Restructuring of System Analysis and Design Course with Agile Approach for Computer Engineering/Programming Departments

Ahmet ALBAYRAK
Department of Computer Programming
Karadeniz Technical University
Trabzon, 61030, Turkey

***Abstract*—Today software plays an increasingly important and central role in every aspect of everyday life. The number, size, complexity and application areas of the programs developed continue to grow. Many software products have serious problems in cost, timing and quality. It has become almost normal for software projects to exceed their planned cost and schedule. A significant number of development projects have never been completed and many of them have not met user requirements. Employers are not satisfied with new graduates for a variety of reasons (They do not know how to communicate. They do not have enough experience and preparation to work as a team member. They do not have the ability to manage their individual works efficiently and productively. This work was reconfigured and conducted with Scrum from the Agile methodologies of the System Analysis and Design course, which is especially taught in Vocational Schools and Engineering faculties. The recommended approach is available for software development departments. The suggested approach is applied in System Analysis and Design course.**

Keywords—Agile software development; scrum; course re-design; computer science education

I. INTRODUCTION

In this study, it was explained to reconstruct the System Analysis and Design course with Scrum which is one of the Agile technologies for the daily adaptation of the course in order to take the project course in the Higher Schools/Engineering Faculties. It is available for parts of computer. Students want relevant assignments/projects that are engaging, creative and prepare them for their careers. Finishing and research projects are realistic and beneficial for students. It also emphasizes the balance between product and process [1], [2]. Trainees recommend real industry projects to prepare students for industry and improve their implementation skills. It is evident that teaching develops learning by supporting with familiar, concrete and related examples. According to Shaw and Dermoudy, competition tends to create higher academic achievement. Competition in engineering team-based is a way of bringing students' achievements to the top while giving "weak" students confidence and motivation [3].

There are thousands of academic programs in the field of computers in the world. These programs give tens of thousands of computer specialist graduates per year. Less than 15% of them are undergraduate education. What remains is the software industry. These programs represent a variety of disciplines (including computer science, computer engineering, information systems and software engineering), and are in various academic units (engineering, research institutes and vocational colleges). Employers complain about new graduates for a variety of reasons (I do not know how to communicate, I do not have enough experience and preparation to work as a team member, I lack the ability to manage my own individual works efficiently and productively, organizational structures are insufficient and business practices are inadequate [4].

Because of the increasing use of agile methods in recent years, teaching agile methods for software development has become an important issue [5]. Although some universities have begun to teach lectures on agile methods, they are still quite new [6]. The course "System Analysis and Design" is a suitable course for starting the development of the Agile software because the basic engineering courses mainly teach the traditional plan-oriented approach.

Agile surveys conducted by VersionOne since 2006 show that Scrum is the most common Agile method and that its share is constantly increasing. One of Scrum's key features is that each sprint consists of repeats called sprints, which finish with a subset of the final product properties. The application of agile methodologies has resulted in impressive results in terms of team performance, product quality and customer satisfaction.

Agile development methods have roots in design and development that are based on repeating, increasingly decades before [2]. The 1970s and 1980s included various forms of agile development with modern methods emerging in the 1990s. For example, XP (eXtreme Programming) began in 1996 at Chrysler Corp. During this time, lightweight methods such as Scrum have gained popularity.

The Scrum Guide describes Scrum as "a framework in which people can solve complex adaptive problems while producing and delivering products with the highest possible value" [3]. Scrum consists of a series of short repetitions called sprints. Each sprint simultaneously performs analysis + design + code + test and ends with an output of a subset of a final product specification. This is the final result, unlike the traditional waterfall method, that results in a long linear progression of requirements, design, code and test. Scrum methods are relatively simple and provide greater visibility,

predictability and flexibility. It also encourages high-performance teams. The user supports prioritization of stories/scenarios and job forecasting with story scores. Each sprint has a target that focuses on the punch [4], [5].

