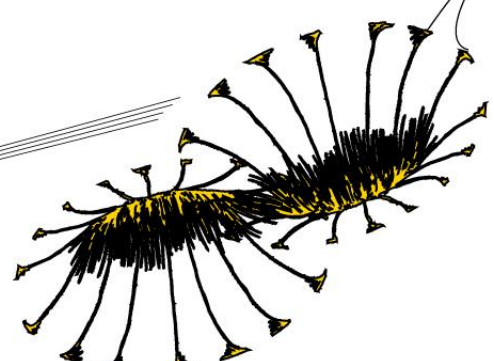



# ***Empirical study of the effects of software reuse in videogames on game and project performance***



**Author:** ing. Paul van den Bosch, S1243667

**Supervisors:** Dr. ir. Erwin Hofman, Dr. ir. Klaasjan Visscher

**Key words:** Software reuse, Game engine, Game Components,  
Game Assets, Game development, Quality.

## **Abstract**

In this paper we represent the results of an empirical study of the effects of software reuse on game performance outcomes (review scores) and project performance outcomes (cost efficiency, development time efficiency, quality, profitability). This study started with an extensive literature review on software reuse and product modularity followed by an exploratory study through several interviews with game and software developers. The results of the literature review and exploratory study was then used to formulate a research model, hypothesis and survey questionnaire. A random sample of 124 games were targeted for this study that were published during the period 2009-2014 on pc and consoles. The period of study is between February 2014 and June 2015. We framed our research mainly around the reuse of Game Components and Game Assets. All game engines contain a familiar set of core components, including the artificial intelligence components, rendering components, physics components, animation components, visual effects components, audio components and the Game specific subsystems. Game Assets are a collection of data files such as models, textures, sound and animation data which support gameplay and are used as input data for the Game engine. The term "game engine" refers to software that is extensible and can be used as the foundation for many different games.

In Game development, Game Components and Game Assets are more easy identifiable across different games over general known software abstractions such as lines of code, number of classes, modules, procedures, functions and prove to be distinct parts that together make up a video game. These specific software parts, often functioning as executable units of independent production, acquisition and deployment can be composed into a functioning game system. It is therefore that investigating these domains typically seen in game development should give game developers more practical value over the generally known software abstractions to further base their game engine-technology and component and assets sourcing strategies upon. To our knowledge no other study has previously reported how reuse impacts game scores and software development economics.

Our findings show that there are significant statistical correlations between the factors of software reuse and project and game performance outcomes. Significant statistical correlations were found between the different Overall reuse and Specific reuse factors and project and game performances outcomes. In our study we found a statistically significant positive correlation between Overall degree of component and Cost efficiency. Looking at the specific components level, Rendering components show a statistically significant positive correlation with Cost efficiency. The Animation components shows both a statistical positive correlation with Cost efficiency and Development time efficiency. This study also found a significant negative correlation between the Overall degree of software reuse and Graphics, Sound & Music, Story and presentation and the Overall game score.

The Overall degree of external reuse correlated negatively with profitability and quality, however this relationship was not significant. The Overall degree of External reuse also correlated negatively with all Review score criteria and Gameplay score showing a statistically significant negative relationship.

Direct implications for game developers and game directors are that they not only need to pay attention the specific game components and game assets they are reusing but

also to the degree they are reusing it. Managers need to find a certain balance in the levels of reuse, as too much reuse can negatively affect game performance outcomes. Our results showed for example for Asset Reuse that as the degree of Overall Game Assets reuse increases, Overall Game score increases but only up to a certain point where we can see that as the degree of Overall Game Assets reuse increases, Overall Game score decreases. The same is concluded for the Specific assets and Game score. The study also shows there is a significant difference between the Low and High reuse modes of Overall components and Project Performance variables. Firms applying a high mode of Overall component reuse scored better on Costs efficiency and Development time efficiency.

Other variables such as whether a Middleware / Internal game engine was used and having a small / large team size reported the same level of effects on the amount of Reuse and the Product and Project performance outcomes and these categories were not statistically different from each other. Firms employing a high level of Systematic reuse process resulted in significant differences in game performance outcomes. The study results show that a Low/high systematic reuse process differ on Game Scores with a Low systematic reuse process scoring higher than a High systematic reuse process on AI, Gameplay, Graphics, Personal Slant, Sound & Music, Story & Presentation and Overall game score.

In sum, game developers, Technical- and Art directors should consider our study results and analyze and compare their own specific reuse choices and the effects on development time and development costs and Game scores. By implementing a systematic reuse process they can potentially achieve substantial benefits in Cost efficiency but they must keep in mind that although a high level of Systematic reuse is positively related to better Cost efficiency it does not necessary result in a 'good' game. Game developers should therefore find a balance in where and where not to follow a systematic reuse process in the different stages of game development and game design which should be further integrated in the firm's software development process.

Also, in addition to a systematic reuse process, component reuse can help to achieve higher levels of Cost efficiency. We did not found any statistically significant positive correlations between the Overall degree of software reuse and the four project performance variables Cost efficiency, Development time efficiency, Quality and Profitability. The result imply that applying a systematic component strategy that includes a high level of reuse of the Rendering components, Animation components and Game specific subsystems can help firms to achieve higher levels of Cost efficiency. Furthermore, as the reuse of Game specific components was significantly negatively correlated with review scores it underscores the importance and need of tailoring Game specific components to the game to achieve higher review scores.

Understanding these reuse effects on project and game performance could help game developers to put emphasize on the right management efforts and financial resources in the different stages of software development and game design. E.g. it can help whether it is worth investing in particular game components or new development methodologies in system and game design to improve development time efficiency, cost efficiency or game quality. While this study hypothesized and found several associations between the variables of software reuse and project- and game performance variables, results need to be interpreted cautious due to small sample size and therefor a larger confirmatory study is needed. A larger sample size provides more precise results.

