Learning Software Project Management by Simulation – Experience and Recommendations from 20 Years of Teaching

Andreas Bollin
Department of Informatics Didactics
University of Klagenfurt
Klagenfurt, Austria
Andreas.Bollin@aau.at

***Abstract*—Learning software project management skills can be supported in various of ways. In particular, using business games or simulators represents added value in the classroom but also has some drawbacks and should be supported by an adequate didactic concept. This paper summarizes extensive data collected over nearly two decades from the AMEISE simulation environment used in European universities. It focuses on the output of simulation runs and aims to achieve three objectives: first, reporting on the setting, secondly, sharing lessons learned; and, finally, providing 34 recommendations for educational and training contexts. The paper starts by describing the simulation environment and its use in the lectures over the past 20 years; then, it presents an analysis of qualitative data collected, showing that, according to students, the simulation is one of the best parts of a course. Following the structure of a typical didactic handout, challenges in preparation and implementation, as well as external factors influencing the quality of teaching, are then highlighted, and approaches to solutions or improvements are systematically derived. Most of the recommendations in this paper also apply to courses without simulators, which can improve software engineering education in the long term.**

***Index Terms*—software project management, simulation, didactic guidelines, experience report, education, game based learning**

I. Introduction

Learning software project management at universities is difficult due to the lack of real-life experience and the inherent complexity of software projects. Although traditional classroom methods like reading books and listening to lectures can provide a solid foundation, they often need to capture the intricacies and unpredictability encountered in real-world situations [10], [17], [27], [30]. However, engaging in simulation runs offers an effective and hands-on approach to learning, as they allow students to practice decision-making, problem-solving, and communication skills in a controlled environment. By mimicking real-life project scenarios and providing the opportunity to learn from mistakes, simulations can significantly enhance students' understanding and preparation for real-world challenges [4], [11]. Thus, while classroom learning is essential, incorporating simulation runs in the curriculum can significantly improve the effectiveness of software project management education, bridging the gap between theoretical knowledge and practical experience [3].

In 2001, researchers at the University of Klagenfurt, the Johannes Kepler University Linz, the University of Stuttgart, and the Carinthia University of Applied Sciences initiated a project known as AMEISE (A Media Education Initiative for Software Engineering), with the primary objectives of (a) providing a highly accurate simulation environment akin to a flight simulator for individuals to gain hands-on experience in software project management, and (b) ensuring that the environment is not perceived as a game, but rather as a realistic representation of real-world scenarios. To accomplish these objectives, AMEISE incorporates a simulation core from the University of Stuttgart [13], which contains data and rules from thousands of software projects meticulously collected and analyzed to create a comprehensive and authentic simulation. The project enables participants to immerse themselves in various project management situations, effectively bridging the gap between theoretical knowledge and practical application while promoting the development of essential skills for successful software project management.

The simulation environment has been used at several European universities for two decades. Extensive data (partially with gaps) on simulation results have been collected. This information has now been combined to learn more about the impact of our classroom interventions. Here, we mainly focus on the output of simulation runs, and so the objectives of this paper are as follows:

- first, to report on the setting that enabled us to collect different types of data;
- second, to present the valuable lessons learned from this wealth of information, highlighting the successes, challenges, and opportunities for improvement;
- third, to derive didactically and methodologically sound recommendations, which hold relevance not only for simulation environments but also for a broader range of educational and training contexts.

With that, the paper contributes to the discourse on practical education by emphasizing the importance of experiential learning and essential skill development across domains.

The paper is structured as follows: Section two addresses related work and discusses ways of learning and training project management skills. Section three briefly describes the AMEISE environment and presents the didactic concept behind the lectures. Section four presents classroom observations and derives recommendations, and finally, Section five concludes with a summary and an outlook.

II. Related Work

Management and simulation games are widely used in the educational sector to teach practical and engaging skills. There are football managers, managers for amusement parks or hospitals - some of them finding their way into the field of education. A typical simulation game is SimCity, which allows players to design and manage their city, making decisions about infrastructure, zoning, and public services. It has been used in several studies [1], [15], but whereas general planning, spatial and management skills are trained, such environments do not support typical software engineering (SE) activities.

Business simulation games and game-based learning (GBL) are more relevant in our context and have been employed to teach software engineering and related processes in recent years. Lin et al. [19] have explored the need for simulating the processes of SE, while Kellner et al. [18] focused on offering a set of simulations as practical guides. Simulation in requirements engineering has also been investigated [29]. Pfahl et al. [24] emphasized the importance of providing computer science and SE students with an understanding of typical phenomena occurring in industrial software projects. Chen and Chong [9] demonstrated that SE education could benefit from simulations that cover collaborative software development, team projects, and social aspects. The role and importance of games and gamification are also mentioned by Bucchiarone et al. in their summary of the 6th International Workshop on Games and Software Engineering [7].

On the other hand, games/simulations to learn software project management are scarce. Wangenheim et al. [31] introduced SCRUMIA, an educational game to teach SCRUM, and observed its effectiveness in undergraduate project management courses. Other researchers [2], [12], [25] have also experimented with teaching SE through games in small software projects, highlighting the importance of using real projects in addition to simulations. Game-based learning has been employed in various fields and is particularly useful for fostering collaboration and active student participation [2]. Studies have shown that learning results from GBL are comparable to traditional methods [14] and that games can improve abilities and skills in software engineering education [8].

Despite the potential advantages, GBL has its drawbacks, such as the need for careful planning and monitoring and the possibility of creating pressure or embarrassment for learners. Hainey et al. [16] suggested considering initial knowledge and educational levels when employing GBL. Paraskeva et al. [23] emphasized the importance of characteristics such as rules, goals, objectives, outcomes, feedback, conflict, interaction, and story representation in successful games. Systematic reviews by Boyle et al. [6] and Connolly et al. [10] explored various aspects of game-based learning, indicating that understanding game enjoyment, cognitive and emotional involvement, and matching desired learning outcomes is crucial for successful game integration in learning.

