ABC-Sprints: Adapting Scrum to Academic Game Development Courses

Jonas Schild  
Entertainment Computing  
University of Duisburg-Essen  
47057 Duisburg, Germany  
jonas.schild@uni-due.de

Robert Walter  
Entertainment Computing  
University of Duisburg-Essen  
47057 Duisburg, Germany  
robert.walter@uni-due.de

Maic Masuch  
Entertainment Computing  
University of Duisburg-Essen  
47057 Duisburg, Germany  
maic.masuch@uni-due.de

ABSTRACT

We propose a course design that fits a practical game development project into a regular game design lecture course. As this approach requires a consistent structure, our concept proposes an adaption of Scrum that is based on the game development life-cycle: ABC-Sprints consist of three Sprints to iteratively create Alpha, Beta and Completed versions of a game. We present a detailed walkthrough of the course and give results of a formal evaluation. These indicate that ABC-Sprints help students to manage their workload and to increase productivity over time. Consequently, three teams each presented a game at the end of the course that technically exceeded typical outcomes of game projects without lectures. We hence encourage to apply ABC-Sprints to other curricula and provide a set of recommendations.

Categories and Subject Descriptors
K.3.2 [Computers and Information Science Education]: Computer science education; K.8.0 [Personal Computing]: General—Games

General Terms
Design, Curriculum, Management

Keywords
games, education, Scrum, curriculum, game development, XNA, team projects

1. INTRODUCTION

Teaching game development in Computer Science (CS) studies is a challenging task. It involves a broad range of disciplines and technologies (e.g., graphics, interaction, game design, software engineering, arts). As a result, many courses that offer a whole game development process are intensive or capstone projects (e.g., [5]). Other classes have been distributed across multiple semesters [16] or are taught in parallel at separate departments [8]. Yet applying such solutions to other institutions and their curricula may be difficult.

To fit a game development course into one semester, previous work recommends to choose between either giving a lecture or conducting a project—not both [17]. Such projects typically allow students to create a game but cover only few theoretical topics often related to brainstorming (e.g., [3]). The focus on game creation is understandable, as making their own game is what students motivates most [2]. In addition, the game industry requires project experience and portfolio examples of self-created games, showing the applicant’s high devotion and commitment [17]. Yet we feel that theoretical knowledge on game production and design (e.g., formalizing a concept into elements, such as objectives, rules, or game mechanics [7]) is crucial and should not be dismissed from academic courses.

According to these requirements, our challenge was to fit a practical CS-based game development process and a set of game design lectures into a non-intensive course for one semester (15 weeks). The weekly amount of time spent on the practical part should not exceed 6-8 hours. This raises special requirements for a structured process that supports both creativity and productivity in a team project. It should further allow for flexible, self-controlled time and task management that enables students to react on varying workload in other courses or even projects.

To structure the practical part of a course as a game development process, we propose Scrum [21] as software development methodology. Scrum has been successfully adopted by the games industry, allowing for high quality games to be developed on time [12].

The following section identifies practical industry needs to be met by the game development curriculum. It discusses Scrum as a possible solution and gives examples of previous implementations. In section 3, we present a course concept combining lectures and project work by adopting Scrum through ABC-Sprints. Section 4 evaluates this design through a walkthrough report and a quantitative survey. Finally, we discuss these results, and provide a list of recommendations and best-practices.

2. GAME CURRICULUM AND SCRUM

Game courses in CS education can be categorized in two approaches: the first category uses it as a motivation vehicle for teaching CS topics, the second teaches development of a game per se. The latter can be further separated into game programming and game development classes [22]. Game development class should cover a comprehensive list of topics (e.g., design, programming, business, production) [11].
These topics include facts and theoretical models to be ideally taught in lectures. However, many industry needs are best met in practical projects, e.g. having worked on a self-contained project, having created a game that demonstrates the applicant’s devotion and perseverance [17]. The three most important management skills in this regard are an effective communication with team members, the abilities to follow processes and to meet objectives, and to transform requirements and feedback into planned goals and actionable items [1]. Regarding process optimization in the games industry, many studios propose applying an agile software development method to improve management of their game creation processes [12]: Scrum [23].

