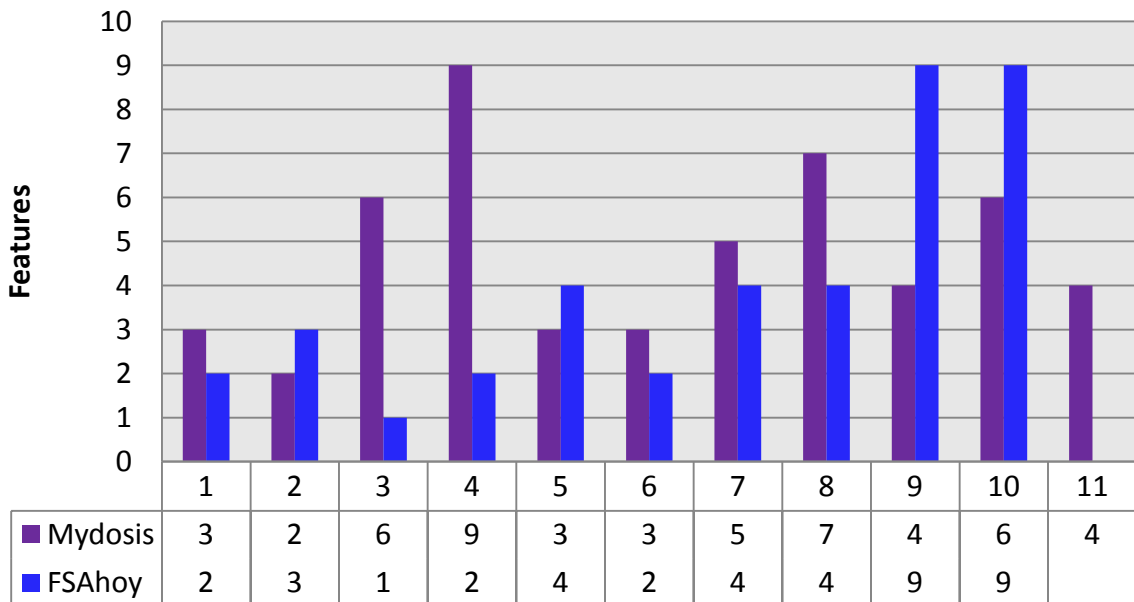


Figure 23: Completed features of Mydosis and FSAhoy

Fixing bugs can sometimes be done during the weekly development process if there is enough time and resources available. If bugs are classified with a severity level like “major”, “crash” or “block” in the bug tracker, they should be put back to the Product Backlog and get prioritized again. The same action took place for a few features in the last two sprints. Besides fixing bugs, smaller features were implemented in the last two sprints to ensure an enhanced usability and user-friendly product. Being aware that the final release 1.0 is imminent, larger user stories were dismissed for the last two sprints.

The observation of a rising number in features towards the end of a project might not

always be true. The Mydosis project describes a rather uncertain situation for the last four sprints. Two of the last sprints, 9 and 11, were falling below the average of 4.7 achieved features. Sprint 8 and 10 were on the other hand exceeding the average of completed features. Markus Stipp was describing this event in the last few sprints by the effort of the development team to “put all the pieces together” [5]. Bug fixes but also missing key features were part of the discussion and retrospective in the last few weeks. This two-track approach of handling bugs and putting key features together at the same time could explain the ups and downs in Figure 23. However, it is difficult to draw a conclusion. More information would be needed to make a clear statement.

5.2.2 Lines of Code

The final dimension used to compare the AMOS projects is the number of all time commits and LOC. The ohloh-output in Figure 24 is basically reduced to these points.

The number of committers in respect of the all time activity is mentioned as well. Yet those numbers are biased as some of the code contributions were done by the same developer with several accounts. There were only 11 developers available at Mydosis and 6 at FSAhoy.

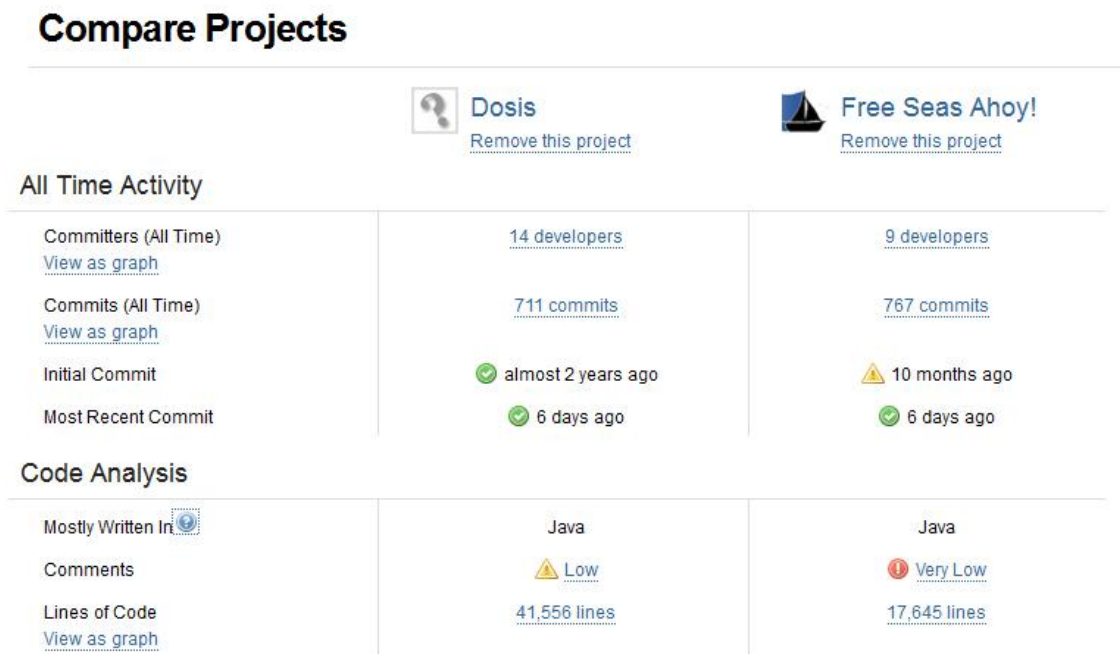


Figure 24: Code analysis of Mydosis and FSAhoy [38]

