International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

An Application of a Game Development Framework inHigher Education

Alf Inge Wang and Bian Wu

Department of Computer and Information Science, Norwegian University of Science and Technology, 7491 Trondheim, Norway

Introduction

Games have been used in schools for many years to help children learn skills in math, language, geography, science, and other domains in an interesting and motivating way. Research shows that integrating games within a classroom with children can be beneficial for academic achievement, motivation, and classroom dynamics [1]. There is also evidence that the teaching methods based on educational games are not only attractive to schoolchildren, but also to university students [2]. There have been conducted researches on games concept and game development used in higher education before, for example, [3–5], but we believe there is an untapped potential that needs to be explored. Games can provide teachers in higher education with teaching aids that can promote more active students, provide alternative teaching methods to improve variation,and enable social learning through multiplayer learning games.

Games can be integrated in higher education in three ways. First, games can be used instead of traditional exer- cises motivating students to put extra effort in doing the

exercises and giving the teacher and/or teaching assistants an opportunity to monitor how the students work with the exercises in real time [6, 7]. Second, games can be used within lectures to improve the participation and motivation of students [8, 9]. In this approach, the students and the teacher participate in knowledge-based games. Third, the students are required to develop a game as a part of a course using a game development framework (GDF) to learn skills within computer science or software engineering [10]. This paper focuses on the latter, where game development and a GDFare used in student projects to learn software engineering skills, extending the use of games as a teaching aid in higher education. The motivation of making students develop games to learn software engineering is to bring the students' enthusiasm from playing games to learn courses through game development. In addition, we wanted to investigateif the specific features of a GDF are suitable for teaching software engineering and how game development can be integrated with the education process. More specifically, we wanted to explore how the use of game development and the GDF would affect the learning of software architecture withfocus on the technical aspects of the GDF.

This paper focuses on how the technical aspects of a GDF affect the learning of software architecture, the selection of appropriate GDF for a software architecture course, andhow a GDF can be applied in a software engineering course.The main contribution of this paper is a presentation of a novel GDF concept that can be used in courses that includes software development, experiences from actual usage of the GDF, and some course design considerations.

The rest of the paper is organized as follows. Section 2 describes and motivates for how a GDF can be used in higher education and what criteria should be considered when choosing one. Section 3 describes a case study of applyinga GDF in a software architecture course. Section 4 describes experiences from using a GDF in a software course. Section 5 describes similar approaches, and Section 6 concludes the paper.

1. Game Development Frameworks inHigher Education

This section presents the motivation for applying GDFs in higher education, a model for how GDFs can be integrated with a course, and requirements for how to choose the appropriate GDF for educational purposes.

1.1. GDF and Education. The main motivation for intro- ducing GDF in software engineering (SE) or computer science (CS) courses is to motivate students to put more effort into software development project in order to improve software development skills. Game development offers an interesting way of learning and applying the course theory.By introducing

C

C

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

a game development project in a course, the students have to establish and describe most of the functional requirements themselves (what the game shouldbe like). This can be a motivating factor especially for group-based projects, as each group will develop a unique application (the game); it will encourage creativity, and it will require different skills from the group members (art, programming, story, audio/music). The result will be that the students will have a stronger feeling of ownership to the project. Furthermore, students also could learn about game development technology. The main disadvantages by introducing a game development project and a GDF into a SE or CS course is that the student might spend too much time on game-specific issues and that the project results might be difficult to compare. It is critical that the students get motivated applying a GDF in a course and that they get increased motivation for learning and applying course theory through a game development project.

Tom Malone has listed three main characteristics that make things fun to learn: they should provide the appro- priate level of challenge, they should use fantasy and abstractions to make it more interesting, and they should trigger the player's curiosity [11]. These characteristics candirectly be applied when developing a game for learning purposes. However, we can also consider these characteristics when introducing a GDF in a SE or CS course. By allowing the students to develop their own games using a GDF, such

projects are likely to trigger students' curiosity as well as provide a challenge for students to design fun games with their knowledge, skills, imagination, and creativity. The levelof the challenge can be adjusted according to the project requirements given in courses by the teacher. Thus, the challenge level can not only be adjusted to the right level formost participants, but also tailored for individual differences. As the students will work in groups, group members helping other group members can compensate for the individual differences. An open platform and agile courses requirements should be provided for students to design their own games,combined with their ability, fantasy, and comprehension oflecture content.

The main benefit of using a GDF as a teaching aid is that it can be a motivating initiative in courses to learn about var-ious topics such as software requirements, software design, software architecture, programming, 2D and 3D graphic representation, graphic programming, artificial intelligence,physics, animation, user interfaces, and many other areas within computer science and software engineering. It is most useful for learning new skills and methods within a specific domain but also useful for testing and rehearsing theory byapplying know skills and knowledge in a project using a GDF.

Circulatory Model of Applying a GDF in a Course. Thereare several good reasons for introducing a GDF and game development projects in CS and SE courses as describedin previous section, but in order to make it a success it is important that the GDF is well integrated with the course. Based on our experiences, we have developed a circular model for how to apply a GDF in a CS or SE course through six steps (see Figure 1). The model is intended for courses where a software development project is a major part of thecourse.