Scrum tries to empower small teams, thus placing the Scrum Team in the center of decision making. Instead of implementing a project manager, a Scrum Master acts as interface between the Scrum Team and the Product Owner. The Product Owner defines the overall objectives in a Product Backlog and is responsible for the commercial success of the product. The Scrum Master guards the team from outside interference during each development iteration, known as Sprint. This makes the Scrum Team an autonomous organization: The members can dynamically prioritize their workload in regard to the goals of a Sprint.

As a recent commercial example, the development of Forza 3 reportedly became more predictable, Scrum helped to improve leadership among small teams through empowerment of individuals, and for the first time in the studio history, they hit their schedule exactly on the day [10]. Another major release, Left 4 Dead 2, was also completed on time. Scrum improved task prioritization in comparison to the prequel which was not developed using Scrum [15]. Other advantages, according to the development team of Brøtleguel which was not developed using Scrum, are the emphasis on features over systems, on people over process, and on iterative design [4].

Opposed to that, the academic domain lacks such positive experiences. Gestwicki et al. applied Scrum in an interdisciplinary project covering two parallel courses on game design and game programming. Yet, no hints were given on the impact of Scrum [9]. Fernández-Vara and Tan applied Scrum to an eight-weeks summer course [6]. They did not recommend Scrum for short projects, rather for major software developments. According to them, Scrum lacks guidelines to deal with finalization of a product within a tight deadline. As a result, the students had to go into crunch time for the last two weeks and canceled using Scrum for this period.

Despite these experiences, we applied Scrum to a lecture course as a software development method. We thus intend to allow for a complete game creation process next to a transfer of theoretical knowledge. With regards to the previously identified industry needs, we further expect Scrum to have a positive impact on the following aspects:

**Teamwork** is vital in game development. This affects communication, reliance, conflict management, etc., thus evolving important soft skills.

**Flexibility** concerns the issue to react on changing requirements in awareness of the general objectives.

**Empowerment** makes individuals more confident regarding their capabilities. Instead of a central decision instance, everyone’s competence has impact on decisions which results in higher motivation and commitment.

**Productivity** is crucial in respect to both the limited time frame as well as to the comprehension of tasks to be handled in a game development process.

**Playable prototypes** accompany the game creation process, providing a concrete idea of how a product evolves, which features work, and which are missing.

## 3. COURSE DESIGN

In this section we propose a framework for a game development course design. This covers the course structure, its schedule, constraints, and our adaption of Scrum: ABC-Sprints. Moreover, we illustrate the application of the design to our Game Development course held in the winter of 2009 at the University of Duisburg-Essen, Germany.

The two primary goals of the course design are to let students experience, first, a theoretical lecture on game design, structure and production, and second, a near-to-professional process of developing a game in a team that covers programming, game design, and agile development using Scrum.

### 3.1 Structure and Schedule

It is vital to parallelize the theoretical and practical lecture threads as good as possible. Thus, the participants are able to deepen the knowledge gained in the lectures by practical experience and vice versa. To focus their limited workload on developing game functionality rather than content production, we inform the students that we do not rate the quality of assets (e.g. graphics, 3D models, sounds), which is different from other course designs [13].

The course covers one semester and consists of 15 weeks of teaching. Based on the game development life cycle (Pre-production, Production, Testing) [11], we divide the course into a Design Phase (Pre-production) and a development phase (Production) that we call ABC-Sprints. We omit Testing as a separate phase and give an insight within the Production phase.

**Design Phase.** The goal of the Design Phase is to group the students into Scrum Teams, each having a distinct game concept described in a Game Design Document (GDD). This is the basis for the Development Phase. A team comprises four to five students as recommended by [24].