CS 4911 is a finishing course organized at Georgia Tech as a sprint throughout the semester [7]. The Rochester Institute of Technology has a computer engineering technology graduation course that uses an agile-like methodology for embedded systems design. Project management emphasizes learning outcomes related to product idea, entrepreneurship and professional skills [8]. [9]. Slovenia Ljubljana University taught and implemented Scrum in a software engineering graduation course [10]. The Royal Swedish Institute (KTH) studied Scrum's integration in mechatronic finishing projects. Their assessment has focused on individual student responsibility and initiative taking [11], [12]. Vicentin applied a two-part survey to assess the impact of Scrum's possible use in a Software Office. Based on the answers, Vincentin Scrum was the result of what could be a good vehicle for managing the squadrons. In Wagh's work [13], Scrum was implemented to provide students with a practical view of the software development industry. In the class project proposed by Wagh, the Scrum application evaluation was carried out through a questionnaire answered anonymously by the students. The aim was to understand how much the students learned and influenced this approach. As a result, many students saw how the project was managed and developed teamwork using Scrum. Questions about the evaluation of the works and the Scrum meetings showed that the day-to-day meetings were the most popular technique in this group and that they needed to evaluate the team calendar, sprint planning, user stories.

In our curriculum, system analysis and design is a tool that allows students groups to develop software in a project course using an Agile (Scrum based) method. One of the skills required to demonstrate this is system design. System design also includes documenting the project. We are designing and introducing an approach to improve the course. In this article we present the relevant approach. This article contributes to the literature by explaining how to reconstruct the courses whose content should be updated according to the conditions of the day.

II. "System Analysis and Design" Course

System Analysis and Design course is a profession which is seen within the scope of completion project in Engineering/Institute/Vocational Schools. This course is mostly designed and handled as a process management. The shortest possible definition of system analysis and design lesson is the act of converting a system into an information system. This conversion action is not limited to software. System analysis and design are provided to meet everything needed for a system such as software, hardware, suitable human resources, appropriate physical space and environment. The system is in the center of all activities. Among the expected achievements of the course:

- Identify the types of computer-based systems that the system analyst needs to determine and fulfill its core roles.
- To plan and schedule a project by its activities.
- To design and manage effective questionnaires.
- Create data dictionary entries for the logical and physical elements of the data processing, storage, streams and running systems, based on the data flow diagrams.
- Build databases for information systems.
- Designing output tables and graphics with input screens for information systems users.

These gains are not enough for the software development departments. Today, software development processes are diversified. The course consists of 4-hour sessions per week. At the end of the semester, there will be project presentations of the students, with or without a midterm or final exam. The content of this lesson, which usually lasts 12-14 weeks:

- Introducing systems, roles and development methods
- Understanding and modeling organizational systems
- Project management
- Knowledge acquisition and prototyping
- Use of data flow diagrams
- Analyze systems using data dictionaries, explain job descriptions and structured decisions
- Designing effective output and input
- Designing databases
- Object-oriented system analysis and design using UML (Unified Modeling Language)
- Successful implementation of information systems
- Group Project Presentations

Looking at the course content, especially the group interactions are lacking. It is not possible to measure features, such as working together, sharing information, taking responsibility. It is also not possible to develop these features. In this course, the teacher is a consultant. The teacher examines the work of each group during the weekly lecture hours and has recommendations. However, the work part of the group and its work do not interfere. This makes it hard to guess who finished the project and how much it contributed. In addition, there is no customer as a product owner in the current course content. The absence of the client removes the developed project from the real world and affects the motivation of students negatively.

In this study, System Analysis and Design course is reorganized as content and operation to be used especially in computer sections (software integrated projects). The reason for this configuration is as follows:

- Existing System Analysis and Design course is not enough for today's computer departments. In particular, the course should be conducted in an observational manner.

- Students who develop computer (software) based projects need to develop projects on more realistic conditions.
- Software development alone is no longer possible with the development of software technologies. It is necessary to develop software as a team. It is necessary to manage alignment, information sharing, and task distribution within the team.
- From the software development methodologies, Scrum is the most appropriate method for managing the course.
- Documenting the developed software in succession with success stories and reports and transferring it to the next students so that a software literature can be obtained.