To choose one appropriate development platform according to the course content, it is important to consider the process of the course related to the development project. This process starts with choosing an appropriate GDF (step A) the course related to some requirements (described in the next section). Next, the design of exercises and projects (step B) must reflect the limitations and constraints of the chosen GDF. In the initial phase of the student project, itis important that the students get the required technical guidance and appropriate requirements (step C) related to the GDF. It is important that the students get to knowthe GDF early, for example, by introducing an exercise to implement a simple game in the GDF. It is critical that thereis sufficient course staff that knows the GDF well enough togive the required feedback. The next step is for the studentsto start designing and implementing (step D) their own game according to the constraints within the course and the GDF. After the students have delivered their final version of theirproject implementation and documentation, the students should get the chance to evaluate and analyze (step E) theirown projects to learn from their successes and mistakes. This information

Received: 8 August Revised: 15 August Accepted: 20 August
Index in Cosmos
September 2020 Volume 10 ISSUE 3
UGC Approved Journal
C

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

should then be used to provide feedback inorder to improve the course (step F). The feedback from thestudents might indicate that another GDF should be used or

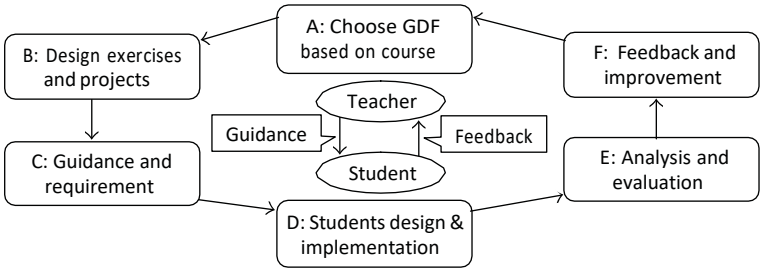


FIGURE 1: Circulatory model of GDF's application in courses.

that the course constraints on the projects should be altered. The core of this model is that the teacher should encouragethe students to explore the course theory through a game development project using a GDF and give the opportunityto improve the game development project through feedback from the students.

1.2. Criteria for Choosing the Right GDF. How to choose an appropriate GDF that easily can be integrated with course content should be based on the educational goals of the course, the technical level and skills of students, and the time available for projects and/or exercises. Based on experiences from using GDFs and from student projects in CS and SE courses, we have come up with the following requirementsfor choosing a GDF for a CS or SE course.

(1) *It must be easy to learn and allow rapid development.* According to Malone's recommendation of how to make things fun to learn, it is crucial that we providethe appropriate level of challenge. If the GDF is too much of a challenge and requires too much to learn before becoming productive, the whole idea of game development will be wasted, as the student will losemotivation. An important aspect of this is that the GDF offers high-level APIs that makes it possible forthe students to develop impressive results without writing too many lines of code. This is especially critical in the first phase of the project.

(2) *It must provide an open development environment to attract students' curiosity.* Malone claims that fantasy and curiosity are other important factors that make things fun to learn. By providing a relatively open GDF without too many restrictions on what youcan produce, the students get a chance to realize the game of their dreams. This means that the GDF itself should not restrict what kind of game thestudents can make. This requirement would typicallyrule out GDFs that are tailored for producing only one game genre such as adventure games, platform games, or board games. In addition, ideally an open development environment should offer public and practical interfaces for developers to extend their own functions. In this respect, open source game development platforms are preferred.

(3) *It must support programming languages that are familiar to the students.* The students should not

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

be burdened to have to learn a new programming language from scratch in addition to the course content. This would take away the focus of the educational goals of the course. We suggest to choose GDFs that support popular programming languages that the students know like C++, C#, or Java. It is also important that the programming languages supported by the GDF have high-level constructs and libraries that enable the programmers to be more productive as less code is required to produce fully functional systems. From an educational point of view, programming languages like Java and C# are better suited than C and C++, as they have more constraints that force the programmers to write cleaner code, and there is less concern related to issues like pointers and memory leakage. From a game development perspective, programming languages like C and C++ are more attractive as they generally produce faster executables and thus faster games.

(4) *It must not conflict with the educational goals ofthe course*. When choosing a GDF it is important that the inherent patterns, procedures, design, and architecture of the GDF are not in conflict with the theory taught in the course. One example of such a conflict could be that the way the GDF enforces event handling in an application is given as an example of bad design in the textbook.

(5) *It must have a stable implementation*. When a GDF is used in a course, it is essential that the GDF has few bugs so the students do not have to fight a lot of technical issues instead of focusing on the coursetopics. This requirement indicates that it is important that the GDF is supported by a company or a development community that has enough resourcesto eliminate serious technical insufficiencies. It is also important that the development of the GDF is not a dead project, as this will lead to compatibility issues for future releases of operating systems, software components, and hardware drivers.

(6) *It must have sufficient documentation.* This require- ment is important for both the course staff andthe students. The documentation should both give agood overview of the GDF as well as document all the features provided. Further, it is important that theGDF provides tutorials and examples to demonstrate

how to use the GDF and its features. The frameworks should provide documentation and tutorials of high quality enabling self-study.

(7) *It should be inexpensive (low costs) to use and acquire.* Ideally, the GDFs should be free or have very low associated cost to avoid extra costs running thecourse. This requirement also involves investigating additional costs related to the GDF such as require- ments for extra or more powerful hardware and/orrequirements for additional software.

The goal of the requirements above is to save the time and effort the students have to spend on coding and understanding the framework, making them concentrate onthe course content and software design. Thus, an appropriate GDF could provide the students with exciting experiences and offer a new way of learning through a new domain (games). The requirements above are also important for thecourse staff, as they will help to find a GDF that would causeless effort spent on technical issues, and incompatibility between GDF and the course contents.