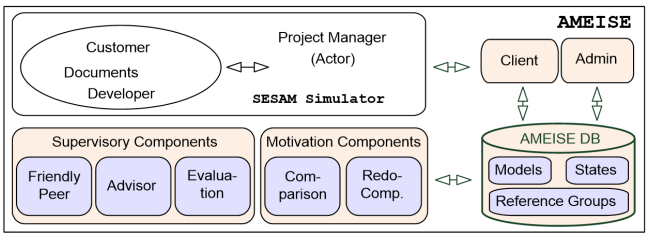

Fig. 1. AMEISE is built around the simulation environment SESAM and extends it by a set of features and supervisory/motivation components for both the trainers (e.g., via Administration, Monitoring, or Evaluation tools) and the trainees (e.g., via the Friendly Peer or Advisor agents).

The AMEISE environment differs from the abovementioned approaches as it explicitly supports trainers and trainees via tools. Furthermore, it is not a game, as its simulation engine relies on a vast set of empirically validated rules. Its virtual developers have their edges and idiosyncrasies; as in real life, events are not always deterministic. Thus, enjoyment and emotional involvement are part of the approach, and being in use for more than 20 years shows that the environment is still a valuable asset for our lectures.

III. The AMEISE Environment

Since its start in 2001, the simulator (free for academic institutions) has been used in schools, universities, and industry. Installation and maintenance can be learned in irregularly offered but free workshops. Apart from the fact that it is not a commercial product, AMEISE is easy to use by trainers and trainees, and this section briefly describes the environment and its components.

A. Components and Models

AMEISE is a Client/Server system that uses a simulation engine called SESAM. Developed by the University of Stuttgart, SESAM is an educational model written in Ada95 and a proprietary rule-based language that the AMEISE team used as an initial prototype, evolving it in a series of iterations to the currently available AMEISE system. The system comprises around 140,000 lines of Ada95, C, and Java code. It utilizes a MySQL database to store crucial data from simulation steps for later assessment and visualization. Developed between 2001 and 2005, the project has been in the maintenance phase, with a significant extension in 2007. This extension introduced a tool for generating ODP files from simulation runs to present key data and decision effects in the classroom. Over 45 developers have contributed to the system, and its stable 32-bit version can handle 300 client requests simultaneously, accommodating larger classroom settings smoothly.

The system has been designed for blended learning situations (see Fig. 1 for a rough overview of the existing modules).

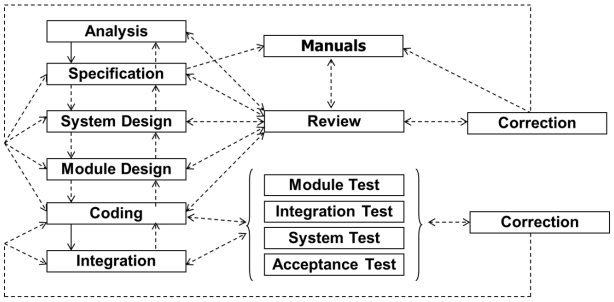

Fig. 2. The QA model is document-driven and implements a Waterfall-like development process. However, a trainee can switch between phases at will during a simulation.

Group	Duration	Costs (€)	AFP C (%)	# E / KLOC	APF Man. (%)	# E / Page
sim-301	274	216,420	95.57	12.86	91.20	0.70
sim-302	254	204,220	97.69	10.50	96.25	0.23
sim-303	214	214,060	94.25	19.08	93.60	0.56
sim-304	218	193,460	96.79	9.45	96.76	0.19
sim-305	148	179,220	93.44	20.44	94.07	0.42
sim-306	242	178,380	97.35	10.96	95.69	0.29

Objectives:
Duration: < 270 days AFP Code: > 95 % AFP Manual: > 95 %
Costs: <= 225,000 € Error/KLOC: < 12 Error/Page: < 0,5

Fig. 3. Exemplary table, generated automatically for the trainers, summarizing the goal achievement of some simulation groups. Color codes are used to highlight successful and less successful simulation runs.

Group	C	E	cpm
sim-303	214,060	18.52	11561.40
sim-306	178,380	23.81	7493.06
sim-311	386,180	25.78	14978.50
sim-426	190,740	12.84	14851.70
Practitioner	217,180	21.64	10,036.00

Fig. 4. Table summarizing the cost of non-productive time in Euro (C ... total project costs, E ... total expenditure in person months, cpm ... costs per person months).

It offers immediate feedback and help during simulation runs, distinguishing it from SESAM. The system includes a Client, an administration environment for user management, three supervisory, and two motivation components, all written in Java. The ”heart” of AMEISE is the MYSQL database, which contains simulation settings, active simulation states, and the complete trace of historical simulation runs. The simulation core itself is not multi-user capable. Still, by storing states in and retrieving states from the database, the system can handle simulation requests from several clients at a time. The system provides supervisory components such as a Friendly Peer (an agent process that takes action and can intervene when bad decisions are made), an Advisor (a plug-in that simulates an experienced colleague to whom you can go with questions), and an Evaluation Component (that produces diagrams and tables for visualizing critical data from simulation runs). At the same time, Redo and Comparison Components are available for motivational reasons (although, as we will see later, these extensions are usually hidden during the first simulation run).

Two models of different complexity and learning objectives are available. There is a model for training maintenance activities (MA-model) and one focusing on quality assurance (QA-model). The QA model is scalable in complexity between 50 and 3000 APFs (Adjusted Function Points), and to a great extent, we use the so-called QA-200 model in our lectures, requiring the trainee to manage a 200 AFP project within nine months and a budget of 225,000 Euro. The model is document-driven (see Fig. 2) and can not be used to simulate agile processes. But, within the given process boundaries, students can move freely and set any activities in the project at any time. Phases can be skipped, and sequences changed, and even a code-and-fix approach is possible.