III. Course Design

System Analysis and Design is an experience for students in the fields of computer science and information technology, where students combine the skills they learn and apply in an important team project. In order to provide a realistic and motivating experience, the course is structured with Agile contacts, exact dates, genuine customers and competitions. We believe that these elements contribute to the success of the course. Scrum's block diagram of the System Analysis and Design course is given in Fig. 1. Sprints are given with start and end times in the time tables. A new sprint begins immediately with the end of the previous sprint.

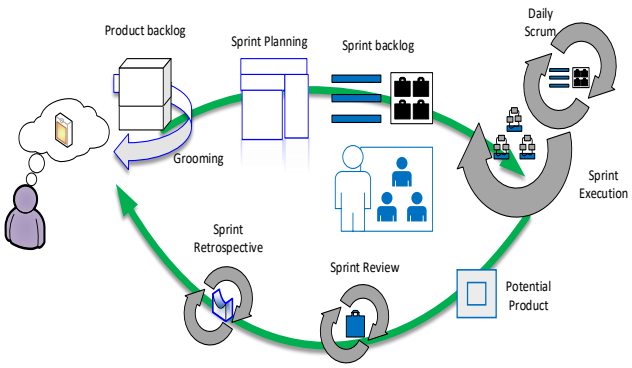

Fig. 1. Scrum process for system analysis and design course.

Table 1 gives the Scrum Sprints used in the course planning. Typically, a SCRUM project requires several iterations or sprints to improve the full set of stories in product accumulation. During the development process, a series of meetings or workshops are organized to configure the entire sprint kit together with the product increment and the product to be developed. These are mainly planning sessions. The Planning sessions development team meets at the beginning of each sprint to plan the work to be developed in future sprint. To do this, the scrum master picks stories from the product's accumulation, and the entire development team analyzes the story and possibly gets a consensus about when it might have been assigned to that story.

TABLE I. Curriculum

Sprint 0	Week 1-3	Lessons on Scrum and user stories Presentation of initial product accumulation, prediction of user stories, development of release plan.
Sprint 1	Week 4	Sprint planning meetings Beginning sprint buildup development
	Week 5-6	Project work. Daily Scrum meetings. Do not take care of Sprint accumulation.
	Week 7	Sprint review meetings. Sprint retrospective meetings.
Sprint 2	Week 8	Making Sprint planning meetings. Beginning sprint buildup development.
	Week 9	Project work. Daily Scrum meetings. Do not take care of Sprint accumulation. Sprint review meetings. Sprint retrospective meetings.
Sprint 3	Week 10	Making Sprint planning meetings. Beginning sprint buildup development
	Week 11-12	Project work. Daily Scrum meetings. Do not take care of Sprint accumulation.
	Week 13-14	Sprint review meetings. Sprint retrospective meetings.
Release		Presentation of observational study results. Present your desired functionality in the project.

The planning process continues until product size reaches the sprue size of the selected set of traces. These stories are the main result of the sprint spool planning session and configure the job during the next sprint. The second session is development. A set of stories to be applied to each developer is assigned and studied. This sprint is generally defined as "30 days" a month. Repetition in this study varies depending on the structure of the product produced. Another session is the daily Scrum sessions. The development team analyzes how many hours of work each day should be working if there are any deviations from the planned one as soon as possible. The time or technical difficulties attributed to each story are determined. This iteration is a sprint cycle that can be labeled "24 hours - 12 hours". The Scrum master notes the amount of time spent and compares it to the amount of time ahead and the expected duration of the sprint. It keeps a graph showing the duration if the last session type is a meeting ending session. After the Sprint is completed, the development team tries to analyze what it does well (to maintain it), analyze what has not been done, and how to improve it so as to follow a retrospective session. After these last negotiations and when the project is not completed, a new sprint will start with a new planning session.

To facilitate project management, each team has a spreadsheet loaded into the special Google Drive/OneDrive folder, which allows students to aggregate basic information and automate all reports and dashboards. The product, which will be developed in the project by looking at the roles and product components in Scrum, consists of laboratory tasks for each team whose main functional characteristics are defined by the content introduced by the teacher in the content.

There are several interpretations of roles in Scrum [14] [15]. However, in this study, product-oriented teachers and

development teams were identified as students. Development teams are encouraged throughout the course to encourage better implementation of mutual knowledge and methodology. Scrum master is one of the students. The role of the Scrum master is decided internally by the team, as it turns between the team members. Thus, the leadership qualities of each member in the team are improved.