From the requirements above, we acknowledge that thereis a conflict between requirements one and two. The level ofthe freedom the developer is given to make whatever game he likes could be in conflict with providing a development environment that allows rapid development and is easy to learn. A more open GDF usually means that the developer must learn more APIs as well as the APIs themselves are usually of lower level, and thus harder to use. However, it ispossible to get a bit of both worlds by offering high-level APIs that are relatively easy to use but still allow the developer toaccess underlying APIs that give the developer the freedom inwhat kind of games can be made. This means that the GDF can allow inexperienced developers to just modify simple APIs or example code to make variants of existing games, orto allow more experienced developers to make unique games by using more of the provided underlying APIs. How hard the GDF is to use will then really depend on the ambition ofthe game developer and not on the GDF itself. This can alsobe a motivating factor to learn more about the GDF's APIs.

2. Case Study: Applying a GDF in a Software Architercture Course

This section describes a case study of a software architecture course at the Norwegian University of Science and Technol-ogy (NTNU) where a GDF was introduced.

Received: 8 August Revised: 15 August Accepted: 20 August
Index in Cosmos
September 2020 Volume 10 ISSUE 3
UGC Approved Journal

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

The Software Architecture Course. The software architec-ture course is a postgraduate course offered to CS and SE students at NTNU. The course is taught every spring, its workload is 25% of one semester, and about 70 postgraduate students attend the course every semester. The students in the course are mostly of Norwegian students (about80%), but there are about 20% foreign students mostly from EU-countries. The textbook used in this course is the "Software Architecture in Practice, Second Edition", by Bass, Clements et al. [12]. Additional papers are used to

cover topics that are not sufficiently covered by the book such as design patterns, software architecture documentation standards, view models, and postmortem analysis [13–16].The education goal of the course is:

"The students should be able to *define and explain* centralconcepts in software architecture literature and be ableto *use and describe* design/architectural patterns, methods to design software architectures, methods/techniques to achieve software qualities, methods to document software architecture, and methods to evaluate software architecture."

The course is taught in four main ways:

(1) ordinary lectures given in English;

(2) invited guest lectures from the software industry;

(3) exercise in design patterns;

(4) a software development project with emphasis onsoftware architecture.

30% of the grade is based on an evaluation of a software architecture project that all students have to do, while 70% is given from the results of a written examination. The goalof the project is for the students to apply the methods and theory in the course to design a software architecture and to implement a system according to the architecture. The project consists of the following phases.

(1) COTS (Commercial Off-The-Shelf) exercise: learn the development platform to be used in the projectby developing some simple test applications.

(2) Design pattern: learn how to utilize design pattern by making changes in an existing system designed withand without design patterns.

(3) Requirements and architecture: describe the func- tional and the quality requirements, and design the software architecture for the application in the project.

(4) Architecture evaluation: use the Architecture Trade- off Analysis Method (ATAM) [12, 17] to evaluate the software architecture in regards to the quality requirements. Here one student group will evaluateanother student group's project.

(5) Implementation: do a detailed design and implement the application based on the created architecture and based on the results from a previous phase.

(6) Project evaluation: evaluate the project after it has been completed using a Post-Mortem Analysis (PMA) method.

In the two first phases of the project, the students workon their own or in pairs. For the phases 4–6, the students work in self-composed groups of four students. The students spend most time on the implementation phase (6 weeks), and they are also encouraged to start the implementation in earlier phases to test their architectural choices (incremental development). In previous years, the goal of the project hasbeen to develop a robot controller for a robot simulator in Java with emphasis on an assigned quality attribute such asavailability, performance, modifiability, or testability.

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

Choosing a GDF for the Software Architecture Course. InFall 2007, we started to look for appropriate GDFs to be used in the software architecture course in spring 2008. We lookedfor both GDFs where the programmer had to write the source code as well as visual drag-and-drop programming environments. The selection of candidates was based on GDFs we were familiar with and GDFs that had developer support. Further, we wanted to compare both commercial and open source GDFs. From an initial long list candidate GDFs, we chose to evaluate the following GDFs more in detail.

(i) *XNA:* XNA is a GDF from Microsoft that enablesdevelopment of homebrew cross-platform games for Win- dows and the XBOX 360 using the C# programming language. The initial version of Microsoft XNA Game Studio was released in 2006 [18], and in 2008 Microsoft XNA Gamestudio 3.0 was released that includes support for making games for XBOX Live. XNA features a set of high-level API enabling the development of advanced games in 2D or 3D with advanced graphical effects with little effort. The XNA platform is free and allows developers to create games for Windows, Xbox 360, and Zune using the same GDF [19]. XNA consists of an integrated development environment (IDE) along with several tools for managing audio and graphics.

(ii) *JGame:* JGame is a high-level framework for develop- ing 2D games in Java [20]. JGame is an open source projectand enables developers to develop games fast using few lines of code as JGame will take care of typical game functionalitysuch as sprite handling, collision detection, and tile handling. JGame games can be run as stand-alone Java games, Java applets games running in a web browser or on mobile devices (Java ME). JGame does not provide a separate IDE but is integrated with Eclipse.

(iii) *Flash:* Flash is a high-level framework for interactive applications including games developed by Adobe [21]. Most programming in Flash is carried out in Action script (a textual programming language), but the Flash environment also provides a powerful graphical editor for managing graphical objects and animation. Flash applications can run as stand-alone applications or in a web browser. Flash applications can run on many different operating systems like Windows, Mac OS X, and Linux as well as on mobile devices and game consoles (Nintendo Wii and Sony Playstation 3). Programming in Flash is partly visualby manipulating graphical objects, but most code is writtentextually. Flash supports development of both 2D and 3D applications.