Before the first meeting, the students write an application letter that describes their backgrounds, skills, and expectations regarding the course. These profiles are intended to balance the teams as recommended widely (e.g. [18]). In the following meetings, several iterations of brainstorm sessions are conducted. Each student has to create a high concept letter that describes their backgrounds, skills, and expectations regarding the course. They are then combined in group discussions, the final GDDs cover the fundamental ideas from every participant, thus they easily can identify with it. Before entering the first Sprint, the teams create a Product Backlog based on the GDD. In consultation with the Product Owner, the team members define and prioritize the necessary features.
ABC-Sprints. The development phase is subdivided into three iterations: Alpha Sprint, Beta Sprint, and Completion Sprint. We hence call this outline of sprints ABC-Sprints. During a Sprint, each Scrum Team is supposed to work autonomously and self-organized. Therefore, the teams must precisely determine their activities for the upcoming Sprint, to be acknowledged by the instructor. One student per team, the Scrum Masters, regularly update the backlogs and provide periodical status updates to the instructor. The instructor also monitors the log files of the versioning system. At the end of each Sprint, the students present their progress to the whole class. This concerns the Sprint goals as well as utilization of Scrum. This way, the teams can exchange their experiences. At the end of the four-week Alpha Sprint, the students should have developed a first prototype in which the game’s basic functionality is implemented. The prototype proofs the game concept, optionally indicating necessary adaptions. During the four-week Beta Sprint, additional levels/enemies/tactics need to be realized, making the game feature-complete. The two-week Completion Sprint is for debugging, polishing and finalizing the assets. Table 1 shows this structure and gives an exemplary lecture outline facing the practical phases and goals. The lecture basis was own material in combination with [7] and [19].

### 3.2 Other Changes to Scrum

Using Scrum within a class curriculum requires modifications [20]. Due to the reduced work schedule, we propose to replace daily Scrum meetings with meeting at least twice a week. In our design, the Sprint durations vary from two (Completion Sprint) to four weeks (Alpha and Beta Sprint).

Another adaptation of Scrum concerns the roles. We propose a student taking the role of the Scrum Master despite being part of the Scrum Team. Other solutions are not feasible due to a lack of resources. The instructor acts as Product Owner, with one exception regarding the creation of the Product Backlogs: In industry projects, the publisher is the client of the Product Owner. But, besides market-driven game concepts such as advertisement games, the game ideas and first prototypes are created by developer teams first and then pitched to a publisher. So the functional requirements—and thus the Product Backlog—can only be defined by the Scrum Team (the students). This proposal then has to be acknowledged by the Product Owner (the instructor).

In addition to the Scrum meetings, all Scrum Masters meet with the instructor once a week in a Scrum of Scrums. In these meetings, the instructor participates as such, not as Product Owner. Thus, only communication problems, or technical issues are discussed, not the features itself. After each meeting, the Scrum Masters report back to their teams. Table 2 summarizes the proposed changes.

### 3.3 Constraints and Tools

Constraints can help to manage a project, but usually the quality and creativity of games is higher without constraints as more freedom induces more motivation among students [17]. One technical constraint for our course was to use Microsoft XNA Game Studio 3.1 which has been successfully applied to several student courses (e.g. [13]), and we had good experiences in previous courses. Code management should be supported through versioning systems, such as

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

### Table 2: Scrum Modifications

<table>
<thead>
<tr>
<th></th>
<th>Scrum</th>
<th>ABC-Sprints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team size</td>
<td>5-7</td>
<td>4-5</td>
</tr>
<tr>
<td>Sprint duration</td>
<td>30 days</td>
<td>2 to 4 weeks</td>
</tr>
<tr>
<td>Weekly workload</td>
<td>40 hrs</td>
<td>8 hrs</td>
</tr>
<tr>
<td>Product Backlog</td>
<td>Product Owner</td>
<td>instructor and students</td>
</tr>
<tr>
<td>created by</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint Backlog</td>
<td>Scrum Team</td>
<td>Scrum Team</td>
</tr>
<tr>
<td>created by</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scrum Master</td>
<td>outside Scrum</td>
<td>part of Scrum</td>
</tr>
<tr>
<td></td>
<td>Team</td>
<td>Team</td>
</tr>
<tr>
<td>Scrums</td>
<td>daily</td>
<td>twice a week</td>
</tr>
</tbody>
</table>
Subversion or Git. Another constraint affects the game design: To limit complexity, the games were restricted to 2D. Developing 3D games requires much more development efforts concerning aspects like modeling, texturing, collision detection, animation, and interaction without necessarily contributing to a good and entertaining gameplay.