B. Didactic Approach

The educational model has already been described elsewhere [21]. Briefly, it consists of the following phases:

1) Preparation (90 min.): Trainees are introduced to the AMEISE simulation environment several days before the first simulation run. They are provided with a handout that includes valuable hints and a list of virtual software developers indicating their hourly rates of wages and specific qualifications in various engineering activities. In addition to relevant model-related knowledge, trainees are given sufficient information about the environment to use it properly in this session.
2) Planning for the first simulation (60 min.): After receiving the introductory information, trainees (up to four students) are requested to prepare a simplified project schedule containing the allocation of virtual developers to phases and activities within their simulated project.
3) The first simulation (180 - 240 min.): The first simulation should occur with an instructor present. Completing their simulation runs takes about three to four hours, depending on the student’s preparation level (quality of project plan, environment). During the simulation, students can hire (and fire) employees (with a given salary and a spectrum of qualifications) and assign tasks such as talking to the customer, designing, coding, participating in review teams, testing, or integrating. Assignments may be canceled instantly or after the completion of the current subtask. The students are asked to take notes and adjust their plans depending on necessary changes caused by unforeseen events.
4) Feedback after the first simulation (90 - 120 min.): Feedback concerns expended time and budget, degree of completion, and the number of errors in code and documentation (see Fig. 3 and Fig. 4 for an example). The reflection is broken down into particular topics. Diagrams show task assignments over time, and quality growth indicators are also given. There are also several assessment options available, including online

assessment in the presence of the instructor (recommended for immediate feedback on the first simulation run), individual online assessment in the absence of the instructor for different simulation runs, delivery of a detailed evaluation report generated for each AMEISE simulation, and instructor feedback in a plenary session with discussion and comparison of exciting results.

5) Planning for the second simulation (60 - 90 min.): A second simulation run is highly advisable for didactic reasons [20]. Trainees demonstrate their ability to improve simulation results through a second simulation. Motivated by the challenges faced during the first simulation, they invest more effort in planning the second run. Before starting, students submit an improved plan, which includes a fine-granular schedule using the open-source version of ProjectLibre (www.projectlibre.com).
6) The second simulation (120-180 min.): Familiarity with the AMEISE environment is gained already during the first simulation. Hence, the presence of an instructor is optional for further simulation runs. The setting allows trainees to decide when to start their second run. They may suspend a simulation run and resume it until a given deadline.
7) Feedback on the second simulation (60 min.): While all feedback options described above exist, trainees should be able to interpret the results provided by the online assessment feature already themselves. In addition to discussing the trainees' performance in the second simulation, a plenary session includes an analysis and discussion of deviations between the first and the second simulation runs.

Overall, the AMEISE educational model effectively allows trainees to learn project management skills through experimental learning by simulating real-life scenarios in a safe environment. With AMEISE, they can learn from their failures and improve their skills without harm.

IV. Lessons Learnt

Since 2002, the AMEISE educational model has been used by more than 2,500 students in over 2,000 simulation runs. Only for some runs and universities have data persisted electronically. Still, we restored hand-written feedback from six simulation runs between 2003 and 2006 within this paper's scope. We collected electronic data from two universities between 2006 and 2022, including results from 2,035 students in 1,774 simulation runs. Of these, data from 10 simulation runs had to be excluded as they had particular tasks to implement (such as deliberately skipping project phases), which would distort the statistics. Regarding the distribution of students, 830 students (in 24 cohorts) are from the University of Klagenfurt, and 1,195 students (in 25 cohorts) are from the Technical University Košice.

After some qualitative insight into typical AMEISE courses (IV-A), the chapter follows the typical structure of didactic handouts. First, challenges/problems are identified: two from the preparation phase (Sec. IV-B), nine during the implementation of the course (Sec. IV-C), and four stemming from external factors (Sec. IV-D). Then each of them is followed by recommendations for action.

A. Qualitative Feedback

Different feedback systems accompany the courses at our universities. At the end of a semester, students evaluate the courses according to various criteria (which have changed over time since 2003). They include, among others, the working atmosphere, knowledge transfer, available documentation, learning method, and equal treatment issues. But, since the beginning, there is also room for personal comments (qualitative responses).

As early as 2003, the evaluations show that the AMEISE simulations are a highlight for students during their studies. The courses are consistently evaluated with the best possible grade; little information can be taken from the quantitative part (except that the trainers could convey the contents appropriately). But, the qualitative part is interesting and influenced the further development of the system and the courses. The following comments were received most often:

- In the first year, we got a lot of feedback concerning the user interface of the AMEISE client. Students asked for progress bars, tables, graphs showing the status of deliverables, and a tool for planning and searching their command history. We improved the look and feel but never implemented game-like elements. Instead, in our lectures, we explained that managers are responsible for keeping track of everything independently in real life. AMEISE is not a computer game.
- Another issue that bothered the students was the high response time of the simulation core (about 4-6 seconds per simulation step). We continuously optimized the system and added more simulation cores to it. Since 2006 the system has been running fast and stable (with approx. one second per simulation step, even at high load).
- After resolving technical impediments, students provided more feedback on the task and simulation. The comments are related to two aspects: the interaction with the developers and estimation problems. Comments were often emotional: "So if one of my employees in my company had complained like that, I would have fired him long ago." Yes, sometimes developers in AMEISE complain about the work to be done, and we use this to reflect on leadership and motivation in our courses. Many students also mentioned: "Oh, it's so hard to tell when a task is finished." Thus, we started to address estimation tasks in our lectures and included several examples and techniques in our handouts.

In addition to that, we also periodically collected qualitative feedback from the lecturers. Apart from hints of technical improvements (that are skipped here), the following two recommendations changed the way we organized our courses:

- Students used the helper components (Advisor, Friendly Peer) instead of thinking deeply about decisions them-

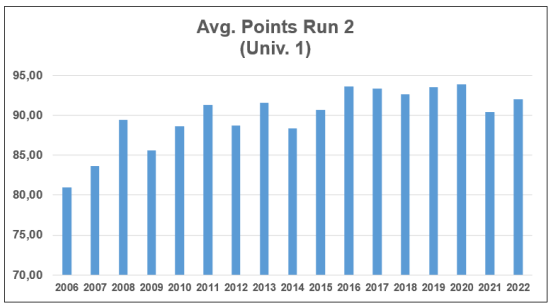

Fig. 5. Number of points achieved at the end of the second simulation (100 points is the maximum).

selves first. As this was observed often at the University of Klagenfurt, we decided to deactivate the helper components for the first simulation runs as default in 2007/2008. This helped to increase the performance of the groups further (see Fig. 5, years 2007/2008).