(iv) *Scratch:* is a visual programming environment devel-oped by MIT Media Lab in collaboration with UCLA that makes it easy to create interactive stories, animations, games,music, and art and to share the creations on the web [22]. Scratch works similar to Alice [23] allowing you to programby placing sprites or objects on a screen and manipulate them by drag-and-drop programming. The main differencebetween Scratch and Alice is that Scratch is in 2D while Aliceis in 3D. Scratch provides its own graphical IDE that includesa set of programming primitives and functionality to importvarious multimedia objects.

An evaluation of the four GDF candidates is shown in Table 1. From the four candidates, we found Scratch to be the least appropriate candidate. The main disadvantage with Scratch was that it would be very difficult to teach software architecture using this GDF, as the framework did not allowexploring various software architectures. Further, Scratch was also very limited in what kind of games that could be produced, limiting the options for the students. The main advantage using Scratch is that it is very easy to learn and use. JGame suffered also from some of the same limitationsas Scratch, as it put some restrictions on what software architecture could be used, and it had little flexibility in producing a variety of types of games. The main advantage using JGame was that it was an open source project with access to the source code and that all the programming was done in Java. All CS and SE students at NTNU learnJava in the two first introductory programming courses. Anattractive alternative would be to use Flash as a GDF. Manydevelopers use Flash to create games for kids as well as gamesfor the Web. Flash puts little restrictions on what kind of games you can develop (both 2D and 3D), but there are some restrictions on what kind of software architecture you can use in your applications. The programming language used in Flash, Action Script, is not very different from Java so it should be rather easy for the students to learn. The main disadvantage using Flash in the software architecture course was the license costs. As the computer and information science department does not have a site license for the Flash development kit, it would be too expensive to use. XNA was found an attractive alternative for the students,as it made it possible for them to create their own XBOX 360 games. XNA puts little restrictions on what kinds of software architectures you apply in your software, and it enables the developers to create almost any game. XNA has strong support from its developer (Microsoft) and has a strong community of developers along with a lot of resources (graphics, examples, etc.). The main disadvantages using XNA as a GDF in the course were that the students had to learn C# and that the software could only run on Windows machines. Compared to JGame and other Java-based GDFs,XNA has a richer set of high-level APIs and a more mature architecture.

Based on the evaluation described above, we chose XNAas a GDF for our course. From previous experience we knewthat it does not require much effort and time to learn C# forstudents that already know Java.

Received: 8 August Revised: 15 August Accepted: 20 August
Index in Cosmos
September 2020 Volume 10 ISSUE 3
UGC Approved Journal
C

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

XQUEST—An Extension of the Chosen GDF. After we had decided to use XNA as a GDF in the software architecture course, we launched a project to extend XNA to make XNA even easier to use in the student project.This project implemented XQUEST (XNA QUick & Easy Starter Template) [24], which is a small and lightweight2D game library/game template developed at NTNU that contains convenient game components, helper classes, and other classes that can be used in the XNA game projects (see Figure 2). The goal of XQUEST was to identify andabstract common game programming tasks and create a set of components that could be used by students of the

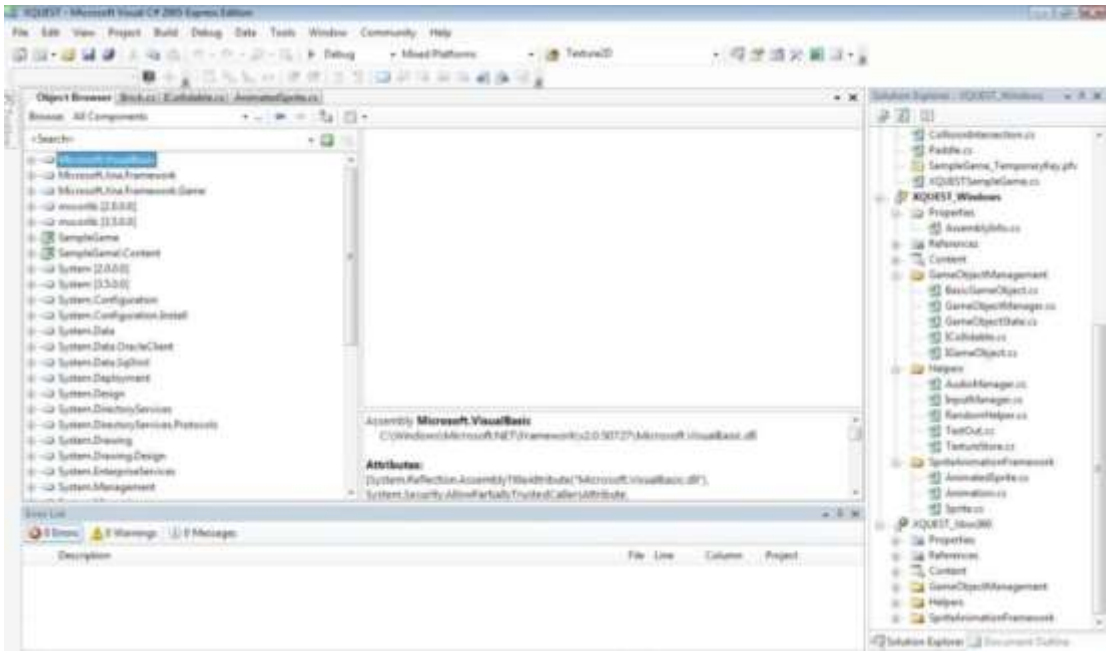


FIGURE 2: The XQUEST library shown in the XNA development environment.