4. EVALUATING THE COURSE DESIGN

We evaluate the course design using two methods. We first provide a detailed walkthrough of our course during the winter semester 2009, in which thirteen students have participated in total. The second part shows results of a quantitative evaluation. At the end of this section, we discuss the results and formulate a set of recommendations for applying the course to other curricula.

4.1 Applying the Design—A Walkthrough

In the following, we provide detailed insight to each phase of the course in chronological order.

4.1.1 Idea Creation in the Design Phase

The Design Phase was a very fruitful period. The various ideas were collected and shaped up to different group concepts. A momentum appeared by which the students reached a common ground regarding their concepts and proactively formed three teams. We desisted from rearranging the students afterwards in order to keep the momentum alive.

The three teams (A, B, and C) iteratively created a high concept and a game design document, pitching it to the Product Owner who accepted the proposals, while requesting changes. Especially Team B had to strip down their concept as it suffered from feature explosion. The experience also helped the other teams on prioritizing their features for the Product Backlog.

4.1.2 Alpha Sprint

During Sprint planning, all teams encountered problems in differentiating between features for the Product Backlog and the more fine-grained activities to define for a Sprint. Their first task sets were vague and separated in too large chunks, similar to the Product Backlog. Hence, the instructor gave concrete examples on how to create a set of manageable tasks. While this individual feedback was useful for two teams (Team A and Team B), Team C still had problems. They hardly succeeded in regularly updating the Sprint Backlog, or in adopting requested changes.

During the Sprint, a new member entered the course. He joined Team C, which had the most problems at this point of progress. The new member brought in experience in software development the team had been lacking so far. This led to a good redesign of their game concept, while keeping the core concept—a tower defense game. In the next update meeting, the Scrum Master reported that the new member also boosted the team’s moral.

At the end of the Sprint, Team A reported an issue with network multi-player support. The feature had been requested by the instructor in order to add more depth and lasting appeal to the gameplay. The team invested a great amount of time in analyzing the requirements and consequences of supporting network gameplay. At the end of the Alpha Sprint, they expressed the wish to replace this feature as implementing it in a sufficiently balanced quality would threaten the overall quality of the game. Together with the team, we assessed their technical problem and network support in XNA and chose to agree on the team’s request. Gratefully, the team announced to invest the gained time in balancing and in even creating new features that contributed to the core gameplay. Although this issue caused some serious discussions, it was easy to handle in terms of project management. The team updated the Product Backlog and derived the necessary activities for the Beta Sprint.

In the Sprint milestone presentations the students reported difficulties on estimating the complexity of tasks. This was to expect, since it was the first Sprint for a new team working on a previously unknown task. Moreover, all teams outlined the importance of regular communication and everyone’s commitment to the project in order to work with Scrum. The Sprint Backlogs displayed a high but evenly distributed workload among the teams during the first Sprint. All teams accomplished their Alpha status within time, delivering a basically functional prototype.

4.1.3 Beta Sprint

During the Beta Sprint, all teams entered into crunch mode. While this being expected to happen in the Completion Sprint, all teams set very ambitious goals for the Beta Sprint. They intended to avoid crunch time during the Completion Sprint, especially in consideration of their other courses and the exams on schedule at the end of the semester. Another reason for the early crunch time was the time consuming content creation. Although it was explicitly demanded not to spend too much time working on assets, all teams were very enthusiastic on delivering also a good looking game. Some teams even tried to acquire external artists, but with moderate success.

Regarding self-management, Teams A and B made exhaustive use of the Sprint Backlog and the Burndown Chart in order to coordinate and rearrange their activities. They utilized these tools as a basis for discussion. Team C, however, moreover neglected these tools. They were rarely updated and did not seem to reflect the latest state of the game, also compared to the Subversion log files. The instructor discussed these problems with the Scrum Master of Team C, leading to a meeting with the whole Scrum Team. It helped a lot to talk about their communication problems. Obviously, the new team member had taken over the program lead, implementing many features all by himself. Yet, he did not use the Sprint Backlog for his activities and had thus undermined the Scrum Master’s role. From that discussion on, the Sprint Backlog was updated more regularly and all participants consistently committed code to the repository.