## **Acknowledgements**

My dad used to program simple video games for my brother and me when we were kids. This triggered my interest and passion for creating computer software. In 2007, while still in high-school my twin-brother and I started our first Internet company specialized in creating websites, webshops and e-commerce software. It is therefore no coincidence that the topic for my master thesis had to be about computer software.

As an Internet Entrepreneur I am always interested in developing successful software products more easier and faster for our potential customers. In my free time not developing new Internet services or other software I love to play a video game on my PC, Playstation 3, Playstation 4, Xbox 360 or Nintendo Wii as well.

When my research supervisor Dr. ir. Erwin Hofman opted the idea for a research topic about software reuse in the gaming industry this immediately caught my interest. This had to be the research subject where I wanted to spent my master thesis on.

I owe my thanks to Dr. ir. Erwin Hofman for this research topic and mentoring me during my research. He always found the time for me to provide constructive feedback and recommendations on my research study. There were never problems in making appointments or contacting him and I got to know him as a very open, calm and at the same time very enthusiastic person.

I also want to thank Dr. ir. Klaasjan Visscher for his suggestions and recommendations for improvements on my research and survey questionnaire. I feel that his contributions and remarks really improved the quality of this study.

Lastly, I want to thank my loving parents Jan van den Bosch and Annelies Kooi for all their support during my pre-university Education, Higher Vocational Education and Academic years. I am very thankful for their infinite support for all those many, many years and that they have given me the opportunity to achieve my educational goals. I feel that without them, I would not be even close to where I am today.

Ing. Paul van den Bosch

Enschede,

07 September 2015

## 1. Introduction:

The Gaming industry has become serious business. According to Gartner, a leading information technology research and advisory company, the global video game marketplace will see an annual growth rate of 18.3% in 2013 and will reach \$128 billion by 2017, up from \$79 billion in 2012 ("Forecast: Video Game Ecosystem, Worldwide, 4Q13," 2015).

Producing large video games can take years of development, with large teams consisting of hundreds of people on board producing the game, even spanning across multiple studios. Developing a game for the Nintendo Wii can cost \$5 million to \$10 million. Games for Xbox 360 and PS3 on average cost between USD 20 million and USD 50 million to develop ("Will the Wii be a set-top box? - CNET," 2007).

Due to significant advances in hardware games have evolved in scale and complexity. As a result AAA games<sup>1</sup> grow more complex and larger in scale by the year and the costs for these games have increased tremendously. It is expected that development costs for the current generation games (PS4, Xbox one) on average may exceed \$60 million ("Games to cost \$60m, says Ubisoft boss - Eurogamer.net," 2009). As technology advances and consumers demand the latest features, games will be required to continue to grow in terms of size and their complexity. In order for development houses to keep costs acceptable, certain realities must be faced: Games can no longer be coded entirely from scratch. To manage development productivity effectively it's important to research strategies to lower development costs and shorten time to market, and analyzing their effects on project- and product performance in the context of the gaming industry.

Software reuse can help organizations to lower development costs and improve software quality. Considerable research has been directed at how software reuse influences productivity, quality and IT project performance (V.R. Basili, Briand, & Melo, 1996); (de O. Melo, S. Cruzes, Kon, & Conradi, 2013); (W. Frakes & Terry, 1996); (Ajila & Wu, 2007).

Software reuse is the systematic use of existing software assets to construct new or modified software or products (Mohagheghi & Conradi, 2007, p. 472). Software assets in this view may be source code or executables, design templates, free standing Commercial-Off-The-Shelf (COTS) or Open Source Software (OSS) components, or entire software architectures and their components forming a product line or product family. The major motivation for reusing software artifacts is to decrease software development costs and cycle time by reducing the time and human effort required to build software products. Some research suggests that software quality can be improved by reusing quality software artifacts. Some work has also stated that software reuse is an important factor in reducing maintenance costs because, when reusing quality objects, the time and effort required to maintain software products can be reduced (V.R. Basili, 1990).

For these reasons the reuse of software products, software processes, and other software artifacts is often considered the technological key to enabling the software industry to achieve required levels of productivity and quality (V.R. Basili & Rombach, 1988).

Copying some source files onto a project, calling an API function, or instantiating a class written by someone else, are all different forms of code reuse. A way to reduce the development costs of games in particular is to reuse specific Game Components or Game Assets in the game. A Game Component can be for example an AI-, Animation-, or Rendering component that is used by the game. These Game Components are generally designed for composability and enable the easy assembly and upgrading of systems out of independent developed pieces of software (Mohagheghi & Conradi, 2007, p. 472). Game Assets are designed for generality and allow cost reduction through reuse of previously developed assets in the development of new games. They usually come in the form of a collection of data files such as models, textures, sounds, animations and

---

<sup>1</sup> Pronounced as "triple A" games, these are highly expected big budget games with high levels of promotion. E.g. GTA 5 with 52 million copies sold and an estimated development and marketing budget of \$250 million.

support gameplay. Adopting complete licensed 3D engines, Commercial Off The Shelf (COTS) components (or Middleware Components) and Game Assets have become widely preferred approaches over proprietary technology to simplify and shorten the development process, potentially leading to better quality, productivity and a shorter time-to-market (Rollings & Morris, 2004).

In this paper, we examine the concepts of software reuse in the context of the video game industry and we frame our investigation around the reuse of Game Components and Game Assets. The aim of this study is to analyze the effects of game Component and game Asset reuse on different game performance outcomes (review scores) and project performance outcomes (cost efficiency, development time efficiency, quality and profitability).

We therefore state the following research question:

RQ: What are the effects of software reuse on game and project performance?