Draw a helicopter sprite on the screen and make it move around on its own.

(1) Move around the helicopter sprite from previous task using the keyboard, change the size of the sprite when a key was pressed, rotate the sprite when another keywas pressed, and write the position of the sprite onthe screen.

(2) Animate the helicopter sprite using several frames and do sprite collision with other sprites.

(3) Create the classical Pong game in XNA.

Before the students started on their XNA introduction exercise, they got a two-hour technical introduction to XNA.During the semester, two technical assistants were assigned to help students with issues related to XNA. These assistantshad scheduled two hours per week to help students with problems, in addition to answering emails about XNA issues.

3.4.2. Requirement and Architecture for the Game Project. After the introduction exercise was delivered, the students formed groups of four students. Students that did not knew anyone were assigned to groups. The course staff then issued the project task where the goal was to makea functioning game using XNA based on students' own defined game concept. However, the game had to be designed and implemented according to their specified and designed software architecture. Further, the students had to develop a software architecture that focused on one particular quality attribute assigned by the course staff. We used the following definitions for the quality attributes in the game projects: *Modifiability*, the game architecture and implementation should be easy to change in order to add or modify functionality; *Testability*, the game architecture

and implementation should be easy to test in order to detect possible faults and failures. These two quality attributes were related to the course content and the textbook. A perfect implementation was not the ultimate quest of this XNA game project, but it was critical that the implementation reflectedthe architectural description. It was also important that thefinal

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

delivery was well structured, easy to read, and made according to the template provided by the course staff.

The first phase of the project was the requirement and architecture phase where the students should deliver requirements and the software architecture of the game along with a skeleton code reflecting the architecture. The requirements document focused on a complete functional requirement description of the game and several quality requirements for the game described as scenario focusing on one particular quality attribute. The architectural description was the most important part of the final delivery of the game project, andthe students had to document their architecture according to IEEE 1471-2000 [25]. The architecture documentation could be altered several times before its final delivery. Table 2 lists main attributes required in the architectural description in the game projects.

We also required that the students wrote the code skele- ton for the architecture they had designed. This was done to emphasize the importance of starting the implementation early and to ensure that students designed an architecture that was possible to implement.

3.4.3. Evaluation of the Game Project. After the requirements, the architecture and the code skeleton were delivered; the student groups were assigned to evaluate each other's architecture using ATAM. The whole idea was for one project group to evaluate the architecture of the other group's game

The workshop provided an open mind environment to let students give each other feedback, brainstorm about improvements and ideas, and to discuss their ideas to give a better understanding of the course content and game architecture design.

3.4.4. Post-Mortem Analysis. In the final task in the project, every group had to perform a post-mortem analysis of their project. The focus of the PMA was to analyze successes and problems of the project. The PMA was documented in a short report that included a positive (successes) and a negative (problems) KJ diagram (struc- tured brainstorm map), a positive and a negative causal map (a diagram that shows cause-effect relationships), and experiences from using PMA [13]. The PMA made the students reflect on their performance in the project and gave them useful feedback to improve in the future projectsand inputs for the course staff to improve the course. The main topics analyzed in the PMA were issues related to group dynamics, time management, technical issues,software architecture issues, project constraints, and personal conflicts.

3. Experiences of using GDF inSoftware Architecture

The experiences described in this section are based on the final course evaluation, feedback from the students during the project, and the project reports.

The final course evaluation made all students (manda- tory) taking the course answer three questions. The results reported below are a summary of the students' responses related to the project and the GDF.

(1) What have been good about software architecture course?

(a) *About the project itself:* "Cool project", "Really inter-esting project", "We had a lot of fun during the project", "It is cool to make a game", "Fun to imple- ment something practical such a game", "Videogame as an exercise is quite interesting", "I really liked theproject", and "The game was motivating and fun".

(b) *Project and learning:* "Good architectural discussion in the project group I was in", "Learned a lot about software architecture during the project", "The project helped to understand better the arguments explained in the lectures, having fun at the mean- time", "Fun project where we learned a lot", "I think that the creation of a project from the beginning, withthe documentation until the code implementation, was very helpful to better understand in practice the focus of the course", "The game project was tightly connected to the syllabus and lectures and gave valuable experience. The main thing I learned was probably how much simpler everything gets if you have a good architecture as a basis for your system", and "The interplay of game and architectural approaches".

Received: 8 August Revised: 15 August Accepted: 20 August
Index in Cosmos
September 2020 Volume 10 ISSUE 3
UGC Approved Journal
C

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

(c) *The project being practical work:* "I think it was pretty good that you guys made us do a lot of practical work", and "To choose C# as a platform is a good idea as it is used a lot in the software industry; at the same time it is very similar to Java so it is rather easy to learn the language."

(d) *Interplay between groups:* "It was also good to see theresults of the others' projects in the final presenta- tion".

(2) What have been not so good about the coursesoftware architecture?

(a) *XNA support:* "The way the student assistants were organized; during the implementation periods at least they should be available in a computer lab and not just in the classroom", "Maybe the use of XNAFramework XQUEST was very difficult because I never use it. Maybe some extra lecture focus on the use of XQUEST Framework was better", and "We didnot have lectures on XNA; could have got some morebasic info.. .Hmm .. ."