In the Beta presentation, all teams stated an improved estimation of effort, even Team C, but also complained about a very high workload compared to other courses. Some students also faced technical issues, especially architectural problems, concerning their implementations. They requested
more guidance in how to structure software, since they had difficulties with adding or changing features in their existing code structure. Other updates showed integration of content from the accompanying lectures, e.g. Team B created their dialogs and some characters based on concepts taught in the lectures during the Sprint. In the end, they all managed to present a nearly fully functional prototype with improved features, a front-end with GUI and even extended assets.

4.1.4 Completion Sprint
In the Completion Sprint, we provided a code review as well as a lecture on refactoring. The students showed a high interest in this topic, even further on in unit testing. As a result, they recommended earlier integration of code reviews in the course design. For polishing gameplay, the teams invited external beta-testers on their own initiative. Thus, they gained useful feedback on balancing the gameplay as well as on existing bugs.

At the final presentation, all teams successfully delivered fully playable, feature complete prototypes. In their talks, Teams A and B vividly compared their Burndown Charts across the Development Phases. They could easily pin-point problematic phases or crunch time moments and were eager to analyze their progress. In conclusion, they perceived Scrum as being very supportive through all phases of the project with one exception: During the Completion Sprint, Team A primarily used their bug-tracker to maintain their tasks, not the Sprint Backlog.

Opposed to that, Team C admitted that they hardly used Scrum except for the meetings. They used the Sprint Backlog as a short list of software requirements but did not really arrange their work according to the Backlog. They still updated the document according to their progress but rather perceived it as documentation overhead.

All in all, the students seemed very pleased with their outcomes and proudly demonstrated their games to other visitors. One of the visitors, coming from Microsoft, was positively surprised by the games’ qualities and recommended one of the games for upload to their website covering independent games.

4.1.5 The Resulting Games
The three different games (Figure 1) all looked great and were fun to play. They ran smoothly and featured multiple levels that go far beyond prototypical demonstration of a pure concept. Compared to games developed in previous full-time projects, these games were more complex, offering fully functional, much more polished, and balanced gameplay. The most compelling aspect in our opinion was that the games actually reflected their creators’ intentions. The teams really liked their own games and were proud of them.

To further improve the quality, polishing should focus on providing more direct feedback to user interactions, as some controls were difficult to learn without instructions.

4.2 Quantitative Evaluation
We asked the participants to anonymously fill out a survey for the purpose of further improving the course in the future. All 13 students participated. We used four- and five-level Likert-scales to assess the students’ experiences with Scrum during the practical part of the course. The following evaluation provides analysis of results through median, arithmetic mean, and standard deviation; N = 13, if not stated otherwise.

Overall, the participants were very pleased with the project ($\bar{x} = 1; \mu = 1.31; \sigma = .63$; with 1=very pleased to 4=very unpleased) as with the project outcome: their own games ($\bar{x} = 1; \mu = 1.15; \sigma = .88$).

Ten of the thirteen students (76.9%) had prior project experience working in a group. Eight (61.5%) had previously heard of Scrum. Only one of the group had practical experience with all Scrum components (Sprints, Scrum Meetings, Product Backlog, Sprint Backlog, Roles and Burndown Chart). Two more had participated in a Scrum Meeting before.

Generally concerning the contribution of Scrum methods to success of a team project, we asked to rate the effectiveness on a five-level scale, ranging from 1=very effective to 5=not effective. The students rated Scrum meetings as very effective ($\bar{x} = 1; \mu = 1.46; \sigma = .66$). Separating a project into Sprints was perceived as effective ($\bar{x} = 2; \mu = 1.85; \sigma = .80$), as was using a Sprint Backlog ($\bar{x} = 2; \mu = 2.54; \sigma = 1.13$). The Product Backlog ($\bar{x} = 3; \mu = 2.67; \sigma = 1.15; N = 12$), the roles ($\bar{x} = 3; \mu = 2.57; \sigma = .62; N = 12$), and the Burndown Chart ($\bar{x} = 3; \mu = 3.08; \sigma = 1.5$) were regarded as neutral. The Burndown Chart caused the most inhomogeneous ratings with half of the students regarding it neutral and the others equally distributed to both sides of the scale.