Comparing the number of commits, it can be figured out that FSAhoy (767 commits) is doing slightly better than Mydosis (711 commits). That means there are indeed 56 additional commits for FSAhoy. A high number of commits can represent a widespread use of practices that are described by the agile method XP. One of the best practices of XP is Continuous Integration. That concept was initiated by Martin Fowler and advises developers to submit and integrate code frequently [41]. If code is committed regularly the system can be integrated and built in a way that feedback is given with immediate effect. The agile mantra also confirms the practice of frequent code commits [11]:

Make it run!

Make it right!

Make it fast!

According to these basic principles, code should be implemented first to make features available and run before the code is refactored and optimized. Bearing the philosophy of agile development in mind, it can be argued that FSAhoy was pursuing the agile development practices to a greater extent than Mydosis. This statement is nonetheless only valid if the way of looking is limited to the commits that were made to the source code.

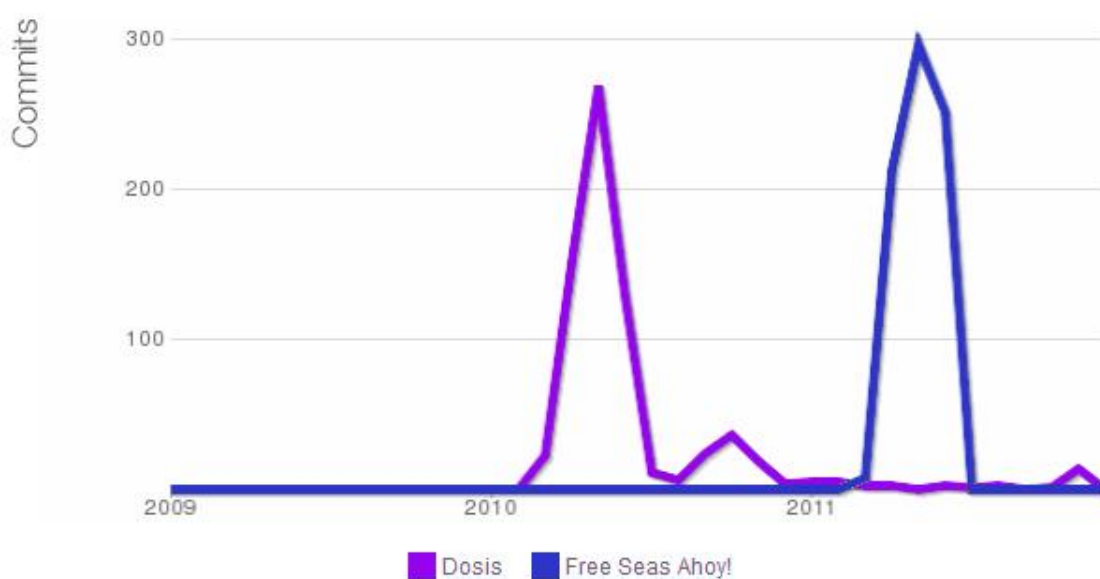


Figure 25: Commits of Mydosis and FSAhoy [38]

Figure 25 presents the number of commits for both AMOS projects given the period 2009 to 2011. There is not only a higher overall value for FSAhoy but also a peak that tops Mydosis. Eye-catching is the fact that Mydosis shows still some activity during the last months in both graphs (Figure 25 and Figure 26) whereas FSAhoy tends to stand still. The reason for the small growth in the number of total LOC and commits at Mydosis is the startup incubation process. There was a successful launch of the Mydosis start-up a few months after the university course was finished. The start-up is still optimizing and working on the code base which is also reflected by the ongoing graph activity.

The total lines of project source code, excluding comments and blank lines are approximately twice as much at Mydosis (see Figure 26). The sharp increase in LOC comes to a total number of more than 40,000 LOC at Mydosis. This number is obviously outnumbering FSAhoy with somewhat less than 20,000 LOC. The depicted trend can account for the difference in the number of developers working on both projects. Twice as much TMs at Mydosis also make up twice as much source code. The calculation that a developer can write approximately 3,500 – 4,000 LOC during one semester draws the conclusion that both projects are almost equal in terms of LOC.