(b) *XNA versus software architecture:* "Took a lot of time getting to know C# , I liked it, but I did not have the time to study architecture" and "The use of game as a project may have removed some of the focus away from the architecture. XNA and games in general limit the range of useful architectures."

(3) What would you have changed for next year's course?

(a) *Project workload:* "Maybe just little more time to develop the game" and "I would change the impor- tance of the project. I think that the workload of theproject was very big and it can matter the 50% of thetotal exam."

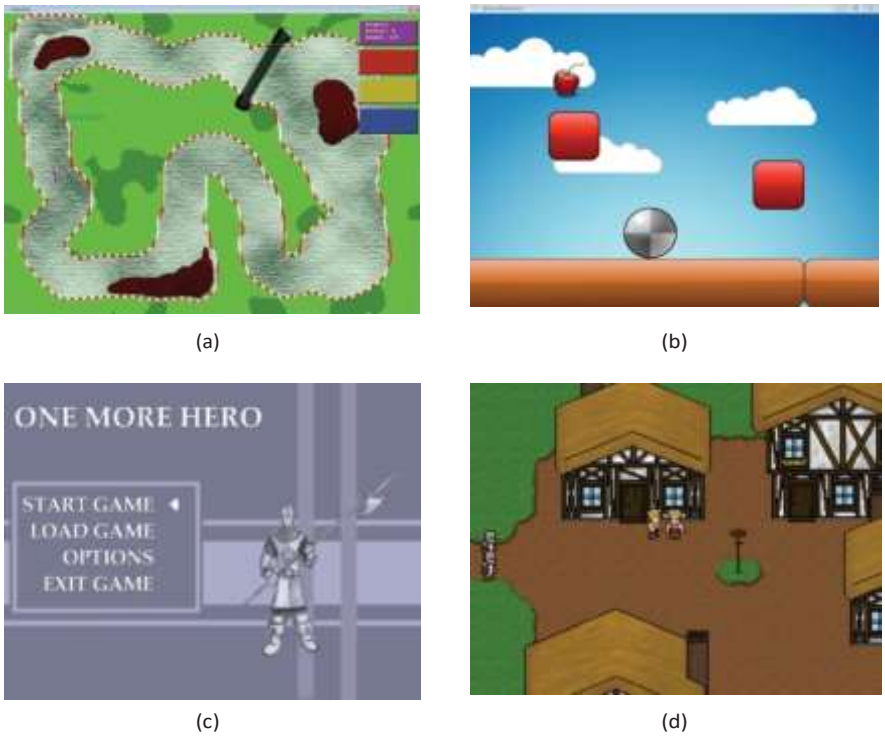
(b) *XNA support:* "Perhaps have some C# intro?" and "Itwould be helpful to have some lab hours".

(c) *Project constraints:* "Maybe more restrictions on game type, to ensure that the groups choose games suited for architectural experimentation."

The responses from the students were overall very positive. In the previous years, the students in the software architecture course had to design the architecture and implement a robot controller for a robot simulator in Java. The feedback from the XNA project was much more positivethan the feedback from the robot controller project. Other positive feedback we got from the students was that they felt they learned a lot from the game project, that they liked thepractical approach of the project and having to learn C#, and the interaction between the groups (both ATAM and the project workshop).

The negative feedback from the course evaluation was focusing on the lack of XNA support and technical support during the project and that some student felt that there was too much focus on C#, XNA, and games and too little on software architecture.

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86



(a) (b)

(c) (d)

FIGURE 3: Game based on XNA framework (Top left: Racing; Top right: Codename Gordon; Bottom: RPG).

The suggestions to improve the course were mainly according to the negative feedback, namely, to improve XNAsupport and to adjust the workload of the project. One student also suggested limiting the types of games to be implemented in project to ensure more focus on software architectural experimentation.

3.1. Snapshots from Some Student Projects. Figure 3 shows screenshots from four student game projects. The game at the upper left corner is a racing game, the game at the upper rightcorner is a platform game, and the two games below are role-playing games (RPGs). Some of the XNA games developed were original and interesting. Most games were entertaining but were lacking contents and more than one level due to time constraints.

4. Related Work

This paper describes experiences from utilizing the special features of a GDF in a software architecture course. The main benefits from applying a GDF in a CS or SE course is that the students get more motivated during the software development project. As far as we know, there are few papers that describe the usage of a professional GDF concept applied in universities courses that is not directly a target for learning game development, especially no papers aboutusage of XNA in higher education. However, there are some related approaches in education described in this section.

El-Nasr and Smith describes how the use of modifying or *modding* existing games can be used to learn computer

Received: 8 August Revised: 15 August Accepted: 20 August
Index in Cosmos
September 2020 Volume 10 ISSUE 3
UGC Approved Journal
C

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

science, mathematics, physics, and ascetic principles [10]. The paper describes how they used modding of the WarCraft III engine to teach high school students a class on game design and programming. Further, they describe experi- ences from teaching university students a more advanced class on game design and programming using the Unreal Tournament 2003 engine. Finally, they present observations from student projects that involve modding of game engines. Although the paper claims to teach students other things than pure game design and programming, the GDFs were used in the context of game development courses.

The framework Minueto [28] is implemented in Java, and it is used by students in their second year of undergraduate studies at McGill University in Montreal, Canada. The framework encapsulates graphics, audio, and keyboard/mouse inputs to simplify Java game development.It allows development of 2D games, such as card games and strategy games, but it lacks in support for visual programming and suffers from limited documentation.