Students can develop their homework (in the lab) or weekly (homework) in order to fulfill the estimated development time. Then, at the beginning of each laboratory session, the teacher talks up to five or ten minutes with each group in the daily scrum session and is guided by the scrum master (leading student) and reviews all the work done up to that date. The Scrum master, the leader of the team, records the "clear development time" and "the actual development time" values of "who" and "which story" he did.

Lesson planning consists of four sprints. There is a preparation Sprint (Sprint 0), three development Sprints (Sprint 1, 2, 3) and a release step that includes a finisher. Sprint lasts for 0, 5 to 20 days (1 or 3 weeks). Sprint 0 is a preparation sprint for students to meet with their customers (teachers) to build their product portfolios. Sprint 0 provides students with enough time to talk to their customers about their offers and decide what can be done in the course schedule on a timely basis. Teams should try to develop high-level, concise, and clear user stories to show all the expected features in the software. The relative amount of time per user story is estimated. It is better to start Sprints (1, 2 and 3) on Monday. On the first day of Sprint, the groups realized the Sprint planning meetings. It took 45 minutes to complete the event. Like many similar events, meetings were held on time, so that the class was used efficiently. Planning is primarily used in the development of Sprint Backlog, where the highest priority user stories are received and decided as a team to be added to Sprint Backlog. The Product Owner (teacher) brings top-level user stories to the Sprint Planning Meeting. The team decides together which stories to handle. At the end of this sprint, all resources related to the project are graded. The operation of the Sprint evaluation process in Scrum is given in Fig. 2. According to Fig. 2, the initial-desired product in this work is defined as Initial Input. The final output represents the final product. With the Sprints constantly recurrent, the process evolves towards the final product through product control and feedback.

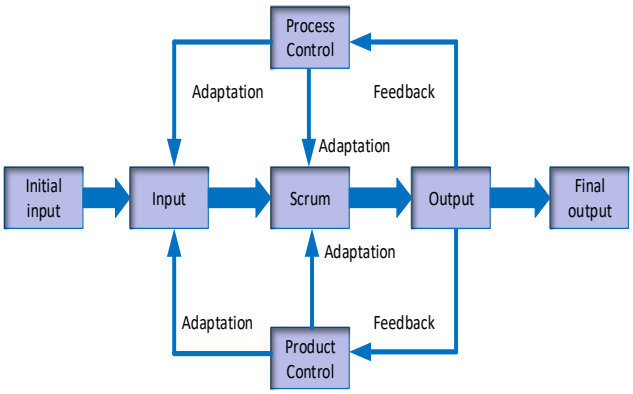

Fig. 2. Sprint processes.

The end of each sprint is designated as Friday. First, each group has to present the status of its projects to the whole class. The groups are obliged to meet with the client (teacher) to talk about progress and present their progress within a few days. Then groups are given 10 minutes to complete the Sprint Retrospective. The release includes the presentation of the results of the observational study of the project. It is also at this stage that the desired functionality is presented in the project.

Following the open questionnaire filled by 28 students before the project presentations, the main results are as follows:

- 92% of the students liked the lab sessions.
- Overall, 84% of the students liked the course to be handled by Scrum.
- 79% of the students liked the evaluation method.
- 86% of students liked SCRUM.
- 72% of the students liked to work in teams.

Also, 3 of the students did not succeed in the class and left it in the middle of the class. The remaining 28 students (89.2%) completed the course successfully. 18 (72%) of the students who completed the course got a score of 8 out of 10. Four of the students (16%) were between 7 and 5. The remaining 3 students (12%) were between 5 and 4 (including four).

The results in Table 2 clearly show that the course fully fulfills its overall objectives. The proposed approach is influenced by new learning (Question 2) during software development.