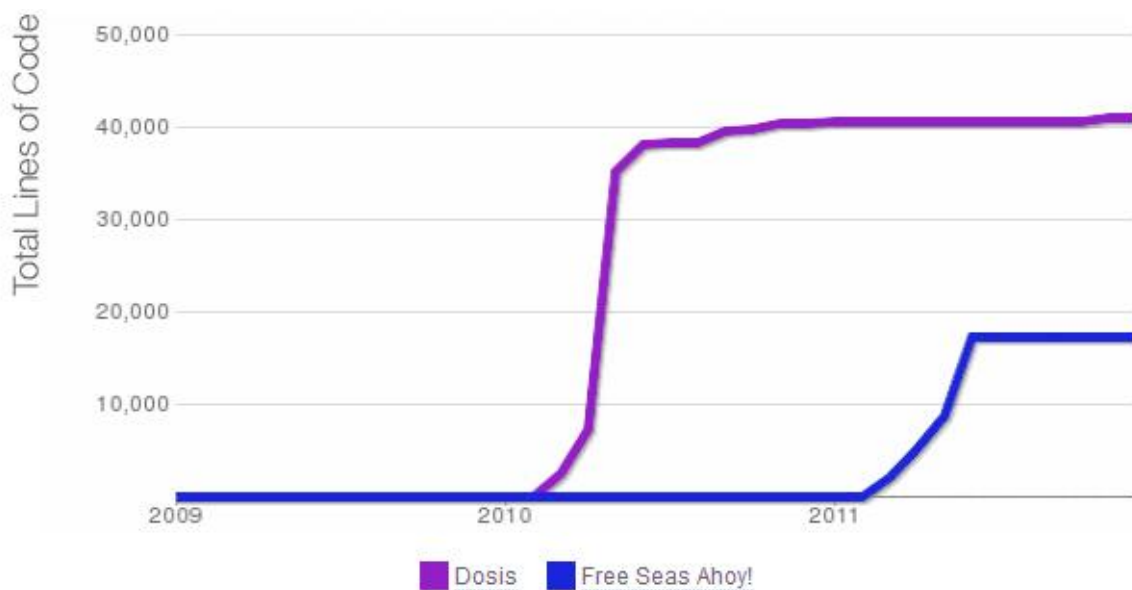


Figure 26: LOC of Mydosis and FSAhoy [38]

6 Discussion

The overarching research question of this thesis is how to improve the AMOS project in future instances. Suggested improvements that are derived from the analysis of the AMOS project of 2011, the comparison between the AMOS project of 2010 and 2011 and literature review are summarized in Table 11. These ideas for improvement are framed as hypotheses and discussed for validation hereinafter.

Table 11: Suggestions for future AMOS projects

Hypothesis	Category	Sub-category	Improvement
H1	Academic course	Motivation	Team building exercises are motivating
H2	Academic course	Team	Small development teams show good performance
H3	Academic course	Distribution	Distributed development does not lower velocity
H4	Scrum methodology	Artifact	Diversifying user stories in the sprint planning reduces risk
H5	Scrum methodology	Artifact	Software tools facilitate the administration of artifacts
H6	Scrum methodology	Practices	Mailing lists work as daily scrum replacement
H7	Scrum methodology	Roles	Proactive customer involvement provides valuable feedback
H8	Velocity	Planning	Documenting planned story points optimizes estimation forecast
H9	Velocity	Domain model	Revised domain models negatively impact velocity
H10	Velocity	Team size	Changes in the team size decrease velocity

The suggested improvements can be categorized in the three categories of the Academic course (Chapter 6.1), Scrum methodology (Chapter 6.2) and Velocity (Chapter 6.3) as it follows.

6.1 Academic course

Chapter 3 explained that the use and adoption of Scrum in an academic setting can differ across projects. Every academic course has distinct underlying principles which require modifications that are specific to the project. The hypotheses H1 to H3 are based on the analysis of the AMOS projects and the comparison between the adaptations of Scrum methods in academic courses.

H1: Team building exercises are motivating

To validate the hypothesis that team building exercises are motivating, the AMOS projects in 2010 and 2011 are compared.

In 2011, there were two team building exercises taking place at the project kick-off before the actual development started (see Chapter 4.2.1). The ball game and construction of a pipeline were not only useful to get to know each other in a large group of participants, but also highly motivating for the main part of students. Subsequent to this motivational session, the amount of students showing up for the AMOS lecture remained stable for the next few weeks. An indicator for the positive effect of team building exercises is also the overcrowded lecture room. Some students had to sit down on a provisory ale-bench, even two to three weeks after the project kick-off, since the room capacity was limited.

The first instance of the AMOS project of 2010 did not involve explicit team building exercises at the project start. The first weeks of the AMOS project were used to become familiar with roles and practices of Scrum [5]. Team building activities were first described by the retrospective in the seventh sprint. The retrospective which was given by the guest speaker Linda Rising helped on the one side, to learn and speak about the problems that occurred during the development and created on the other side a pleasant team atmosphere. The positive outcome of this instructive and motivational retrospective is also reflected in Figure 19 (see Chapter 5.2.1), which shows an increased velocity for the sprint 7 and 8 subsequent to the retrospective.

Both AMOS projects validate the motivational effect of team building activities that can be part of the project kick-off or retrospective throughout the project.

H2: Small development teams show good performance

The number of members in a Scrum team can vary as it can be concluded from the comparison of academic projects that use the agile method Scrum (see Chapter 3.1). The analyzed projects show that development teams are composed of 4 up to 7 students [6], [28], [29].

Teams with too few or too many members are often inefficient [31]. Figure 20 (see Chapter 5.2.1) proves that teams with less than 4 members lack the manpower and knowledge to perform as good as teams with 4 to 7 members. After two developers left the AMOS project of 2010 during the semester, the team was reduced from 5 to 3 members. The same time the velocity dropped from 5 completed story points in the sixth sprint to 2 completed story points in the seventh sprint. Although that there were still more than 50% of the developers left, this trend describes a disproportional decrease of 60% in story points.

Teams with more than 7 members can require more time to coordinate tasks and tend to have a higher social loafing than smaller teams, argues Cohn [35]. The organization of meetings is for instance a greater coordination effort the more members are invited. The same time, members of large teams exert less effort if they rely on other TMs doing the job. Cohn defines small teams with 4 to 9 members and large teams with 14 to 18 members. He points out that a team with 5-7 people is presenting the best performance.

Regarding the experience which was gained in the AMOS project of 2010, the academic courses in comparison and the literature on the size of development teams, the hypothesis that small development teams show the best performance can be corroborated.

H3: Distributed development does not lower velocity

Distributed Scrum as it is attaining more and more attraction in the software industry comes with several benefits and challenges. The benefits of lower labor costs, talented employees and scalable projects are opposed to the challenges of communication, coordination and control issues [42], [43].