The Labyrinth [29] is implemented in Java, and it is a flexible and easy-to-use computer game framework. The framework enables instructors to expose students to very specific aspects of computer science courses. The framework is a finished game in the Pac-Man genre, highly modular, and it lets the students change different aspects of the game. However, it cannot be used to develop different genres of game, and there is little room for changing the software architecture of the framework.

The JIG (Java Instructional Gaming) Project [30] is a collaborative effort between Scott Wallace (Washington State University Vancouver) and Andrew Nierman (University of

Puget Sound) in conjunction with a small group of dedicatedstudents. It has three aims: (1) to build a Java Instructional Game Engine suitable for a wide variety of students at all levels in the curriculum; (2) to create a set of educational resources to support the use of the game engine at small, resource-limited, schools; (3) to develop a community of educators that use and help improve these resources. The JIG Project was proposed in 2006, after a survey of existinggame engines revealed a very limited supply of existing 2D Java game engines. JIG is still in development.

GarageGames [31] offers two game engines written in C++. The Torque Game Engine targets 3D games, while the Game Builder provides a 2D API and encourages programmers to develop using a proprietary language (C++ can also be used). Both engines are aimed at a wide audience, including students and professionals. The engines are available under separate licenses ($50 per license per year for each engine) that allow full access to the source code. Documentation and tutorials cover topics appropriate for beginners and advanced users.

The University of Michigan's DXFramework [32] game engine is written in C++. The current version is targeted specifically for 2D games, although previous versions have included a 3D API as well. This engine is designed for game programming education and is in its third major iteration. The DXFramework is an open source project. Compared to XNA, DXFramework has no competitive advantage as it haslimited support for visual programming, and it is not easierthan XNA to learn.

The University of North Texas's SAGE [33] game engine is written in C++, and targets 3D games, not 2D. Like the DXFramework, SAGE is targeted specifically for game programming educational usage. The source code can be downloaded and is currently available without license.

Marist College's GEDI [34] game engine provides a second alternative for 2D game design in C++, and is also designed with game programming educational use in mind. Source code can be downloaded and is currently available without license, but GEDI is still in the early phases of development. Only one example game is distributed with the code, and little documentation is available.

For business teaching, Arena3D [35] is a game visu- alization framework with its animated 3D representations of the work environments; it simulates patients queuing at the front desk and interacts with the staff. IBM has also produced a business game called INNOV8 [36] which is "an interactive, 3-D business simulator designed to teach the fundamentals of business process management and bridge the gap in understanding between business leaders and IT teams in an organization".

5. Conclusion and Future Work

In this paper we have presented a case study of how a GDF was evaluated, chosen, and integrated with a software architecture course. The main goal of introducing a GDF and a game development project in this course was to motivate students to learn more about software architecture during the game development project. The positive feedback from

the students indicates that this was a good choice as the students really enjoyed the project and learned software architecture from carrying out the project.

We will continue to explore the area of using games, games concept, and game development in CS and SE edu- cation and evaluate how this affects the students' motivation and performance. The choice of XNA as a GDF proved tobe a good choice for our software architecture course. The main disadvantage using XNA is the lack of support for non- Windows operating systems like Linux and Mac OS X. Mono. XNA is a cross platform implementation of the XNA game framework that allows XNA to run on Windows, Mac OSX, and Linux using OpenGL [37]. The project is still in an early phase. An alternative to solve this problem is to let thestudents choose between different GDFs, for example, XNAand a Java-based GDF. The main challenge for this approachis that the course staff needs to know all the GDFs offered to the students to give proper technical assistance. Based onthe feedback from the students; the technical support is very important and must be considered before providing choicesof more GDFs.

Acknowledgments

The authors would like to thank Jan-Erik Strøm and Trond Blomholm Kvamme for implementing XQUEST and for their inputs to this paper. They would also like to thank Richard Taylor and Institute for Software Research (ISR)at University of California, Irvine (UCI), for providing a stimulating research environment and for hosting a visitingresearcher.

References

R. Rosas, M. Nussbaum, P. Cumsille, et al., "Beyond Nintendo: design and assessment of educational video games for first and second grade students," *Computers & Education*, vol. 40, no. 1, pp. 71–94, 2003.

M. Sharples, "The design of personal mobile technologies forlifelong learning," *Computers & Education*, vol. 34, no. 3-4, pp. 177–193, 2000.

A. Baker, E. O. Navarro, and A. van der Hoek, "Problemsand programmers: an educational software engineering cardgame," in *Proceedings of the 25th International Conference on Software Engineering (ICSE '03)*, pp. 614–619, Portland, Ore, USA, May 2003.

L. Natvig, S. Line, and A. Djupdal, ""Age of computers"; an innovative combination of history and computer game elements for teaching computer fundamentals," in *Proceedings of the 34th Annual Frontiers in Education (FIE '04)*, vol. 3, pp. 1–6, Savannah, Ga, USA, October 2004.

E. O. Navarro and A. van der Hoek, "SimSE: an educational simulation game for teaching the software engineering pro- cess," in *Proceedings of the 9th Annual SIGCSE Conferenceon Innovation and Technology in Computer Science Education(ITiCSE '04)*, p. 233, Leeds, UK, June 2004.

G. Sindre, L. Natvig, and M. Jahre, "Experimental validation of the learning effect for a pedagogical game on computer fundamentals," *IEEE Transactions on Education*, vol. 52, no. 1,pp. 10–18, 2009.