TABLE II. SOME QUESTIONNAIRE SURVEYS

Survey results 2014-2015/2015-2016 (N=28)			
Questions		**Median**	**p-value**
1	Benefits of agile software development methodology	4	0.081
2	Contribution of method to new learning	5	0.024
3	Contribution of method to software project development and management	4	0.92
4	Contribution of method to group work in software development process	5	0.85
5	Contribution of method to group communication in software development process	4	0.067

IV. Results and Discussion

The new course content implemented as a semester in 2014-2015 has increased the success. The questionnaire in Table 2 showed that the opinions of the students were overwhelmingly positive (see Table 2). All students found the course useful (92%). Scrum itself is simple, but its implementation is difficult. For this reason, there is no need for extensive formal courses to promote the concepts. Rather, projects should be preferred as means for bringing together all the different issues that affect implementation and for spreading them against each other. Nevertheless, when compared to traditional plan-based software development, students must have sufficient background in the philosophy and techniques of software engineering and information systems development to accurately assess the benefits and deficiencies of the agile approach.

In addition to teaching everyone involved in the project, the ScrumMaster Scrum should ensure that everyone follows Scrum's rules and practices. It is especially important to prevent the Product Owner from interfering with team management, redefining the scope or objectives of a Sprint, or adding new requirements when a Sprint starts.

In this study, both the product owner (customer) and the teacher in the position of consultant reduced the performance of the students. Students do not want to see the same teacher in two different roles, and the real customer wants to focus. For this purpose, this consultant has come to the conclusion that a consultant teacher is an assistant but also a teacher in the customer role.

The study is proposed to identify, discuss and quantify how Scrum can become flexible and collaborative in software development teaching. To assess the impact of this approach, a survey was conducted among students who completed the course at the end of 2014/15. Given the perspective of the pupils, the approach allows students to gain experience. The adaptations that students make when they consider the flexibility, communication and release of the functional parts of the system are important.

The System Analysis and Design course is often project-based and can be easily designed to meet the needs of others. Such needs are also available at the university level, as new technologies are constantly being sought to support faculty, research, teaching and services throughout the university. This course also enhances intra-departmental collaboration opportunities where the skills and efforts of computer science students focus on needs. This model provides students with authentic learning opportunities that they have been working on in college and real-world projects that the community has benefited from.

The grading of the note was obtained at the end of each Sprint as a result of observations of the Scrum master and the product owner (teacher). Table 3 shows the notes the groups received. Here it appears that the grades are generally increased with each Sprint. The group with the highest rating is Group 7. The overall increase in grades indicates that students are used to Scrum.

TABLE III. Group's Notes During the Project Development Process

Sprint name	Group name	Grade	Total Grade
Sprint 0	Group 1	60	485
	Group 2	70	
	Group 3	75	
	Group 4	65	
	Group 5	65	
	Group 6	70	
	Group 7	80	
Sprint 1	Group 1	70	500
	Group 2	75	
	Group 3	60	
	Group 4	70	
	Group 5	70	
	Group 6	75	
	Group 7	80	
Sprint 2	Group 1	75	548
	Group 2	75	
	Group 3	80	
	Group 4	75	
	Group 5	80	
	Group 6	80	
	Group 7	83	
Sprint 3	Group 1	70	538
	Group 2	75	
	Group 3	76	
	Group 4	80	
	Group 5	77	
	Group 6	80	
	Group 7	80	
Release	Group 1	75	570
	Group 2	78	
	Group 3	80	
	Group 4	82	
	Group 5	80	
	Group 6	85	
	Group 7	90	

The last line in Table 3 shows the final grades received by the release groups. Comparison of these notes with the past three-year average is given in Fig. 3. Group 6 and Group 7 do not have notes in 2011 and 2012. Since the number of students was not sufficient, these groups were not formed in the relevant years.