A distributed Scrum team can be beneficial for an academic course where skilled students are absent. Labor costs are on the other side a negligible factor, if distributed Scrum is considered in a university course. In the AMOS project of 2010, there have been 11 developers at the outset of the project. This number of developers was sufficient to build two teams. A year later, there were just 5 developers and one senior developer available. This trend runs the risk of too few TMs in future instances. A solution therefore can be described by the use of distributed Scrum and an offshore development team.

However, this initiative is only promising if the velocity is not being decreased in a global development team. A few case studies that surveyed several distributed Scrum projects came to the conclusion, that Scrum has to be tailored to fit the special requirements of a global project [44]. If Scrum is adapted in a suitable way to meet the conditions of a distributed environment, it might require additional practices such as Scrum of Scrums [45]. This practice connects distributed Scrum teams by a regular meeting of SMs across locations. Studies show that distributed Scrum teams achieve performance-wise the same velocity as collocated teams [42]. This result could also be reproduced in many cases [45].

The literature review validates the hypothesis of an equal velocity of collocated and distributed teams for industry projects. If the same outcome is true for distributed Scrum that is tailored for academic courses, has to be validated by further research.

6.2 Scrum methodology

The Scrum methodology, which is premised on the Agile Manifesto (see Chapter 2.2), allows for changes as long as the idea and principles of Scrum are not infringed. The practices, artifacts and roles can be customized in a way that they better meet the requirements of an academic setting. The hypotheses H4 to H7 propose changes to the Scrum methodology that are a result of the AMOS experiences and case study research.

H4: Diversifying user stories in the sprint planning reduces risk

The writing and prioritization of user stories is crucial to the success of a project. User stories constitute the features of the final product and require a careful planning to reduce the risk of not completed or partly implemented features. Since every user story contains a certain business value, it is helpful to assess these features according to the four categories of “must have”, “should have”, “could have” and “won’t have” [37].

Features described as “must have” highly contribute to the profit or loss of a project if they are implemented properly. “Should have” features, which are accompanied by a lower priority than “must have” features, are presenting an essential part of the final product. The features specified as “could have” provide merely optional functionality. User stories that are classified as “won’t have” express on the other side just a desired functionality that is not needed but could be a part of future releases [6].

Next to categorizing features, user stories can also be allocated according to component or feature teams [35]. Component teams are basically created to implement one component of the final product. These teams are in the manner of speaking responsible to deliver a single component independent from other parts of the software. This approach can result in a product, where components are not working in convergence with other components. Feature teams are on the other side focusing on one feature that is implemented across all components or layers. Latter approach tries to ensure functionality and reduce the probability of failure.

The strategy to diversify teams according to features was not exerted in the AMOS project of 2010. The Mydosis project consisted of two component teams. One team was responsible for the development of the Mydosis Web service. The other team was in charge of the Android application. In 2011, there was not this distinction made since only one team was available and working on the Web service. However, user stories were split in components in 2011 after the implementation of Spring security failed in the third sprint (see Chapter 4.2.4). User stories that were touching several components such as the social network, OSM or logbook, were introduced from the

fourth sprint on in order to avoid the dependency of features. The risk of a show-stopper such as Spring security could be avoided by this procedure.

The analysis of user stories diversified by component or feature teams does not reveal an ideal way. In consequence, the hypothesis of risk reduction by using a component or feature team cannot be confirmed. Further research and background information is needed to clarify this topic.

H5: Software tools facilitate the administration of artifacts

The analysis of Scrum projects in an academic setting has shown that there are basically two ways how Scrum artifacts can be managed (see Chapter 3.3.1). Scrum artifacts can be administrated manually or by the support of software tools.

The development process of Android applications at the Rochester Institute of Technology relied on traditional artifacts of Scrum [28]. Poster papers and index cards were used to illustrate user stories. These stories and broken down tasks were then pinned on the Product Backlog which was available on a Board. This Scrum board also contained the Sprint Backlog and Burndown chart.

A Web application that supports and enhances the use of Scrum was deployed in the agile project at the University of Ljubljana [6]. This software is available as the proprietary software “Agilo for Scrum” and the Open Source solution “Agilo for trac”. Both software solutions, which are customized to meet the requirements of Scrum, are offered with a bundle of features. A digital whiteboard, online Product Backlog, dashboard with Burndown chart and integrated Wiki are just a few features that are available by “Agilo for trac” [32]. All these features support the roles of SM, PO and TM in the Scrum process and offer in addition a valuable analysis tool.

A compromise that was used in the AMOS project of 2010 and 2011 is Google Docs. The spreadsheets of this Web service were edited to work as Product Backlog, Sprint Backlog and Feature Archive for the agile development project [39]. The usage of Scrum was to some extent facilitated by the online repository of Google. Nevertheless, there were reoccurring issues in illustrating user stories and copying

user stories from one spreadsheet to another. A more customized software and advanced tool support would have probably improved the handling of user stories during the AMOS projects.

Future AMOS projects should try to implement Open Source software like “Agilo for trac” or use proprietary software such as the “Team Foundation Server” of Microsoft that is available for free via a university license program. After implementing new software tools that support Scrum, they should be compared with Google Docs in terms of ease of use and administration of Scrum artifacts.

H6: Mailing lists work as daily scrum replacement

Daily Scrums are an important practice of the agile methodology Scrum. They are scheduled to take place every day during the Sprint execution and aim at updating and synchronizing everybody who is involved in the Scrum project [11].

The three questions of the Daily Scrum “What did you do since the last Scrum?”, “What are you doing until the next Scrum”? and “What is stopping you getting on with your work?” are basically analyzing the goals and impediments of the current sprint [15]. This meeting is supposed to take no longer than 15 minutes in order to briefly inform all participants on the project’s progress. It can also serve as a basis for further meetings if major issues are arising. A later meeting can then help to clarify these problems instead of discussing them in detail at the Daily Scrum.