B. A. Foss and T. I. Eikaas, "Game play in engineering education: concept and experimental results," *International Journal of Engineering Education*, vol. 22, no. 5, pp. 1043–1052,2006.

A. I. Wang, O. K. Mørch-Storstein, and T. Øfsdahl, "Lecture quiz—a mobile game concept for lectures," in *Proceedings ofthe 11th IASTED International Conference on Software Engi- neering and Application (SEA '07)*, pp. 305–310, Cambridge, Mass, USA, November 2007.

A. I. Wang, T. Øfsdahl, and O. K. Mørch-Storstein, "An eval- uation of a mobile game concept for lectures," in *Proceedingsof the 21st Conference on Software Engineering Education and Training (CSEET '08)*, pp. 197–204, Charleston, SC, USA, April 2008.

M. S. El-Nasr and B. K. Smith, "Learning through game modding," *Computers in Entertainment*, vol. 4, no. 1, pp. 45–64, 2006.

T. W. Malone, "What makes things fun to learn? Heuristics for designing instructional computer games," in *Proceedings of the 3rd ACM SIGSMALL Symposium and the First SIGPC Symposium on Small Systems (SIGSMALL '80)*, pp. 162–169, ACM Press, Palo Alto, Calif, USA, September 1980.

P. Clements, L. Bass, and R. Kazman, *Software Architecture in Practice*, Addison-Wesley, Reading, Mass, USA, 2nd edition, 2003.

A. I. Wang and T. Stålhane, "Using post mortem analysis to evaluate software architecture student projects," in *Proceedingsof the 18th Conference on Software Engineering Education and Training (CSEET '05)*, pp. 43–50, Ottawa, Canada, April 2005.

J. O. Coplien, "Software design patterns: common questions and answers," in *The Patterns Handbook: Techniques, Strategies,and Applications*, pp. 311–320, Cambridge University Press, New York, NY, USA, 1998.

A. Rollings and D. Morris, *Game Architecture and Design:A New Edition*, New Riders Games, Indianapolis, Ind, USA, 2003.

D. P. Perry and A. L. Wolf, "Foundations for the study of software architecture," *ACM Sigsoft Software Engineering Notes*,

Received: 8 August Revised: 15 August Accepted: 20 August

vol. 17, no. 4, pp. 40–52, 1992.
R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere, "The architecture tradeoff analysis method," in *Proceedings of the 4th IEEE International Conference on Engineering Complex Computer Systems (ICECCS '98)*, pp. 68– 78, Monterey, Calif, USA, August 1998.
Microsoft Corporation, "XNA Developer Center," June 2008,http://msdn.microsoft.com/en-us/xna/aa937794.aspx.
B. Nitschke, *Professional XNA Game Programming: For Xbox360 and Windows*, John Wiley & Sons, New York, NY, USA, 2007.
JGame project, "JGame: a Java game engine for 2D games," November 2008, http://www.13thmonkey.org/~boris/jgame/index.html.
Adobe, "Animation software, multimedia software—Adobe Flash CS4 Professional," November 2008, http://www.adobe.com/products/flash.
Lifelong Kindergarten Group, MIT Media Lab, "Scratch: Imagine, Program, Share," June 2008, http://scratch.mit.edu.
Carnegie Mellon University, "Alice.org," June 2008, http:// www.alice.org.
T. Blomholm Kvamme and J.-E. Strøm, *Evaluation and extension of an XNA game library used in software architecture projects*, M.S. thesis, Department of Computer and Informa- tion Science, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, June 2008.
IEEE Std 1471-2000, "IEEE Recommended Practice for Archi- tectural Description of Software-Intensive Systems," Software Engineering Standards Committee of the IEEE Computer Society, 2000.
P. Kruchten, "The 4 + 1 view model of architecture," *IEEE Software*, vol. 12, no. 6, pp. 42–50, 1995.
A. BinSubaih and S. C. Maddock, "Using ATAM to evaluate a game-based architecture," in *Proceedings of the 2nd International ECOOP Workshop on Architecture-Centric Evolution (ECOOP '06)*, Nantes, France, July 2006.
A. D. Minueto, *An undergraduate teaching development framework*, M.S. thesis, School of Computer Science, McGill University, Montreal, Canada, 2005.
J. Distasio and T. Way, "Inclusive computer science education using a ready-made computer game framework," in *Proceed-ings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '07)*, pp. 116–120, Dundee, Scotland, June 2007.
Washington State University Vancouver and University of Puget Sound, "The Java Instructional Gaming Project," June 2000, http://ai.vancouver.wsu.edu/jig.
GarageGames, "GarageGames," June 2008, http://www.garagegames.com.
C. Johnson and J. Voigt, "DXFramework," June 2008, http://www.dxframework.org.
I. Parberry, "SAGE: a simple academic game engine," June 2008, http://larc.csci.unt.edu/sage.
R. Coleman, S. Roebke, and L. Grayson, "GEDI: a game engine for teaching videogame design and programming," *Journal of Computing Science in Colleges*, vol. 21, no. 2, pp. 72–82, 2005.
Rockwell Automation Inc, "Arena Simulation Software," June2008, http://www.arenasimulation.com.
IBM, "INNOV8—a BPM Simulator," June 2008, http://www.ibm.com/software/solutions/soa/innov8.html.
Monoxna, "Monoxna—Google Code," November 2008,http://code.google.com/p/monoxna.

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

Received: 8 August Revised: 15 August Accepted: 20 August
Index in Cosmos
September 2020 Volume 10 ISSUE 3
UGC Approved Journal
C