- The standard set of introductory slides took a lot of work to teach. The slides were very dense and contained a lot of technical information. Thus, in 2015/16, we started replacing the old set of slides with a new, improved set (which follows the brain-based teaching approach [28]). With this, again, we managed to increase the performance of the groups at the University of Klagenfurt. But, most important, teachers and students had more fun imparting or developing the topics.

B. Preparation

The study revealed typical problems that trainees encountered during the planning phases. Specifically, two common challenges were identified:

P1 Lack of knowledge about processes, reviews, tests, and
P2 Lack of experience in estimating the duration of tasks (especially review activities).

Students taking part in simulation runs are generally at the end of their Bachelor's program in Computing Science or Information Management. They know how to specify and design software, know at least one object-orient programming language, can use collaborative tools, and should have had at least two major software engineering courses (including lab classes). Thus they should know about process models, effort estimation, and quality assurance activities. However, they usually have not been put into the manager role, so they know about facts and rules, but the experience on how to apply the knowledge still needs to be added.

Several recommendations can be made to address the challenge **P1** of trainees needing more experience in processes, reviews, and tests.

R01 Trainees should be encouraged to engage in group discussions about typical activities involved in the project. The reflection will help them to understand the processes better and identify potential challenges or areas for improvement.
R02 Trainees should be motivated to do a dry-run of activities (in a fictional software company) and think about an imaginary product to be developed, which can help them identify any issues and make necessary adjustments before starting the project.
R03 Checklists or process maps can be provided to trainees, which can serve as a helpful reference tool throughout the project and help them to stay on track and ensure that essential steps are noticed.

Several recommendations can be made to address the challenge **P2** of trainees needing more experience in estimating the duration of tasks, especially reviews.

R04 A summary of the most important rules of thumb and calculation models can be provided to trainees to help them with estimate tasks.
R05 Trainees can be encouraged to use planning poker, a technique where team members collaboratively estimate the effort required to complete a task.
R06 The first project plan should be created together in the classroom to get instant feedback and to provide a better understanding of the requirements and challenges involved.
R07 Trainees should hand in their plans a few days before the simulation starts, allowing other groups to provide feedback and identify potential issues.
R08 A refined project plan should be created for the second simulation run to improve the overall quality of simulation runs and allow trainees to demonstrate their ability to improve their results.

These recommendations help trainees feel better prepared for the tasks, yield better plans and thus improve their project planning skills.

C. Execution

The study also identified several common pitfalls substantiated in the collected data that trainers encountered during the execution phase of the project. The nine most frequent cases are:

E1 Adding too many personnel at the beginning of a project (producing high cost and a lot of communication overhead);
E2 Focusing too much on duration in the first simulation run and neglecting it in the second run;
E3 Forgetting about essential phases, such as integration and system test;
E4 Struggling to keep documents and code consistent;
E5 Testing-in quality instead of guaranteeing the quality of documents from the beginning;
E6 Starting follow-up phases too early;
E7 They let developers waste time which costs a lot of money;
E8 Sticking too much to the plan and running into hard-to-solve situations;
E9 Not monitoring the budget and activities of developers effectively.

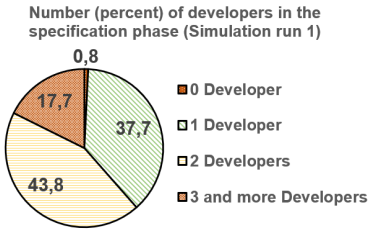

Fig. 6. Proportion of developers hired to work on the system specification (n=769). Due to the simplicity of the task, a single (and highly-skilled) developer would have been enough.

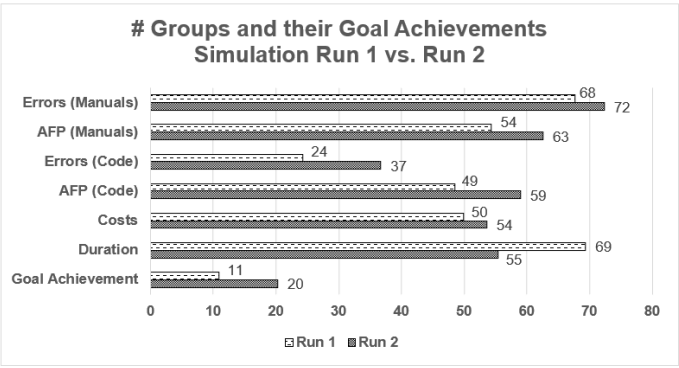

Fig. 7. Number of student groups who have achieved the respective objectives. The parameter "Goal Achievement" means that all objectives have been fulfilled.

The following two recommendations can be made to address challenge **E1** of adding too many personnel at the beginning of a project.

R09 Trainees should be encouraged to discuss effort distribution permanently during the simulation run. For practice reasons, they should estimate the duration of a simple task when working alone and then with an increasing number of developers. They should calculate the number of necessary communication channels and interfaces for that. The necessary considerations can help them to identify the most effective distribution of resources and ensure that everyone understands their roles and responsibilities.

R10 Trainers should discuss the influence of skills on productivity and costs. The discussion will help trainees decide when and how to add which personnel resources to a project. Fig. 6 shows that less than 40 percent of the groups in the first simulation run have only one person, the most appropriate person, to write the specification.

By implementing these recommendations, trainees can be supported to manage resources and optimize productivity throughout the project life-cycle effectively.

Several recommendations can be made to address the challenge **E2** of trainees focusing too much on duration in the first simulation run instead on quality. Fig. 7 shows that most of the students keep the duration constraints in the first run. Only

1swm-03						
Document	**Analysis**	**Sys.Des.**	**Mod.Des.**	**Code**	**Manual**	**Total**
Spec.	35.0	-	-	-	-	35.0
Sys.Des.	74.0	25.0	-	-	-	99.0
Mod.Des.	72.0	21.0	37.0	-	-	130.0
Code	72.0	11.0	10.0	21.0	-	87.0
Manual	47.0	-	-	-	75.0	85.0

Fig. 8. Typical example for a table showing the types of errors in the artifacts. Errors propagate if no quality assurance activities are conducted. In the simulation run of user "1swm-03" there is an inconsistency in the Specification and System Design document as the number of Analysis errors increases.

in the second run quality in terms of numbers of errors and functionality is valued.