Implementing Daily Scrum in an academic course can become a challenge. If there are only one or two days per week available where students can meet for the Daily Scrum, it is necessary to adjust the Daily Scrum to weekly or biweekly Scrum meetings. The university courses that aimed at developing Computer games and a Web service included as a consequence biweekly Scrum meetings [6], [29]. In 2011, there was only one day per week available during the AMOS project where all students could meet. This day, however, was used for the Sprint Review and planning of the next sprint. As there was no other time slot given and nothing related to a daily Scrum established in the AMOS project a year earlier, it was needful to improvise.

The mailing list which was initially used to exchange information between developers was now assigned a superior role. All the questions that should be asked during the Daily Scrum were now successfully communicated on the developers' mailing list (see Figure 10). A new "Daily Scrum" thread accounted in the end of the semester for more than 39 posts, after it was created halfway through the semester. It was on top also available to the public. That means, not only the developer could get an idea of the progress and impediments but also the SM, MPO, SPO and anyone else interested in the project.

The positive experience with the virtual Daily Scrum and the demand for such a practice, as it was asked for in the AMOS survey 2010 [5], will drive the implementation of this practice in further instances of the AMOS project. The replacement of the Daily Scrum by a mailing list is an important improvement that increases the communication between the developers and facilitates the problem solving process.

H7: Proactive customer involvement provides valuable feedback

Involving the customer as the stakeholder of a project in the Scrum process is also a key element of every agile method. The customer has to portray the end consumer, describe the domain model and draw up guidelines.

The customer can offer help by providing feedback or giving response to any question that concerns the look and feel of the final product. The domain is usually known best by the customer who should be contacted if there are domain-related questions. A close feedback-loop with the customer, who actively commits to the project, is important.

The customer of the AMOS project of 2010 was actively engaged in the development process of Mydosis. He was not only describing the domain model but also delivering data as a valuable input for the development and working side by side with the TMs to test the product [5].

The AMOS project of 2011 displayed a different situation. The customer in FSAhoy was not involved to the same extent as Dr. Bernitzki a year before. Although that the customer Markus Bärlocher was giving a motivating introductory speed (see Chapter 4.2.1), it was not possible to arrange a face-to-face interaction with him for a second time. Telephone conferences with the customer and weekly status updates per Email could not replace the physical presence of a customer who can proactively get involved in the Sprint Review. The feedback that was given by the customer per telephone and Email would have been more helpful, if it had been based on a working prototype and an on-site interaction with the customer.

To improve future AMOS projects, it is imperative to integrate and involve the customer proactively in a way that the customer knows about the progress and issues of a project. A face-to-face interaction between the customer and the Scrum team presents a valuable asset for any agile development project.

6.3 Velocity

The velocity is a significant metric that allows for tracking and estimating the progress of project (see Chapter 4.3.1). There are a lot of factors that can influence the velocity of a project or team. The hypotheses H8 to H10 are contemplating the effect of velocity in the AMOS projects and are describing what can be derived from literature review.

H8: Documenting planned story points optimizes estimation forecast

Forecasting the velocity of a development team can be crucial for any agile project. Knowing how many story points the team can achieve per sprint, makes it possible to foresee how many sprints will be required to release a certain product version. This planning figure is also important for describing and updating the project plan.

Completed story points are often the only metric that is used in order to forecast the future velocity by confidence intervals [35]. This approach to consider only completed user stories (see Chapter 4.3.2) originates from several reasons [37]. First of all, it is difficult to estimate the percentage of user stories that are partly completed. A user

story that is 80% complete could still account for more time than the remaining 20% of effort may suggest, if there is critical work left to do. Secondly, the forecast becomes inaccurate if decimal values are the basis for a calculation. User stories are already classified according to a relative measurement (see 4.3). If story points would be further broken down, it would be even more difficult to use them as a reliant metric. Thirdly, not fully implemented user stories do not present a business value to the customer. If user stories are not passing acceptance tests, they are not implemented as described by the customer and are not presenting the functionality as supposed to.

Planned story points can on the other hand also prove their value. A target-comparison analysis can be created by comparing planned and completed story points [37]. The difference in story points highlights possible impediments that were preventing the team to complete all assigned story points and shows if the team was able to complete more story points than expected. The planned value is, as described by Figure 16, also less prone to variations in story points.

What helps to predict the future velocity are planned and completed story points. Taking completed story points on its own will not make a valid forecast. The values of planned story points are more reliant as they provide a greater consistency after the first few sprints. The TMs are often able to complete the planned number of story points if there are no problems arising during the Sprint execution.

H9: Revised domain models negatively impact velocity

A domain model is the overarching vision of a project that describes the main components of a product. The domain model should be created by the customer along with the PO and TMs. It is the most important document for creating the Product Backlog and should not face too many changes.

After the domain model of FSAhoy was introduced at the outset of the AMOS project by the customer (see Chapter 4.2.1), there were only small adjustments made throughout the course in order to maintain the idea and concept of FSAhoy. The conference call with the customer in the seventh sprint was helpful to see which

components were already implemented and what areas require more attention (see Chapter 4.2.8). The focus of FSAhoy was, as a result of the telephone call, directed on the logbook as it is the main element in the domain model for the AMOS project of 2011. Incremental changes, which concerned user stories of the next sprints due to the feedback of the customer, did not show a remarkable change in velocity.