In order to answer this question we investigated empirically whether:

1. The degree of Components reuse has effect on review scores and project performance outcomes (cost efficiency, development time efficiency, quality and profitability).
2. The degree of Assets reuse has effect on review scores and project performance outcomes (cost efficiency, development time efficiency, quality and profitability).
3. There is a difference between internal and external software reuse (via COTS or open-source) on review scores and project performance outcomes.
4. A systematic reuse process affects the degree of software reuse and project performance outcomes.

While there are many articles and various books covering game development aspects such as programming (Rollings & Morris, 2004); (Gregory, 2009); (Schmidt, Crnkovic, & Heineman, 2007) project-management and game design (McGuire & Chadwicke Jenkins, 2008) we found little empirical studies and conclusions about the application of software reuse within the context of the gaming industry and the effect on product- and project performance. This paper aims to fill this gap by exploring facets of software reuse commonly seen in game development and analyzing their effects on different game performance outcomes and project performance outcomes.

To our knowledge no other study has previously reported how reuse impacts game scores and software development economics. This study could be highly beneficial to different entities like independent developers, large game development studios or game publishers that are considering making a game or are in the process of making a game. Additionally it aims to further expand on current literature such as the works (V.R. Basili et al., 1996); (W. Frakes & Terry, 1996); (Ajila & Wu, 2007) about the effects of software reuse on productivity and quality which will further grow our academic knowledge on this subject. Understanding these reuse choices and their effect on product performance could help game developers to put emphasis on the right management efforts and financial resources in the different stages of software development.

This study is based on an extensive literature review within the fields of software reuse and software quality. In addition we also conducted interviews with game and software developers to develop a model for our study and identifying variables. A survey questionnaire was eventually developed and analyzed which formed the basis of our empirical study in the context of the Gaming industry.

Our findings show that there are strong significant statistical correlations between the factors of software reuse and project and game performance outcomes. The study also shows there is a significant difference between the Low and High reuse modes of Overall components and Project Performance variables. The study results imply that game developers and game directors not only need to pay attention the specific components and assets they are reusing but also to the degree they are reusing it. Furthermore, by implementing a systematic reuse process they can achieve substantial benefits in Cost efficiency but they must keep in mind that although a high level of Systematic reuse is positively related to better Cost efficiency it does not necessary result in a 'good' game.

The remainder of this paper is organized as follows:

Section 2 and 3 introduces the theoretical framework, reviews relevant literature and introduces our hypotheses and illustrates the research constructs. Section 4 presents our research methodology, sample and measures. Section 5 presents results from the conducted survey on video game development. Section 6 contains an overview and discussion of our findings and provides suggestions for further research.

## 2. Theoretical Background and hypotheses

In the paragraphs below literature around software reuse, areas of reuse, reuse strategies and reuse in computer games is reviewed. Hypotheses are then developed how software reuse in video games affects project and game performance.

### 2.1 Definition of Software reuse

The following generic definition has been adopted from (Biggerstaff & Perlis, 1989): "The reuse of software is renewed use of artifacts and collected knowledge arising from the development of a software system when developing a new software system, in order to reduce the expenditure for creating and maintaining this new system."

Another definition for software reuse as a whole, i.e. for the reuse process, is provided by (Ezran, Morisio, & Tully, 2002): "Software reuse is the systematic practice of developing software from a stock of building blocks, so that similarities in requirements and/or architecture between applications can be exploited to achieve substantial benefits in productivity, quality and business performance."

### 2.2 Reuse artifacts

Reuse is a very broad term covering the general concept of a reusable asset.

An *asset* can be any *artifact* that is used in the development and maintenance of software (Schach, 2011, p. 3). Software assets may be source code or executables, design templates, free standing Commercial-Off-The-Shelf (COTS) or Open Source Software (OSS) components, or entire software architectures and their components forming a product line or product family.

Software reuse can apply to any life cycle product, such as documents, system specifications, design structures, and any other development artifacts not just fragments of software code (Barns & Bollinger, 1991).

In Ajila & Wu (2007) the authors explain that reuse can also occur in many levels of granularity which could be a few lines of code, methods, component, classes or whole systems (Ajila & Wu, 2007). Table 1 lists a set of assets that can be reused across software projects.

Intermediate artefact	Implemented artefact	Project management and quality assurance artifacts
Requirements	(sub)systems	Process models
Architectures	Frameworks, components, modules, package	Planning models
Designs	UML models, interfaces, patterns	Cost models
Algorithms	Libraries	Review and inspection forms
Documentation	Test cases	Analysis models
Program code	Classes, procedures, routines, functions, methods, source code, data	Design & coding conventions

**Table 1. A variety of reusable assets (adopted from Biggerstaff 1983).**



Freeman (1993) also identified and classified different types of reusable artifacts:

- *Code fragments*, which come in a form of source code, PDL, or various charts;
- *Logical program structures*, such as modules, interfaces, or data structures;
- *Functional structures*, e.g. specifications of functions and their collections;
- *Domain knowledge*, i.e. scientific laws, models of knowledge domains;
- *Knowledge of development process*, in a form of life-cycle models;
- *Environment-level information*, e.g. experiential data or users feedback;
- *Artefact transformation* during development process.

Technical approaches to reuse mentioned in literature include:

- Compositional; reuse of functions or subroutines (fine-grained);
- Reuse of templates which can be of any kind;
- Reuse of software modules or components;
- Object-Oriented (OO) frameworks;
- Domain engineering for product families;
- Component-based with adherence to component models such as CORBA/CCM/EJB;
- Generative programming;
- Reuse repository or library, which can be generic or domain-specific, and can be combined with other approaches.

### **2.3 Software reuse and the effect on software development economics**

The main motivations found in literature for reusing software artifacts is the potential of increased software *quality* and *productivity* in software development and lower maintenance costs (V.R. Basili et al., 1996); (de O. Melo et al., 2013); (W. Frakes & Terry, 1996); (Ajila & Wu, 2007).

Research shows that there is positive and significant evidence on lower problem density (defect-, error- or fault density) and effort spent on corrections (rework effort) with introducing systematic reuse of quality software artifacts.