R11 Point out that achieving high-quality results requires investing effort in quality assurance already in the early phases. Show simulation runs that even saved money by investing effort in the early phases of the development. This will encourage them to focus on producing high-quality work from the outset rather than sacrificing quality for cost savings.

R12 The importance of the quality of predecessor documents (necessary in the process life-cycle) should be emphasized to trainees. Let the students talk about errors of different types in a fictive document that is the basis for follow-up activities. Then, let them ponder the effects of these errors (and when they can be detected) during the whole project's lifetime. Actively dealing with the life span of errors will help them understand how the quality of early work can impact the project's overall success.

By implementing these recommendations, trainees can learn to prioritize cost and quality throughout the project life cycle.

Several recommendations can be made to address the challenge **E3** of trainees forgetting essential phases, such as integration and system test.

R13 Trainees should be advised to track their process and progress throughout the project. This will help them to identify potential gaps or issues and ensure that all critical phases are completed.

R14 Checklists can be provided to trainees as a reference tool and ensure that essential steps are noticed.

R15 Trainees should be reminded that, even though they are using a simulation environment, AMEISE is not a game, and there are no save points where one could go back and get rid of wrong decisions. This will help them stay focused and take all necessary steps to ensure the project's success.

By implementing these recommendations, trainees can learn that managing projects means actively using tools and techniques to remember decisions and work through endless piles of activities.

Several recommendations can be made to address the challenge **E4** of keeping documents consistent.

R16 Trainees should be shown how errors propagate throughout the project, using a visual representation such as a

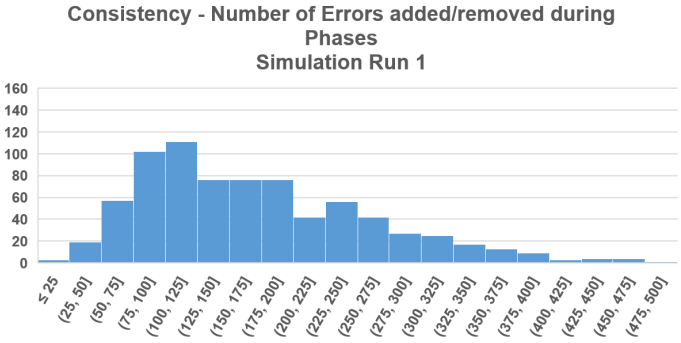

Fig. 9. Diagram showing how many groups of the first simulation introduce or eliminate a certain number of errors over the respective phases. It can be seen here that the peak is at 100-125 errors, but many groups accumulate significantly more errors.

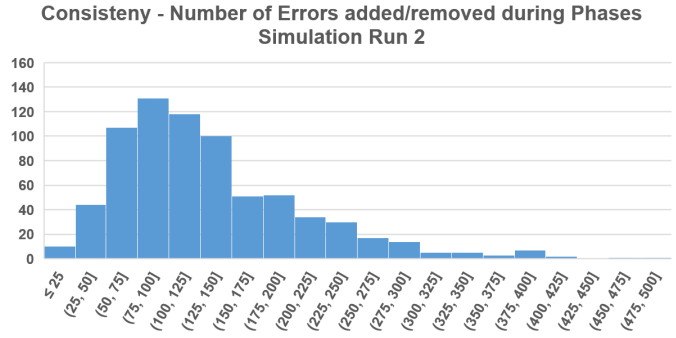

Fig. 10. Diagram showing how many groups of the first simulation introduce or eliminate a certain number of errors over the respective phases. The peak is still at 100-125 errors, but the distribution then falls off more sharply.

diagram or table (see Fig. 8). This will help trainees to understand the importance of document consistency and the potential impact of errors.

R17 Instructors should (again) explain the different types of errors that can occur in documents and how they are related. This will help trainees to identify potential errors and take steps to ensure document consistency.

By implementing these recommendations, trainees can learn to effectively manage different types of documents and ensure that errors are identified and corrected before they harm the project. Figures 9 and 10 show that the explanation after the first simulation runs helps reduce error propagation between the documents.

Several recommendations can be made to address the challenge **E5** of testing-in quality instead of guaranteeing the quality of all inter-project documents.

R18 Trainees should be encouraged to consider which is more cost-effective: extensively testing a buggy system or maintaining high-quality standards from the outset. This will help them understand the importance of ensuring document quality and how it can save time and resources in the long run.

R19 In the simulation environment, cost-saving can be demonstrated by showing how investing effort in quality assurance from the beginning can reduce the likelihood of errors and ultimately save costs in testing and bug-fixing. Data showing the cost-saving or neutral effects should be presented in the feedback after the first simulation run.

By implementing these recommendations, trainees can learn to prioritize document quality and improve their project management skills.

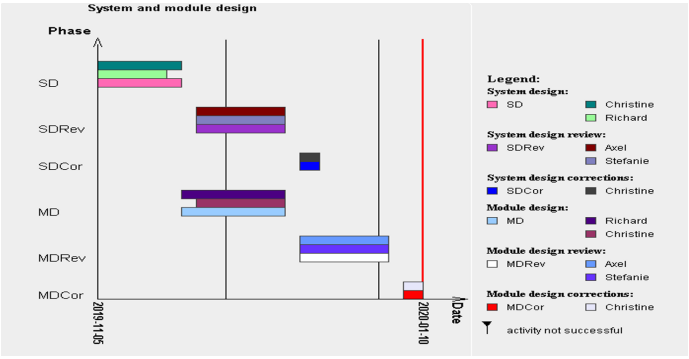

Fig. 11. Simulation run where module design (MD) started too early after the system design (SD). The review process (SDRev) and also the correction of system design errors (SDCor) started (and ended) after the module design was finished. With that, a lot of SD errors were propagated to MD.

Two recommendations can be made to address the challenge **E6** of starting follow-up phases too early.