The domain model of Mydosis was not created before the actual development work started. It was, to be more precise, the job of a few TMs to work on the domain model during the semester-long course. This approach and new requirements led to a lot of changes in the domain model. The impact of these changes was greater than those of the AMOS project one year later. In end, the domain model turned out to be the main cause of frustration throughout the AMOS project of 2010 [5]. Students were claiming in the AMOS survey 2010 that they had to work with a poor domain model which was only created by a few students and not the entire team. Asking the question if the domain should be agreed beforehand, 50% of the TMs agreed. The consequences of the changes in the Mydosis domain model are also reflected in Figure 19 that draws the ups and downs of completed story points. The drop from 11 story points in the fifth sprint to 5 story points in the sixth sprint was due to the revision of the domain model and the consequent leaving of two students (see Chapter 5.2.1).

Having said all this, it can be concluded that the domain model should be created jointly by the customer and POs before the development starts. It can be defined as well in collaboration with the TMs. Frequent revisions of the domain that negatively impact the velocity of a development project should to be avoided in any case.

H10: Changes in the team size decrease velocity

The composition of development teams can change in agile projects as well as in any other software project. There are TMs that are leaving or joining a team. Both events the adding and quitting of TMs are decreasing the development speed at first.

At the beginning of each AMOS course has to be made a decision on the Scrum roles for students. Since there is every year a different cohort of students

participating in the agile project, it is difficult to foresee how many students will be assigned the role of TM or SPO. As it is not easy to foresee the amount of students that are taking part in the AMOS project for every year, a decision on the Scrum roles has to be taken spontaneously.

In 2010, there were 11 developing students involved in Mydosis whereas FSAhoy had to get along with 5 developers and one senior developer. A team that consists of more than 7 members should be broken down into smaller teams with 4 to 7 members as it was described by the second hypothesis in Chapter 6.1. This procedure was not necessary for FSAhoy but for Mydosis. The two teams of Mydosis were composed of 5 and 6 members each to ensure a high productivity. However, after two students left team two in the seventh sprint, both teams were merged to a team consisting of 9 members. This change in the team structure was also described by Table 9. Contrary to what was expected and described by literature, the velocity per developer even increased from 1.16 in the first few sprints to 2.06 after two members left the Mydosis project in the seventh sprint. The velocity is according to Mike Cohn more likely to drop when the team size changes [35]. An increased communication can be necessary to get new TMs productive. This phenomenon could not be observed for the Mydosis project. However, the situation was slightly different for Mydosis as there were no new TMs added but TMs, which already knew each other, merged to one team.

A change in the team size can affect the velocity. But it does not necessarily need to decrease the velocity. It depends on the TMs and the structure of the development process if a new TM or merged team will have a lower performance at first or not. Further research on this topic is necessary to make a clear statement, how the adding or merging of TMs and teams will influence an agile project.

7 Conclusions

This paper demonstrates how the AMOS project can be improved in future instances. By analyzing the AMOS project of 2011 and comparing it with Mydosis, the previous AMOS project at the University of Erlangen-Nürnberg and similar projects in an academic setting, several ideas for improvement have been generated. These suggestions were framed as hypotheses and evaluated for validity.

The research approach of this thesis has some limitations as it is based on a post-facto analysis and case study research. These restrictions are outlined in the following Chapter 7.1. Nevertheless, the lessons learned that could be drawn from the analysis are especially valuable for further AMOS projects or academic courses that follow a comparable course design. The recommendations that could be validated are summarized in Chapter 7.2. A few suggestions for how to improve the AMOS lab course could not be confirmed. Further research is necessary to validate those changes related to the AMOS project as it is described in Chapter 7.3.

7.1 Limitations

There are limitations for this thesis that must be mentioned when evaluating the outcome of this research. A post-facto analysis and confirmatory case studies are the scientific methods that were used for this empirical research. Both, the methods and specific context of this study limit the validity and representativeness of this work.

The post-facto analysis is built on the analysis of the diploma thesis on the AMOS project of 2010. Markus Stipp, the author of that thesis was assigned the role of a Product Owner Proxy and carried out two surveys throughout the development course to evaluate it and provide recommendations for future AMOS courses. The results and interpretations of this approach to the survey formed the basis of a descriptive research used to analyze the AMOS project of 2010. However, the findings of this research only served as part of a comparison since the survey design was already predefined and two courses were differing in many aspects.

Confirmatory case studies, involved in part of the comparison of Scrum in academic courses, can help to validate or invalidate theories. Multiple cases across different course designs and development projects were used in this thesis. Most of these case studies followed a structured survey approach to evaluate student opinion and the use of agile methods in an academic course. Since the projects each applied Scrum differently, it was not always possible to illustrate or confirm theories. The observations from these case studies are also biased by external validity as some of the projects lacked sufficient data to portray the case findings to other academic projects.

The outcome of this research mainly contributes to the improvement of the AMOS lab course in future instances. The reason being is that the AMOS project of 2010 and 2011 are unique in character and driven by a lot of influencing factors. Hence, suggested improvements are most likely to work for AMOS projects. Given these underlying circumstances, it is difficult to generalize, that the findings will fit into any academic course that is about to apply Scrum. Also, the representative of this study cannot be entirely proven since only two AMOS projects were the foundation of this analysis. More AMOS courses are required to make a final statement on how Scrum can be tailored to be the best fit in this setting.

7.2 Recommendations

The following recommendations are made to improve the AMOS project in the future. The lessons learned which are obtained from this thesis, are summarized below and validated by the comparison between the AMOS projects of 2010 and 2011 and literature review on similar courses. The suggested improvements to the AMOS course embrace change within the agile methods of Scrum and course design.

- **Team building exercises are motivating.** The project kick-off and the first weeks of a project are crucial to set the stage and motivate participants. In the AMOS project of 2011, there was a ball game played and a pipeline constructed with the intent to get to know each other. These icebreaker activities not only helped to familiarize with names and hobbies, but also motivated students.