Because work products are used multiple times, the accumulated defect fixes results in a higher quality work product (Lim, 1994). Reused components may be designed more thoroughly and be better tested, since faults in these components affect several products and the prevention costs are amortized over several products (Mohagheghi, Conradi, Killi, & Schwarz, 2004). Research also indicates that Rework effort is significantly reduced with systematic reuse. In (Selby, 2005), rework effort is lowest for modules reused verbatim and small in size. The difference is also significant for modules with slight revision. In component-comparison studies, systematic reuse (either verbatim, with slight modification or mixed with new code) is related to significant decrease in problem density in four studies (Lim, 1994); (Mohagheghi et al., 2004); (Selby, 2005); (Thomas, Delis, & Basili, 1997).

There is positive and significant evidence on apparent productivity gains in small and medium-scale studies. Apparent productivity improves significantly with systematic reuse (Lim, 1994); (Morisio, Romano, & Stamelos, 2002); (Baldassarre, Bianchi, Caivano, & Visaggio, 2005) and the positive relation with reuse rate is reported in (Lim, 1994). Because the work products have already been created, tested and documented, apparent productivity will increase. However, increased productivity does not necessarily shorten time-to-market because reuse must be used effectively on the critical path of a development project (Lim, 1994).

Additionally, the study of (V.R. Basili et al., 1996, p. 115) offers significant results showing the strong impact of reuse on productivity and product quality, or defect density and rework density, in the context of Object Oriented systems. Some work has also stated that software reuse is an important factor in reducing maintenance costs because, when reusing quality objects, the time and effort required to maintain software products can be reduced (Victor R. Basili, 1990).

## 2.4 Reuse types

On average, only about 15 percent of any software product serves a truly original purpose. The other 85 percent of the product in theory could be standardized and reused in future products (Jones, 1984, p. 1). Software reuse appears in two major forms: *systematic* and *ad-hoc*. Implementing systematic reuse within a company can be expensive as it takes time to specify, design, implement, test, and document a software component. Ad-hoc reuse means that reuse is opportunistic and not part of a repeatable process, as opposed to systematic reuse; meaning planned reuse. Verbatim reuse means reusing an asset "as-is" in a black-box style; or modified in a white-box style to make an asset reusable for a new target. In Frakes & Terry (1996) different types of reuse have been identified as shown in table 2.

Type of reuse	Description
Ad-hoc	Refers to the selection of components which are not designed for reuse from general libraries. Reuse is conducted by the individual in an informal manner.
Systematic (planned)	Planned reuse is the systematic and formal practice of reuse as found in software factories.
Compositional	Compositional reuse is the use of existing components as building blocks for new systems.
Black-box / Verbatim	Reuse of software components without modification or "as is".
White-box	Reuse of components by modification and adaption.
Internal	Software items come from an internal repository.
External	Software items come from an external repository.

**Table 2. Types of reuse adapted from Frakes & Terry (1996).**

## 2.5 Benefits of software reuse:

Reuse-based software engineering is a software engineering strategy where the development process is geared to maximize the reuse of existing software. The move to reuse-based development has been in response to demands for lower software production and maintenance costs, faster delivery of systems, and increased software quality. An obvious advantage of software reuse is that overall development costs should be reduced. Fewer software components need to be specified, designed, implemented, and validated. Sommerville (2011) lists multiple benefits of reusing software assets and impediments to reuse (Sommerville, 2011, p. 427).

Benefit	Explanation
Increased dependability	Reused software, which has been tried and tested in working systems, should be more dependable than new software.
Reduced process risk	Less uncertainty in development costs due to known costs and possible risks of existing software.
Effective use of specialists	Application specialists can develop reusable software that encapsulates their knowledge.
Standard compliance	Some standards, such as user interface standards, can be implemented as a set of reusable components.
Accelerated development	Reusing software can speed up system production because both development and validation time may be reduced.

**Table 3. Advantages of software reuse adapted from (Sommerville, 2011, p. 427).**

## 2.6 Drawbacks of software reuse:

Literature addresses several impediments (See table 4) to reuse which can be for example of Managerial & organizational (lack of management support, procedures, or incentives), economical (investment hurdle), psychological (NIH syndrome, threat to creativity and independence), legal (liabilities, data rights) and technical nature (Sametinger, 1997, p. 15).

Technical difficulties around reuse have been explained by Talvalsairi (1993) and include: Agreeing on what a reusable component constitutes, understanding what a component does and how to use it, understanding how to interface reusable components to the rest of a design, designing reusable components so that they are easy to adapt and modify (in a controlled way), and organizing a repository so that programmers can find and use what they need (Taivalsaari, 1993).

In addition, successful reuse requires having a wide variety of high-quality components, proper classification and retrieval mechanisms, sufficient and proper documentation of components, a flexible means for combining components, and a means of adapting components to specific needs (Sametinger, 1997).

Risks	Explanation
Increased maintenance costs	If the source code of a reused software system or component is not available, then maintenance costs may be higher because the reused elements of the system may become increasingly incompatible with system changes.
Lack of tool support	Some software tools do not support development with reuse. It may be difficult or impossible to integrate these tools with a component library system.
Not-invented-here syndrome	Some software engineers prefer to rewrite components because they believe they can improve on them. This is partly to do with trust and partly to do with the fact that writing original software is seen as more challenging than reusing other people's software.
Creating, maintaining and using a component library	Populating a reusable component library and ensuring the software developers can use this library can be expensive. Development processes have to be adapted to ensure that the library is used.
Finding, understanding and adapting reusable components	Software components have to be discovered in a library, understood and, sometimes, adapted to work in a new environment. Engineers must be reasonably confident of finding a component in the library before they include a component search as part of their normal development process.  It can be difficult to find suitable assets that can potentially be used to solve a given problem. It is difficult to match a problem description to a solution description.
limited extensibility and modifiability.	Another impediment arises when commercial off-the-shelf (COTS) components are reused. Rarely are developers given the source code of a COTS component, so software that reuses COTS components has limited extensibility and modifiability.