Regarding project work in general, we asked for a yes/no decision on which of the following aspects were important to them. Twelve of thirteen (92.3%) each stated that teamwork, individual responsibility, and structured planning were important. Among those who had positively answered, we collected forced-choice ratings regarding the game development project on a four-level scale. The students were very pleased with their teamwork ($\bar{x} = 1; \mu = 1.33; \sigma = .49; N = 12; \text{with} 1=\text{very pleased to} 4=\text{very unpleased})$. Individual responsibility within the team was rated high ($\bar{x} = 2; \mu = 1.58; \sigma = .61; N = 12; \text{with} 1=\text{very high responsibility to} 4=\text{no responsibility})$. The project work seemed rather structured to them ($\bar{x} = 2; \mu = 1.83; \sigma = .58; N = 12; \text{with} 1=\text{very structured to} 4=\text{very unstructured})$.

Inspired by [18] to identify the biggest problems in teamwork, we further asked for the three most positive and three most negative aspects related to their own teams. The most mentioned on the positive side were even distribution of work among team members, mutual motivation and support. On the negative were communication problems and lack of discipline or attitude among team members, further on organizational issues. One other aspect was the workload which was commented to be very high compared to other courses.

The students rated their workload and productivity distributed over the Design Phase and the three Sprints on forced-choice four-level scales. The workload was low in the Design Phase ($\bar{x} = 2; \mu = 2.77; \sigma = .93$; with 1=very low to 4=very high) but very high during the Alpha ($\bar{x} = 4; \mu = 3.77; \sigma = .44$) and Beta Sprints ($\bar{x} = 4; \mu = 3.69; \sigma = .66$).
As a major success, the proposed course design enabled the participants to develop their own playable and balanced game during a lecture course. The participants gained comprehensive practical experience in game development but also in setting their own goals and following them through a plan, and to take responsibility in being part of a self-controlled team.

4.3 Scrum

We think that Scrum has strongly contributed to the project outcome and both ratings and comments affirm this impression. Mapped onto a stripped version of the game development life-cycle, the ABC-Sprints provide a decent frame for self-deployed game development in a student team. Most students also know the concept of alpha and beta versions from commercial products and like this paradigm. They provide clear goals for them. Further on, it was very helpful to always have a running prototype with a functional alpha version by half of the semester and we strongly recommend to follow this iterative approach.

Monitoring the project through Burndown Charts and versioning log files allowed easy supervision for the instructor—and for the team members. Though, not all teams always maintained their logs. Especially Team C had problems updating their Backlogs and, as a result, did not use it. So, to apply Scrum even more consistently requires that all team members, particularly the Scrum Master, is trained in using Scrum. In unbalanced teams a member who significantly outperforms the others, should act as Scrum Master or at least actively support using Scrum.

4.3.2 Lecture Content

Regarding the content of the lectures, we managed to find a sequence of topics that corresponds quite well to the game development process. Yet, not all content needed can be provided before the Sprints start. Thus, some students would have preferred to have more lectures at the beginning of the course. This would provide further insight on the game development process and leave more time during the semester for practical work. From our point of view though, the idea creation phase should not be overloaded to still allow for a creative process. We also feel that the weekly lectures provide a rhythm for meeting with the whole group that supports the overall structure of collaboration. We intend one adjustment regarding teaching game interaction: we would recommend to emphasize more on interactive feedback during the beta sprint as some game features were difficult to learn without instructions.

4.3.3 Workload

One main issue according to both comments and the evaluation was a very high workload. Of course, the ability to handle high workload is a success criteria in the games industry. Previous research highlights industry demands for devoted students, who have proven the right attitude and perseverance, even spent their own time on creating a game until finished [14, 17]. Moreover, the instructor had announced a high workload at the beginning of the course. Despite the course being optional for most participants, no one left the course during the semester which we consider an enormous success, taking the high workload into account.
However, besides a lecture course, students have to fulfill other obligations. Creating a game tempts to leave other tasks unfinished, rather focusing on content creation (which was announced not to be rated). In order to reduce workload there are two options: One is to further reduce the concepts and their features which the Product Owner already did by up to 80%. The second is to externalize content creation or to offer pre-created content from web-databases. Still, though, workload always remains high with the spirit of a team—which was great among our group. Even better, the workload went back down during the Completion Sprint which is the opposite of what we expected and of what we experienced in other courses. The ABC-Sprints seem to have significantly improved planning over time. This is also indicated by the slight increase in perceived productivity across the course duration.