R20 Trainees should be explained how errors can propagate throughout a project and between phases of a project. This will help them to understand the importance of completing each phase whenever possible before moving on to the next and avoiding potential issues.

R21 Within the simulator, the consistency between documents and Gantt charts can be shown with optimal and impeding process steps (see Fig. 11 for a counter-example). This will help trainees to understand the importance of following proper project management practices and completing each phase before moving on to the next.

By implementing these recommendations, trainees refresh their knowledge about dependencies between documents and phases.

Several recommendations can be made to address the challenge **E7** of developers goofing around and non-productive time costing money.

R22 Trainees should be given an overview of a typical working day of a software developer, including their various tasks and how time is managed. This will help trainees to understand the importance of efficient time management and how non-productive time can impact project budgets.

R23 The cost-saving potential of reducing idle time can be demonstrated in the simulator (see Fig. 4 as an example), showing trainees how significant savings can be achieved when non-productive time is minimized. When looking at typical simulation runs of students, these savings might be more than half the monthly budget per person and

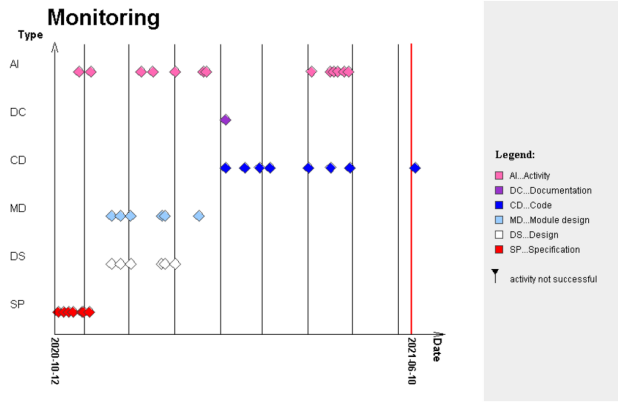

Fig. 12. All the monitoring activities during a simulation run. The specification, system, and module design documents, as well as the code, have been checked regularly. Also, the activities of the developers (labeled with "AI") have been checked. However, one activity is missing and needs to be included: checking the budget.

often range from 5,000 to 10,000 Euros per developer per group.

By implementing these recommendations, trainees actively deal with personnel costs in a project, and they learn (sometimes painfully) that even minor differences can cost big.

The following two recommendations help to address the challenge **E8** of sticking too much to the plan and running into hard-to-solve situations.

R24 Trainees should be advised that the most successful project managers have more than one plan and regularly update their plans according to changing circumstances. This will help trainees to understand the importance of flexibility and adaptation in project management.

R25 Trainees should be encouraged to remember their experiences during the first simulation runs and how likely they could not follow their plans. Ask them always to make notes and to apply their experience with planning accuracy to future projects.

By implementing these recommendations, trainees can learn to prepare several plans, deal with unexpected situations and thus optimize project outcomes again.

Several recommendations can be made to address the challenge **E9** of not monitoring the budget and activities of developers.

R26 Trainees should know that real-time monitoring takes time but is a crucial aspect of project management (you can present and discuss counter-examples like in Fig. 12).

R27 Instructors should explain the importance of balancing loose and active control, where too much control can stifle creativity and too little can lead to poor outcomes.

R28 Show successful groups and bring examples of those who effectively monitor their projects.

R29 If available, give examples of your own (or existing) projects that nearly failed due to loose control.

By implementing these recommendations, trainees can learn to balance control and creativity and improve their leadership skills.

These recommendations should open the eyes of the students in several aspects: avoid idle time, do not only one plan, do several plans, discuss unclear process steps with others, and keep track of what you are doing but be flexible in changing environments.

D. External Factors

The teaching process is very important [26] and among others, preparation, the formation of teams, the environment as well as implementation decisions control the quality of teaching.

1) Quality of preparation: Dealing with the influence of external factors on the AMEISE simulation runs is crucial for ensuring a successful learning experience. Before the simulation runs, ensuring that all necessary materials are available online or in print and that every student confirms their receipt is essential. This includes descriptions of the simulation model, instructions, and relevant theoretical materials. Additionally, instructors should consider these materials optimized for brain-based teaching [28], an approach we successfully applied in our AMEISE lectures from 2015 on. We learned that the duration of preparatory lectures should be carefully considered to maximize the learning effect. Too short a duration may hinder trainees' understanding of project management concepts, while too long may lead to disengagement. In our case, the introductory lecture now lasts 90-100 minutes.

After the simulation runs, it is crucial to have a set of best practices and examples of project decisions that had very positive and negative effects at hand. This allows instructors to provide detailed feedback and guide trainees on improving their performance in future simulations. Moreover, instructors should involve examples from every group to ensure that all trainees remain engaged and motivated. This can be achieved by showcasing successful projects, those that faced challenges, and how they overcame them. Additionally, every group should receive a handout in either PDF or print format containing helpful hints, a list of virtual software developers, and an overview of the AMEISE environment. With that we can recommend the following:

R30 It is advantageous to provide sufficient materials and split courses into 15-20 minute parts, and to include activation phases with tasks between each of these parts (following brain-based teaching recommendations).

By implementing these additional recommendations, instructors can ensure trainees receive a comprehensive learning experience and better understand project management principles. This will help them to effectively manage real-world projects and enhance their career prospects in the software engineering industry.

2) Team Size/Composition: The size and composition of teams during AMEISE simulation runs have been shown to impact the results achieved significantly. It is crucial to consider

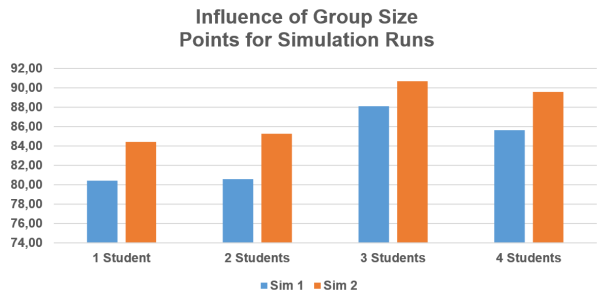

Fig. 13. Average points (100 would be the maximum) achieved for the first simulation runs, depending on the number of students per group.