**Table 4. Problems with software reuse adapted from (Sommerville, 2011, p. 427).**

### **3. Software reuse in the gaming industry**

The gaming industry is increasingly making use of Commercially-off-the-shelf middleware components like the movie industry (Rollings & Morris, 2004, p. 382); (Gregory, 2009, p. 13).

To help solve customers' heterogeneity and distribution problems, and thereby enable the implementation of an information utility, software vendors are offering distributed system services that have standard programming interfaces and protocols. These services are called middleware, because they sit "in the middle," layering above the OS and networking software and below industry-specific applications (Philip A. Bernstein, 1993).

As development of new titles have become so complex and development of games can take years of development it is no longer financially practical and reasonable to also completely rewriting core components such as AI behavior or character animation systems. As a game developer, time not being spent on a new 3d-engine or an engine feature is time that can be spent on creating a better game instead. Also on a publisher point of view, the use of common set of (middleware) components and standard game assets assures that a project can be easily moved to another developer.

*"At one point you just have to decide if you want to develop technology or make creative experiences," – Adrian Tingstad Husby, - Krillbite Studio.*

Using complete COTS 3D engines, commercial off-the-shelf (COTS) components (or middleware components) and standard game assets have now become widely preferred approaches over developing proprietary technology to simplify and shorten the development process, potentially leading to better productivity and a shorter time-to-market. The next section will concentrate about specific reusable areas of computer games.

#### **3.1 Game engines:**

In order to understand which parts of a game are specific and which are general we have researched different game engines that allows us to understand the separations and relations between the different parts of a game design.

The term "game engine" refers to software that is extensible and can be used as the foundation for many different games without major modification. Virtually all game engines contain a familiar set of core components, including the rendering engine, the collision and physics engine, the animation system, the audio system, the game world object model, the artificial intelligence system and so on (Gregory, 2009, p. 3).

Latest game engines provide a full development studio that provide core functionalities such as a rendering engine, sound-engine, animation, scripting, AI, networking, memory management, and publishing modules which makes it easy to publish the game to various platforms.

Next sections will further discuss a game engine's software architecture, identifies reusable areas of games and we will introduce our related hypotheses.

### **3.2 Software architecture in games**

A product architecture is the scheme by which the function of a product is allocated to physical components. The architecture of the product can be a key driver of the performance of the manufacturing firm (Ulrich, 1995, p. 419).

Architectural decisions are linked to the overall performance of the firm and to specific R&D issues, including the ease of product change, the division between internal and external development resources, the ability to achieve certain types of technical product performance and the way development is managed and organized.

In software products, the highest level abstraction is called the software architecture i.e. the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. The software architecture is an important artifact in the development of any system as it allows early analysis of the provided quality of a system such as performance, maintainability. ("ISO/IEC Standard for Systems and Software Engineering - Recommended Practice for Architectural Description of Software-Intensive Systems," 2007).

A game engine generally consists of a tool suite and a run-time component. Figure 1. on the next page shows all of the major runtime components (these components exist while the system is running) that make up a typical 3D game engine. Notice the various different engine layers that make up the game engine and its hierarchy in the system.