### 4.3.4 Adaption to Other Curricula

We intend to provide a basis for establishing more game lectures in other curricula based on ABC-Sprints. The proposed course design fits in one semester while providing both theoretical background and practical experience on developing a game. Based on the preceding evaluation, we provide the following recommendations for adapting the course design:

**Design Phase**
- Grouping is a dynamic process where the game and the motivation for a certain concept is more important than balancing skills among team members.
- Iterative planning from high concept to GDD and further into the Product Backlog requires a loose guidance, best given by example.
- To maximize motivation it is crucial that all team members agree on a common vision and identify with the game.
- Regarding software engineering, students may require early guidance on how plan the structure of game objects and game components.

**Alpha Sprint**
- The main goal is getting used to Scrum (especially using Backlogs and Burndown Charts) and establishing regular communication.
- This Sprint is in danger of a too rough planning, as students tend to underestimate the complexity of their tasks.
- Early code reviews and providing support on architectural problems can help on the development side.

**Beta Sprint**
- This Sprint planning might be over-detailed in order to compensate flaws from the Alpha Sprint.
- Content creation is a time-consuming task, students are in danger of creating assets beyond the feasible.
- The beta version should be played by external testers before the end of the Sprint to allow integration of feedback into the Completion Sprint planning.

**Completion Sprint**
- Bug-tracking can be supported with appropriate software tools. It should still be maintained as a task in the Sprint Backlog.
- The final presentation should be open to visitors, especially from the industry, creating a release event.
- Announcing such an event or popular guests during the semester has a positive impact on motivation. This should be saved for difficult moments.

**Overall**
- The proportion of 4:4:2-weeks for the ABC-Sprints defines a structure and leaves enough freedom for individual responsibility. Other proportions might fit as well, but we recommend neither having less than three Sprints nor to have more two-week Sprints. Longer Sprints help to build the team.
- Not every student makes a good Scrum Master. As the role might be misinterpreted (e.g. as leader who gives orders), switching roles between Sprints could provide a solution.
- Constant reviews of documentation and giving feedback on managing Backlogs is crucial.
- Further integration of agile software development methods such as Test Driven Developments should be considered.
- Interactive feedback on user input is important for gameplay which is often not realized by students. They should learn about this before the Beta Sprint starts in order to adjust the Completion Sprint.
- Lectures and practical phases could be re-arranged. They should consider holiday seasons or other events.

### 5. CONCLUSIONS

In this paper we presented a course design that allows a practical game development project within a regular game design lecture course, resulting in a fully playable game. As described, this approach requires a consistent structure for allowing both ambitious teamwork and flexible time management. As a solution we propose ABC-Sprints, an adaption of Scrum that is based on the game development lifecycle.

We successfully conducted a course using this design. Three teams could each present a decent, well-functional game at the end of the course. They reported a very high workload but were very pleased with the course as with the outcome.

Scrum, despite being used for the first time of most of the students, was easily applied. The majority even would use it in the future on their own initiative. For future iterations, we plan to move coding and refactoring issues to the first phase and offer additional training on using the Sprint Backlogs and the Burndown Charts before the Development Phase starts. To strengthen the role of the Scrum Master, we consider switching roles between the Sprints.
To conclude, ABC-Sprints allow easy integration of Scrum in student game development projects. They provide a feasible structure that cultivates both individual responsibility and communication in a small team, resulting in flexible and goal-oriented teamwork. The ABC-Sprints help students to develop their planning skills through multiple iterations of managing high workload, thus improving productivity. Providing that the students are supported in applying Scrum, we highly recommend to use ABC-Sprints in other game development projects, even if—as in our case—they are no intensive full-time projects.

6. ACKNOWLEDGEMENTS
Many thanks go to the participants of the course at the University of Duisburg-Essen: Team Mario, Team Luigi, and Team Yoshi.

7. REFERENCES