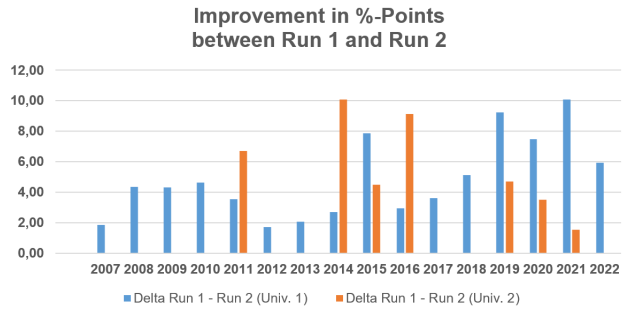

Fig. 14. Average improvement (in points) between the first and second simulation run the University of Klagenfurt and the Technical University Košice.

the number of students on a team carefully, considering factors such as the amount of space available in front of the computer and the opportunity for lively discussions. Our experience has shown that teams of three students are the most effective (see Fig. 13), as they balance space requirements and the ability to collaborate effectively. The dynamic of a team of three students also tends to lead to more creative problem-solving, resulting in better project outcomes. However, it is essential to manage team size to ensure equal work distribution carefully and that everyone has a chance to contribute.

Beyond team size, group dynamics can also significantly impact simulation results. Different personality types have different strengths and weaknesses, and it is essential to consider these when forming teams. In one AMEISE experiment, we investigated the impact of personality types (according to the five-factor model) on simulation results. We found that teams with a mix of personalities (with one manager or coach personality type) tended to perform better than teams with homogeneous personalities [22]. So, by forming teams with a balanced mix of personality types, instructors can optimize the learning experience and help trainees develop critical project management skills.

Based on these observations, two further recommendations can be made:

R31 Whenever possible, form groups of three students for AMEISE simulation runs.

R32 Trainers should take care of different types of personalities and, whenever possible, should form groups with one manager or coach type.

3) Equipment and Environment: The working environment significantly influences the learning and performance outcomes of the AMEISE simulation runs. A recent study [5] found that the simulation outcomes could be improved by creating a suitable environment that fosters effective communication, collaboration, and knowledge sharing among team members.

The study also recommended using appropriate tools and equipment, such as laptops, projectors, and software, to support the simulation runs. In addition, it is essential to ensure that the working environment is conducive to learning, such as minimizing distractions and providing adequate lighting and ventilation. So, instructors must consider the working environment and provide the necessary resources and support to maximize the learning outcomes in their lectures (and simulation runs). The following recommendation can be made:

R33 Students should be able to set up their workspace in the best possible way. This freedom includes space for whiteboards, good screens, and familiar tools for communication.

4) On-Site vs. Online: According to the reports of lecturers, providing a shared communication space is necessary for both on-site and online simulation runs. However, it is advisable to allow students to organize their way of communication to promote their creativity and adaptability. Additionally, the instructors must be easily reachable in case of student questions or concerns. This can be done through various means, such as messaging platforms or video conferencing tools. By ensuring the availability of communication channels, students can receive prompt and effective support throughout the simulation runs.

Besides many problems, one positive effect of the COVID-19 pandemic is that it has forced many educational institutions to shift their teaching and learning practices to online environments.

Data gathered so far shows that whether the simulation runs are conducted on-site or online, there is no significant difference in goal achievement (see Fig. 5 for the final results and Fig. 14 for the improvement between run one and run two). Both methods of simulation delivery have yielded comparable results in improving project management skills. This suggests that instructors can choose on-site or online simulation delivery methods based on their preferences, availability of resources, and students' learning preferences. Larger groups can be accommodated in online simulation runs, leading to better resource utilization and enhanced student collaboration. Overall, the shift towards online simulation runs has shown promising results, and we are expected to continue to do simulation runs in the virtual room again. Based on this observation, a final recommendation can be made:

R34 One should not overthink about holding (or not holding) an online/remote simulation. Students must have enough opportunities for communication and collaboration.

V. Conclusion

Being a mixture of experience report and didactic guidance, this contribution briefly introduced the AMEISE simulation environment and its usage in courses over the past 20 years. Three goals were defined. The first goal was to describe the context in which data could be collected. For this purpose, a well-tested teaching concept was presented. The second goal was to identify challenges in teaching, and the third goal was to provide recommendations. These two remaining goals were achieved by a very systematic approach (problem identification - recommendation - rationale): qualitative and quantitative data were utilized from the historical point of view, and 34 recommendations were derived from them to improve teaching or as advice to teachers in similar courses. The problems identified (such as wrongly placed focus during development phases, underestimating non-productive times, or choosing employees who are not best suited) are independent of a simulation environment. Thus, the recommendations should be helpful in other teaching situations as well.

From today's perspective, the simulation environment will continue to be used due to its curricular fit. However, some changes and extensions are now imminent. First, it is planned to make the system a web service and thus further facilitate access to simulations. On the model level, we are also working on implementing an agile model in addition to the document-driven model focusing on quality assurance. Here we are still collecting data so that an agile rule model leads to an accurate simulation, not a game.

References

[1] P. C. Adams, "Teaching and learning with SimCity 2000," *Journal of Geography*, vol. 97, no. 2, pp. 47–55, 1998.

[2] C. Alvarez, R. Alarcon, and M. Nussbaum, "Implementing collaborative learning activities in the classroom supported by one-to-one mobile computing: A design-based process," *Journal of Systems and Software*, vol. 84, no. 11, pp. 1961–1976, 2011.

[3] A. Bollin, E. Hochmüller, R. Mittermeir, and L. Samuelis, "Experiences with Integrating Simulation into a Software Engineering Curriculum," in *Proceedings of 25th IEEE Conference on Software Engineering Education and Training CSEE&T 2012*, L. H. D. Chen, M. Baker, Ed. IEEE Computer Society Press, April 2012, pp. 62 – 75.

[4] A. Bollin, E. Hochmüller, and C. Szabó, "Teaching Software Project Management by Simulation: Training Team Leaders for Real World Projects," in *Proceedings of the 28th IEEE Conference on Software Engineering Education and Training. Florenz, Italy*. IEEE Computer Society Press, 2015.