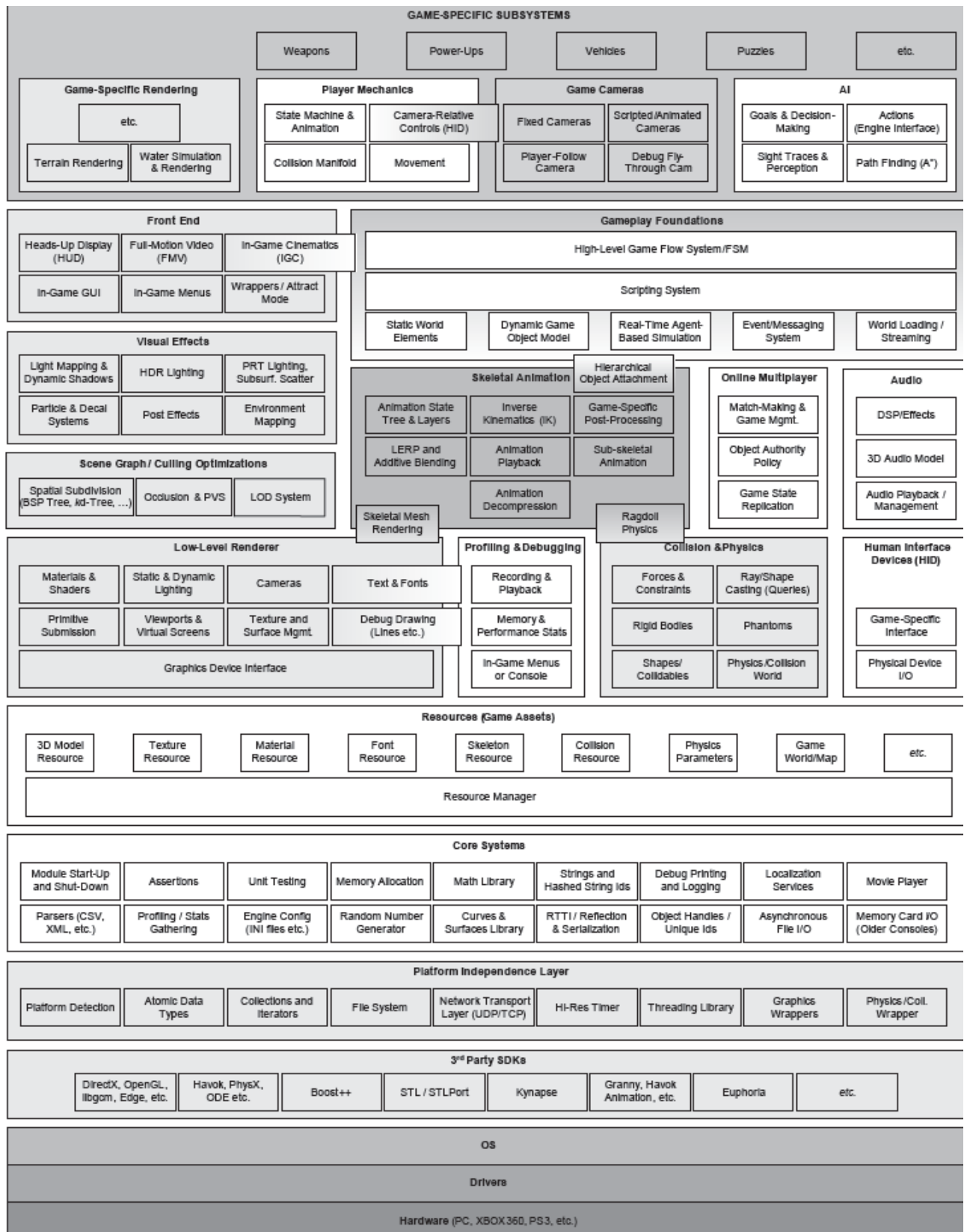


Figure 1. Game engine architecture. Adopted from (Gregory, 2009, p. 29)

### 3.3 Software Components

Games are complex software systems in that they comprise a large number of components with many interactions between them. During architectural design (also called high-level design), a modular decomposition of the product is developed. In this phase specifications are carefully analyzed, and a module structure that has the desired functionality is produced. The output from this activity is a list of the modules and a description of how they are to be interconnected (Schach, 2011, p. 466). During architectural design, the existence of certain modules is assumed and the design then is developed in terms of those modules.

This concept of modularity has become increasingly important (Miguel, 2005, p. 165). In many industrial sectors like automotive (Morris, Donnelly, & Donnelly, 2004), computers and software (Baldwin & Clark, 1997), electronics systems (Sanchez & Mahoney, 1996, p. 67) migrate toward increasing modularity to deal with the growing complexity in systems. Some systems that were originally tightly integrated may be disaggregated into loosely coupled components that may be mixed and matched, allowing much greater flexibility in end configurations (Schilling, 2000, p. 313).

Component Based Software Engineering is a branch of software engineering and a reuse based approach to defining, implementing and composing loosely coupled independent components into systems. It encourages the use of predictable architectural patterns and standard software infrastructure, thereby leading to a higher-quality result.

Software components are executable units of independent production, acquisition, and deployment that can be composed into a functioning system. Composite systems composed of software components are called component software and provides a rationale for breaking a product into modules as a way to reduce the cost of maintenance which is a major component of the total software budget (Stevens, Myers, & Constantine, 1974). The maintenance effort is reduced when there is maximal interaction within each module and minimal interaction between modules. A good software design thus is a design in which modules have *high cohesion* and *low coupling*.

Abstractions, such as procedures, classes, modules, or even entire applications, could form components, as long as they are in an executable form that remains composable (Szyperski, Gruntz, & Murer, 2002, p. 4).

The goal of component-based technology is to construct a standard collection of reusable components. Then, instead of reinventing the wheel each time, in the future all software will be constructed by choosing a standard architecture and standard reusable frameworks and inserting standard reusable code artifacts into the hot spots of the frameworks. For this technology to work, the components have to be independent and fully encapsulated and like objects and only communicate by means of exchanging messages. This principle allows for reduction of development time through technical facilities that enable the easy assembly and upgrading of systems out of independently developed pieces of software over multiple projects (Schach, 2011, p. 594).

From a high-level business perspective, both CBSE and Reuse-oriented software engineering have the same goals: Increasing productivity and quality. Research shows a positive and significant evidence on lower problem density (defect-, error- or fault density) and effort spent on corrections (rework effort) with introducing systematic reuse in industry (V.R. Basili et al., 1996); (de O. Melo et al., 2013); (W. Frakes & Terry, 1996); (Ajila & Wu, 2007); (Mohagheghi et al., 2004).

### **3.4 COTS Game Components**

A way to reduce the development costs of games in particular is to reuse specific Game Components in the game. Rather than reinventing the wheel when developing a game engine, a physics engine or a network component, game developers can choose to use an existing components from an internal component library or use Commercial of the Shelf (COTS) Components.

COTS-Game engines and COTS-components have become widely preferred approaches over proprietary technology to simplify and shorten the development process, potentially leading to better productivity and a shorter time-to-market.

A survey by gamesutra.com in 2009 found that 55% of the respondents were using a middleware game engine, such as the Unreal Engine on their project ("Gamasutra: Mark DeLoura's Blog - The Engine Survey: Technology Results," 2009).

Commercially available middleware 3d engines are for example Unity 3d, Unreal Engine & Cry-engine. These packages are evolving to become the industry-standard game development studios with all necessary tools and components such as graphics, AI ,animation, sound, scripting, publishing and multi-platform support built in to it to simplify game development.

One of the primary reasons for using a COTS is to give programmers and artists more time to work on the title, especially during the prototyping and early concept stage allowing more refined or unique gameplay. Financial and creative efforts can be put into creating a game, rather than on the R&D that is needed for creating a modern game engine.

It also allows programmers to focus on creating technology that distinguishes the game from others of a similar genre. Another benefit that comes with COTS components is that it is easier to hire somebody who has already experience with a COTS Engine as opposed to a custom engine. Finally, a big advantage of the usage of a third-party game engine is that financial and creative efforts can be put into creating a game, rather than on the R&D that is needed for creating a modern game engine ("Gamasutra: Mark DeLoura's Blog - The Engine Survey: Technology Results," 2009).