- **Small development teams show good performance.** Scrum relies on small development teams that are composed of 4 up to 7 students. Teams that are smaller or larger in size are often not efficient due to a lack of knowledge or increased communication costs. Productive performance can only be delivered by teams that are composed of 4 to 7 students.
- **Mailing lists work as Daily Scrum replacement.** The practice of Daily Scrum can be difficult to implement in an academic setting. There are possibilities to retain Daily Scrums even if the physical attendance of TMs is not present. A mailing list can be deployed to replace the Daily Scrum by a virtual tool.
- **Proactive customer involvement provides valuable feedback.** A proactive customer plays an important role in Scrum by defining the domain model of the project in collaboration with the PO. The customer has also to be informed about the development progress. It is important to involve the customer in such a way that feedback on the development of the project can be provided.
- **Documenting planned story points optimizes estimation forecast.** Story points are a valuable planning figure used to track and forecast the progress of a project. Planned and completed story points are often two different things that both serve as indicators of velocity. The documentation of planned story points supports the target-performance analysis and helps to predict future velocity.
- **Revised domain models negatively impact velocity.** A domain model should be defined at the outset of a development project that is using Scrum. Revisions to the domain model can be frustrating for the development team, if they are recurring. Revised versions of the domain model in this consequence also negatively impact the velocity of a team.

7.3 Future research

Not all of the suggested improvements could be confirmed by the time this thesis was written. Further research is necessary to obtain more data that can be used to validate some of the suggestions that were lacking sufficient information. Future AMOS projects should therefore apply the proposed recommendations, analyze and evaluate them in order to make a qualified statement on the suggested improvements.

In particular, the diversification of user stories and changes in the team size require more data for validation. Diversifying user stories through feature teams instead of component teams should result in a greater functionality and lower probability of failure. This hypothesis could not be corroborated since there were only component teams available in the AMOS project of 2010. The impact on the velocity of a project from changes in the team size could, just as the diversification of user stories, not be demonstrated clearly. The merging of teams and thereby adding of new TMs revealed a positive effect in the AMOS project of 2011 even though that some case studies professed, that changes in team size are always accompanied by a decline in velocity.

Next steps for future AMOS projects could be the analysis of existing software tools that support the development methodology Scrum and the consideration of Scrum in a distributed setting. There are a few Open Source and proprietary software solutions available that are tailored to meet the requirements of the Scrum development process. These tools could be beneficial and facilitate the administration and documentation of artifacts in Scrum, if they are easy to understand and use. A distributed development team could also be part of a future instance of the AMOS project. If there are too few developers enrolling in the AMOS course, one possible scenario would be to work with an offshore development team. This approach may present a lot of challenges, but the reviewed case studies also indicate that this does not necessarily have to lower the velocity of a project.

References

- [1] D. Riehle. (2010, Dec.) Dirk Riehle's blog about everything computer science, applied and more. [Online]. <http://dirkriehle.com/2010/12/06/das-amos-projektkonzept-2011/>, last access: 20.03.2012.
- [2] Forrester Research, "The Forrester Wave™: Agile Development Management Tools," 2010.
- [3] S. Ramakrishnan, "Innovation and Scaling up Agile Software Engineering Projects," *Issues in Informing Science and Information Technology*, no. 6, pp. 557-575, 2009.
- [4] D. A. Umphress, T. D. Hendrix, and J. H. Cross, "Software Process in the Classroom: The Capstone Project Experience," *IEEE Software*, vol. 19, no. 5, pp. 78-85, Sep. 2002.
- [5] M. Stipp, "Agile Methods and Open Source Software Development in Student Projects," Diplomarbeit, 2010.
- [6] V. Mahnic, "A Capstone Course on Agile Software Development Using Scrum," *IEEE Transactions on Education*, vol. 55, no. 1, pp. 99-106, Feb. 2012.
- [7] V. Devedžić and S. R. Milenković, "Teaching Agile Software Development: A Case Study," *IEEE Transactions on Education*, vol. 54, no. 2, pp. 273-278, May 2011.
- [8] D. F. Rico and H. H. Sayani, "Use of Agile Methods in Software Engineering Education," *Agile Conference*, pp. 174-179, 2009.
- [9] L. Pries-Heje and J. Pries-Heje, "Why Scrum works: A case study from an agile distributed project in Denmark and India," in *Agile Conference*, 2011, pp. 20-28.
- [10] W. W. Royce, "Managing the Development of Large Software System," *Proceedings of IEEE WESCON*, vol. 26, pp. 1-9, Aug. 1970.
- [11] D. Riehle, "Agile Methoden und Open Source," Friedrich-Alexander University of Erlangen-Nuremberg Lecture, 2011.
- [12] J. O. Clark, "System of Systems Engineering and Family of Systems Engineering From a Standards, V-Model, and Dual-V Model Perspective," *IEEE International*