[5] A. Bollin, E. Reci, C. Szabó, V. Szabóv'a, and R. Siebenhofer, "Applying a Maturity Model during a Software Engineering Course - Experiences and Recommendations," in *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T), Savannah, USA*. IEEE Press, 2017.

[6] E. A. Boyle, T. M. Connolly, T. Hainey, and J. M. Boyle, "Engagement in digital entertainment games: A systematic review," *Computers in Human Behavior*, vol. 28, no. 3, pp. 771–780, 2012.

[7] A. Bucchiarone, K. M. L. Cooper, D. Lin, E. F. Melcer, and K. Sung, "Games and Software Engineering: Engineering Fun, Inspiration, and Motivation," *SIGSOFT Software Engineering Notes*, vol. 48, no. 1, pp. 85–89, January 2023. [Online]. Available: https://doi.org/10.1145/3573074.3573096

[8] N. E. Cagiltay, "Teaching software engineering by means of computer-game development: Challenges and opportunities," *British Journal of Educational Technology*, vol. 38, pp. 405–415, 2007.

[9] C. Y. Chen and P. P. Chong, "Software engineering education: A study on conducting collaborative senior project development," *Journal of Systems and Software*, vol. 84, no. 3, pp. 479–491, 2011.

[10] T. M. Connolly, E. A. Boyle, E. MacArthur, T. T. Hainey, and J. M. Boyle, "A systematic literature review of empirical evidence on computer games and serious games," *Computers & Education*, vol. 59, pp. 661–686, 2012.

[11] M. Despeisse, "Games and simulations in industrial engineering education: A review of the cognitive and affective learning outcomes," in *Proceedings of the 2018 Winter Simulation Conference*, ser. WSC '18. IEEE Press, 2018, pp. 4046 – 4057.

[12] A. Drappa and J. Ludewig, "Quantitative modeling for the interactive simulation of software projects," *Journal of Systems and Software*, vol. 46, no. 2–3, pp. 113–122, 1999.

[13] ——, "Simulation in Software Engineering Training," in *Proceedings, 23rd International Conference on Software Engineering*. IEEE-CS and ACM, May 2001, pp. 199 – 208.

[14] M. Ebner and A. Holzinger, "Successful implementation of user-centered game based learning in higher education: An example from civil engineering," *Computers & Education*, vol. 49, no. 3, pp. 873–890, 2007.

[15] J. Gaber, "Simulating planning: SimCity as a pedagogical tool," *Journal of Planning Education and Research*, vol. 27, no. 2, pp. 113–121, 2007.

[16] T. Hainey, T. M. Connolly, M. Stansfield, and E. A. Boyle, "Evaluation of a game to teach requirements collection and analysis in software engineering at tertiary education level," *Computers & Education*, vol. 56, no. 1, pp. 21–35, 2011.

[17] W. S. Humphrey, *Introduction to the Team Software Process*. Addison Wesley, 2000.

[18] M. I. Kellner, R. J. Madachy, and D. M. Raffo, "Software process simulation modeling: Why? what? how?" *Journal of Systems and Software*, vol. 46, pp. 91–105, 1999.

[19] C. Y. Lin, T. Abdel-Hamid, and J. S. Sherif, "Software-Engineering Process Simulation model (SEPS)," *Journal of Systems and Software*, vol. 28, pp. 263–277, 1997.

[20] P. Mandl-Striegnitz, "How to successfully use software project simulation for educating software project managers," in *Proceedings - Frontiers in Education Conference*, vol. 1, 02 2001, pp. 19 – 24.

[21] R. Mittermeir, A. Bollin, E. Hochmüller, S. Jäger, and D. Wakounig, "AMEISE - An Interactive Environment to Acquire Project-Management Experience," in *Proceedings of HUBUSKA Third Open Workshop, Klagenfurt, Austria*, April 2006, pp. 1 – 16.

[22] A. Mujkanovic and A. Bollin, "Personality-Based Group Formation - A Large-Scale Study on the Role of Skills and Personality in Software Engineering Education," in *Empowering Learners for Life in the Digital Age, OCCE 2018, Linz, Austria. IFIP Advances in Information and Communication Technology*. Springer, 2019, pp. 207 – 217.

[23] F. Paraskeva, S. Mysirlaki, and A. Papagianni, "Multiplayer online games as educational tools: Facing new challenges in learning," *Computers & Education*, vol. 54, no. 2, pp. 498–505, 2010.

[24] D. Pfahl, M. Klemm, and G. Ruhe, "A CBT module with integrated simulation component for software project management education and training," *Journal of Systems and Software*, vol. 59, no. 3, pp. 283–298, 2001.

[25] S. Qin and C. Mooney, "Using game-oriented projects for teaching and learning software engineering," in *20th Annual Conference for the Australasian Association for Engineering Education*, 2009, pp. 6–9.

[26] E. Reçi and A. Bollin, "Managing the quality of teaching in computer science education," in *Proceedings of the 6th Computer Science Education Research Conference*, ser. CSERC '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 38–47.

[27] M. Shaw, "Software Engineering Education: A Roadmap," in *Future of Software Engineering 2000*, A. Finkelstein, Ed. ACM, 2000, pp. 373 – 380.

[28] M. Sprenger, *Brain-based teaching in the digital age*. ASCD, 2010.

[29] E. Suescun-Monsalve, P. Vallejo, R. Mazo, and D. Correa, "Transparency as a learning strategy to teach software engineering," in *Proceedings of the Congreso Colombiano en Computacion*, 09 2017.

[30] J. B. Thompson, "Software Engineering Practice and Education: An International View," in *SEESE '08: Proceedings of the 2008 International Workshop on Software Engineering in East and South Europe*. ACM, 2008, pp. 95 – 102.

[31] C. G. Von-Wangenheim, R. Savi, and A. F. Borgatto, "SCRUMIA – An educational game for teaching SCRUM in computing courses," *Journal of Systems and Software*, vol. 86, no. 10, pp. 2675–2687, 2013.