Some concerns come with using a COTS game engine as well. For example, a particular game engine may not work for a specific game genre or can be tied to specific platforms. Another concern is the difficulty of working with, extending and modifying an unfamiliar code base. Some developers pointed out that they have spent more time debugging poorly-crafted middleware than they would have spent writing it from scratch themselves. Source code access is also a vital concern to being able to evaluate the engine to make sure that it integrates well with existing or other COTS libraries. Legal agreements is also another issue in case of a company files bankruptcy or is acquired by another company.

In addition a COTS based approach benefits the game industry as a whole as successful COTS developers can focus on one particular aspect of a game e.g. physics, or even building a complete 3d engines.

This allows them to advance their technology at a faster rate than when they were building games. These advances are then available for more games to use, which benefits the industry.



### 3.5 Reusable game components and project performance

All game engines contain a familiar set of core components, including the rendering engine, the collision and physics engine, the animation system, the audio system, the game world object model and the artificial intelligence system (Gregory, 2009, p. 29).

Because of the rapid evolution of video games the last decade, game developers now can choose from a wide variety of components dealing with various aspects of games e.g. rendering, object management, physics, artificial intelligence and so on.

By studying various game engine architectures we have identified the following major components or specialized game domains that we found present across several game engine architectures. These components can be seen as some part of a software system that is identifiable and reusable.

We will further use the term asset to denote a unit of reuse and component to denote a unit of composition.

Identified components	Description
Artificial Intelligence components	Component handling path finding, actions, goals & decision making etc.
Rendering components	Component handling terrain rendering, materials & shaders, cameras, static & dynamic lightning, Scene Graph etc.
Physics components	Component handling collision, ragdoll, cloth etc.
Animation components	Component handling HDR lighting, Post effects, Particle & Decal systems, Light mapping & shadow etc.
Visual effects components	Component handling DSP/effects, 3d audio model, audio playback /management etc.
Game specific subsystems	Component handling Player mechanics, Game Cameras, weapons, power-ups, puzzles etc.

**Table 5. Reference Game engine components.**

In the beginning reuse can be expensive as states three costs that are involved: The cost of making something reusable, the cost of reusing it, and the cost of defining and implementing a reuse process. Tracz (1994) estimates that just making a component reusable increases its cost by at least 60 percent (Tracz, 1994).

It is expected that when the Game Components are used multiple times, the accumulated defect fixes eventually results in a higher quality work product (Lim, 1994). Over time reused Game Components may be designed more thoroughly and be better tested, since faults in these components affect several products and the prevention costs are amortized over several products (Mohagheghi et al., 2004).

A big advantage of the usage of a third-party game engine is that financial and creative efforts can be put into creating a game, rather than on the R&D needed for the in-house technology creation that is needed, which is likely to exist in current game engines.

Component reuse holds the potential to reduce overall system development costs and development time because many high quality components (such as AI-, physics-, audio- or animation components) can be bought off-the-shelf or reused from other projects instead of having to be developed from scratch. Buying the component is usually cheaper as the development costs for the component are being spread out over the multiple game titles in which the component is incorporated.

A higher quality of game components is also to be expected as one can assume that bought or reused components are being used in different games, in different environments; more rigidly testing and stressing the quality of the component than in a single game setting.

Based on our initial exploratory study through several interviews with game and software developers (Chapter 4.1 and Appendix A) and the earlier claimed benefits on software reuse in our literature review we propose the following hypotheses related to the degree of game components reuse and project performance and the degree of Overall software reuse on Project performance:

*H1 (a): An increase in the degree of component reuse has a positive effect on cost efficiency.*

*H1 (b): An increase in the degree of component reuse has a positive effect on the development time efficiency of the game.*

*H1 (c): An increase in the degree of component reuse has a positive effect on product quality.*

*H1 (d): An increase in the degree of component reuse has a positive effect on profitability.*

*H1 (e): An increase in the degree of overall software reuse has a positive effect on project performance.*

### **3.6 Reusable game components and game performance**

A large number of games have been built with existing game technologies. ("100 Most Popular Game Engines - Mod DB," 2015). For many years FPS engines like the Doom engine, Unreal, Cry-engine, have set the standard in terms of graphical fidelity and they have spawned numerous successful games. These game engines have primarily focused on the rendering technology and relevant sub domains such as AI, physics and animation ("The evolution of PC graphics will blow your mind | TechRadar," 2015).

New advances in computer hardware and rendering algorithms over the years caused the average game engine to grow in scale and complexity with new engine features and capabilities added each year. Each newly released game showcasing a typical new feature would set the benchmark for future games. As a result, gamers are rapidly expecting new features to be standard included in each new game. ("Matt Chat 99: Duke Nukem with Scott Miller - YouTube," 2014).

Since computer hardware becomes more powerful over time and most popular game engines predominantly focus on producing better graphics, we posit the following hypotheses:

*H1 (f): An increase in the degree of reuse of the Rendering components and the Visual Effects components has a positive effect on Graphics score.*

*H1 (g): An increase in the degree of reuse of AI components, Physics and Game specific subsystems has a negative effect on Gameplay score.*

In addition to above hypotheses we hypothesize that too much (unmodified) overall software reuse and Overall component reuse in general throughout the game will negatively impact review scores. Some components may need to be adapted specifically to requirements of the game. If certain game components make it unmodified into the game this could negatively affect the review scores of the game.

*H1 (h): An increase in the overall degree of software reuse and overall degree of component reuse has a negative effect on Review scores.*

### 3.7 Reusable game assets and project performance

A game engine's input data comes in a wide variety of forms, from 3D-mesh data to texture bitmaps to animation data to audio files. Game Assets are a collection of such data files such as models, textures, sounds, animations which support gameplay. These assets are usually designed for generality and allow cost reduction through software reuse of previously developed assets in the development of new games.