- Conference on System of Systems Engineering*, pp. 381-387, Apr. 2009.
- [13] M. Kuhrmann, D. Niebuhr, and A. Rausch, "Application of the V-Modell XT – Report from a Pilot Project," in *Unifying the Software Process Spectrum*, L. J. Osterweil, Ed. Berlin Heidelberg NewYork, Deutschland: Springer Verlag, 2005, pp. 463-473.
- [14] K. Beck. (2001) Manifesto for Agile Software Development. [Online]. <http://agilemanifesto.org/>, last access: 15.01.2012.
- [15] F. H. Cervone, "Understanding agile project management methods using Scrum," *OCLC Systems & Services*, vol. 27, no. 1, pp. 18-22, Nov. 2011.
- [16] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd ed. Amsterdam, Netherlands: Addison-Wesley Longma, 2004.
- [17] S. Warden, *Extreme Programming Pocket Guide*, 1st ed., T. A. Diaz, Ed. Sebastopol, CA, USA: O'Reilly Media, 2003.
- [18] L. Lindstrom and R. Jeffries, "Extreme Programming and Agile Software Development Methodologies," *Information Systems Management*, vol. 21, no. 3, pp. 41-52, 2004.
- [19] S. Smith and S. Stoecklin, "What we can learn from extreme programming," *Journal of Computing Sciences in Colleges*, vol. 17, no. 2, pp. 144-151, Dec. 2001.
- [20] G. Vanderburg, "A Simple Model of Agile Software Processes - or - Extreme Programming Annealed," *Proceedings of the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA '05)*, vol. 40, no. 10, pp. 539-545, Oct. 2005.
- [21] B. Hindel, K. Hörmann, M. Müller, and J. Schmied, *Basiswissen Software-Projektmanagement*, 3üth ed., H. Heilmann, Ed. Heidelberg, Deutschland: dpunkt Verlag, 2009.
- [22] C. Keith, *Agile Game development with Scrum*. Amsterdam, Netherlands: Addison-Wesley Longman, 2010.
- [23] K. Schwaber, *Agile Project Management with Scrum*, K. Atkins, Ed. Redmond, Washington, USA: Microsoft Press, 2004.
- [24] O. S. Initiative. Open Source Initiative. [Online].

- <http://www.opensource.org/docs/osd>, last access: 23.01.2012.
- [25] D. Riehle, et al., "Open Collaboration within Corporations Using Software Forges," *IEEE Software*, vol. 26, no. 2, pp. 52-58, Apr. Thomas .
- [26] W. Scacchi, "Free and Open Source Development Practices in the Game Community," *IEEE Software*, vol. 21, no. 1, pp. 59-66, Jan. 2004.
- [27] B. Kogut and A. Metiu, "Open-Source Software Development and Distributed Innovation," *Oxford Review of Economic Policy*, vol. 17, no. 2, pp. 248-264, 2001.
- [28] T. Reichlmayr, "Working Towards the Student Scrum – Developing Agile Android Applications," in *Proceedings of the 118th ASEE Annual Conference & Exposition 2011*, Vancouver, 2011.
- [29] J. Schild, R. Walter, and M. Masuch, "ABC-Sprints: Adapting Scrum to Academic Game Development Courses," in *Proceedings of the Fifth International Conference on the Foundations of Digital Games (FDG '10)*, New York, 2010, pp. 187-194.
- [30] D. Riehle. (2010, Sep.) Dirk Riehle's blog about everything computer science, applied and more. [Online]. <http://dirkriehle.com/2010/09/11/the-2010-amos-project-from-osr-group/>, last access: 27.01.2012.
- [31] U. Wolz and S. M. Pulimood, "An integrated approach to project management through classic CS III and video game development," in *Proceedings of the 38th SIGCSE technical symposium on Computer science education (SIGCSE '07)*, New York, 2007, pp. 322-326.
- [32] Agilo Software GmbH. (2011) Agilo for trac - Feature List. [Online]. <http://agilofortrac.com/en/features/>, last access: 25.02.2012.
- [33] J. Sutherland. (2010, Apr.) Scrum Log. [Online]. <http://scrum.jeffsutherland.com/2010/04/story-points-why-are-they-better-than.html>, last access: 20.02.2012.
- [34] M. Cohn, *Agile Estimating and Planning*, first edition ed. Upper Saddle River, NJ, USA: Prentice Hall Professional Technical Reference, 2005.
- [35] M. Cohn, *Succeeding with Agile: Software Development Using Scrum*, first edition ed. Amsterdam, the Netherlands: Addison-Wesley Professional, 2009.

- [36] R. Pichler, *Agile Product Management with Scrum: Creating Products that Customers Love*, first edition ed. Amsterdam, The Netherlands: Addison-Wesley Professional, 2010.
- [37] M. Cohn, *User Stories - für die agile Software-Entwicklung mit Scrum, XP u.a.*, 1st ed., Heidelberg, Ed. München, Deutschland: mitp, Verlagsgruppe Hüthig Jehle Rehm GmbH, 2010.
- [38] ohloh. (2012, Feb.) ohloh - Committed to Code. [Online]. http://www.ohloh.net/p/compare?project_0=Dosis&project_1=Free+Seas+Ahoy!, last access: 02.03.2012.
- [39] D. Riehle. (2012, Feb.) Google Docs - FAU AMOS Schedule. [Online]. <http://goo.gl/BZpU8>, last access: 05.03.2012.
- [40] R. Wirdemann, *Scrum mit User Stories*, 1st ed., M. Sommer, Ed. München, Deutschland: Carl Hanser Verlag, 2009.
- [41] M. Fowler. (2006, May) Martin Fowler. [Online]. <http://www.martinfowler.com/articles/continuousIntegration.html>, last access: 15.02.2012.
- [42] J. Sutherland, G. Schoonheim, N. Kumar, V. Pandey, and S. Vishal, "Fully Distributed Scrum: Linear Scalability of Production between San Francisco and India," in *2009 Agile Conference*, Chicago, Aug. 2009, pp. 277-282.
- [43] M. Paasivaara, S. Durasiewicz, and C. Lassenius, "Using Scrum in Distributed Agile Development: A Multiple Case Study," in *2009 Fourth IEEE International Conference on Global Software Engineering*, Limerick, 2009, pp. 195-204.
- [44] E. Hossain, P. L. Bannerman, and R. Jeffery, "Towards an understanding of tailoring scrum in global software development: a multi-case study," in *Proceedings of the 2011 International Conference on Software and Systems Process (ICSSP '11)*, New York, 2011, pp. 110-119.
- [45] J. Sutherland, G. Schoonheim, and M. Rijk, "Fully Distributed Scrum: Replicating Local Productivity and Quality with Offshore Teams," in *Proceedings of the 42nd Hawaii International Conference on System Sciences (ICSS '09)*, Big Island, 2009, pp. 1-8.