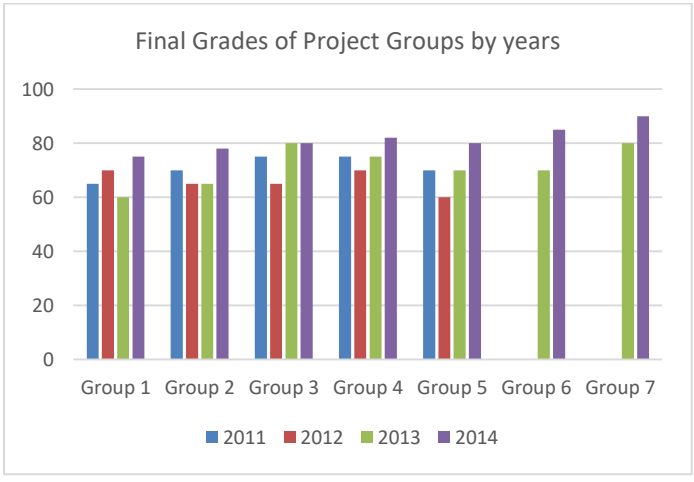

Fig. 3. Comparison of grades by years.

According to Fig. 3, the average of the final grades of the last three years is (2011, 2012, and 2013) lower than the year 2014 when the course was processed with Scrum. These results show that if the course is processed with Scrum, it will be more efficient. Not only the notes, but also the self-confidence that students gain is influencing the performance of the proposed approach positively.

REFERENCES

[1] K. Z. Watkin, T. Barnes, "Competitive and Agile Software Engineering Education",IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the, Concord, NC, USA, 2010.

[2] J. Polack-Wahl, K. Anewalt, "Undergraduate research: Learning strategies and undergraduate research", in ACM SIGCSE", pp. 209–213, 2006.

[3] K. Shaw and J. Dermoudy, "Engendering an empathy for software engineering", in ACM Australasian Computing Education, pp. 135–144, 2005.

[4] T. B. Hilburn, G. Hislop, D. J. Bagert, M. Lutz, S. Mengel, M. McCracken, "Guidance for the development of software engineering education programs", The Journal of Systems and Software, 49, 163-169, 1999.

[5] L. Layman, L. Williams, K. Slaten, S. Berenson, M. Vouk, "Addressing diverse needs through a balance of agile and plandriven software development methodologies in the core software engineering course", Int. J. of Eng. Educ., vol. 24, no. 4, pp. 659– 670, 2008.

[6] D. F. Rico and H. H. Sayani, "Use of agile methods in software engineering education", in Proc. Agile 2009 Conf., Chicago, USA, pp. 174–179.

[7] D. Rover, R. Scheel, C. Ullerich, J. Wegter, C. Whipple, "Advantages of Agile Methodologies for Software and Product Development in a Capstone Design Project", Frontiers in Education Conference (FIE), 2014 IEEE, 22-25 Oct. 2014.

[8] Mondragon-Torres, A., A. Kozitsky, C. Bundick, E. McKenna E. Alley, M. Lloyd, P. Stanley, R. Lane, "Work in progess – an agile embedded systems design capstone course", Proc. 41st ASEE/IEEE Frontiers in Education Conference, 2011, pp. F4F-1-3, 2011.

[9] Mondragon-Torres, A., "An agile embedded systems capstone course", Proc. 2013 ASEE/IEEE Frontiers in Education Conference, pp. 127-133, 2013.

[10] Stansbury, R., M. Towhidnejad, J. Clifford, M. Dop, "Agile methodologies for hardware/software teams for a capstone design course: lessons learned", *Proc. ASEE Annual Conference*, 2011.

[11] Grimheden, Martin, "Can agile methods enhance mechatronics design education", Mechatronics, Elsevier Science, 23(8), pp. 967-973, 2013.

[12] Grimheden, Martin, "Mutual learning experiences: mechatronics capstone course projects-based on Scrum", Proc. ASEE Annual Conference, 2012.

[13] R. A. Vicentin, M. T. P. Santos, "O impacto de adoçãoo do scrum na atual organização das equipes de desenvolvimento", in Revista T.I.S. - Tecnologia, Infraestrutura e Software, vol. 3, no. 2, pp. 134– 157, 2014.

[14] Viljan Mahnic. "Teaching Scrum through Team-Project Work: Students' Perceptions and Teacher's Observations". International Journal of Engineering Education, 2010.

[15] L. Layman, L. Williams, K. Slaten, S. Berenson, M. Vouk, "Addressing Diverse Needs through a Balance of Agile and Plan-driven Software Development Methodologies in the Core Software Engineering Course", International Journal of Engineering Education, 24(4), 659-670, 2008.