The data files that make up an asset usually adhere to some particular asset conditioning pipeline so that it can be used by the game-engine. This is because all the different data formats used by digital content creation (DCC) applications are rarely suitable for direct use in-game (Gregory, 2009, p. 49). Therefore, data produced by a DCC application is usually exported to a more accessible standardized format, or a custom file format, for use in-game. Once data has been exported from the DCC application, it often must be further processed before being sent to the game engine. When a game studio is shipping its game on more than one platform, the intermediate files might be processed differently for each target platform.

Game Assets such as 3d-models, artwork, music, sound, animations, scripts, even complete game libraries can be bought off the shelf or found for free on the Internet. There are tons of libraries of 3D objects and animations available from various sources such as Digimation.com, 3drt.com, tf3m.com and Maximo.com. Some COTS 3d-engines such as the Unity 3d engine and Unreal 4 Engine even feature a built-in Asset Store where it's easy to purchase and sell specialized game components and game assets.

We identified the following game assets that can be reused in a computer game:

Identified assets	Description
Game objects	Any object in the game with special properties.
Game environments	Complete game environments: e.g. a forest scene complete with models of trees and grass.
3D models	3D models represent a 3D object using a collection of points in 3D space.
Audio files	Audio and music files.
Scripts	Generally relatively small software code snippets to automate certain domain tasks, e.g. AI.
Textures	Images applied to the surface of a 2D or 3D objects.
Materials	Materials that are attached to game objects to copy real-life properties of objects.
Animations	Animations animate an game object.
Shaders	Shaders are used to create appropriate levels of color in an image, for special effects or post-processing.
Story elements	Specific story elements can be problems, plots, characters.

**Table 6. A collection of standard assets that can be reused across different game projects.**

The above table makes clear that game engine must be fed a great deal of data, in the form of game assets such as game objects, scripts, animations, audio files and so on.

With game worlds ever growing in size and more detailed game companies must manage to create generic objects that can be used and combined in many ways by environment artist to create new objects or complete environments. These generic objects form the building blocks for artist such as level artist to create a game scene. They will search through an asset manager and find the most appropriate game assets and instance them into the game engine (van Beek & Valient, 2011).

In cases when game assets are bought of an Game Engine's proprietary Asset store or are used from an internal asset library these assets are generally directly suitable for usage in the game engine. These assets have already been optimized for the Game Engine's asset conditioning pipeline by their creators (Gregory, 2009, p. 49) saving time and potential costs, thus we posit the following hypotheses:

*H2 (a): An increase in the degree of asset reuse has a positive effect on cost efficiency.*

*H2 (b): An increase in the degree of asset reuse has a positive effect on development time efficiency.*

There can also be significant cost associated with understanding whether or not an asset is suitable for reuse in a particular situation, and in testing that asset to ensure its dependability. Some external assets can be difficult to adapt and modify or have only limited capabilities in this respect. It is therefore important to know one's own requirements for external assets and, in case they do not completely fulfill them, to determine whether it is possible and how difficult and time-consuming it is to make any necessary modifications. However, in cases when an asset being reused already closely matches the need evoked from the asset's new requirement, contain a well-structured document and have been designed for a comparable scenario as the modified asset requirements then lower development cost, development time and a higher product quality (less defects in the game) and therefor higher profitability is likely to be expected. We propose the following hypotheses related to the degree of Game Assets reuse and product quality and the degree of Game Assets reuse and Profitability:

*H2 (c): An increase in the degree of asset reuse has a positive effect on product quality.*

*H2 (d): An increase in the degree of asset reuse has a positive effect on profitability.*

### **3.8 Reusable Game Assets and Game performance**

The greater assets are reused unmodified throughout a game such as 3d models, textures and sounds the less diverse the game could look and feel. A perfect example are Indie games created with COTS-game engines where people buy the same components or assets over and over again from the Game Engine's Asset store for a game engine such as Unity or Unreal 4 and put them in into the game. Even complete game templates in many different game genres can be bought of these Asset stores such as racing, puzzle, or shooters ("Unity3d Asset Store," 2015); ("Marketplace - UE4 Marketplace," 2015). The result from this is that extra resources will be needed to more effectively distinguish the game from other similar games.

Another good example are game sequels where the same assets from the previous game could potentially be reused to save costs on the development of new game assets. Game developers should wisely consider what to reuse and update or rebuild from scratch. If certain assets make it unmodified into the game this could negatively affect the review scores of the game. For example, too much reuse of visible elements could make the levels look generic and less diverse, ultimately negatively affecting review scores. ("Visceral Games Speaks Out on Battlefield Hardline Re-Using Battlefield 4 Assets," 2014)

We hypothesize that too much (unmodified) reuse throughout the game negatively impacts review scores.

*H2 (e): An increase in the degree of asset reuse has a negative effect on Review scores.*

### 3.9 Internal and External software reuse

Software considered for reuse may come from external sources through COTS or open-source components developed by others outside the company and may have a significant influence on product outcomes such in terms of performance, scalability, manageability, portability and quality (Philip A. Bernstein, 1993).

The quality for example of external software components can be a major concern of managers and developers. A list of known defects and reference sites may give a first indication of a component's quality. In addition, any external component might prove more effective if the component is well documented, generalized and of high quality (Sametinger, 1997).

Using external components may reduce a product's development time, but it also means increased dependence on component suppliers. Costs can potentially be avoided by not having to develop and maintain certain game assets. Potential costs lie in the possibility of having to adapt and modify them and integrating it into the product under development, but this depends on the requirements.

Making reuse cost-effective can be accomplished by increasing the level of reuse, by reducing the average cost of reuse, and by reducing investments to achieve reuse benefits such as making components easy to find, adapt and integrate into new systems (Barns & Bollinger, 1991).

We hypothesize that in general, an increase in external assets reuse increases development time efficiency and costs efficiency compared to creating the asset from scratch yourself.

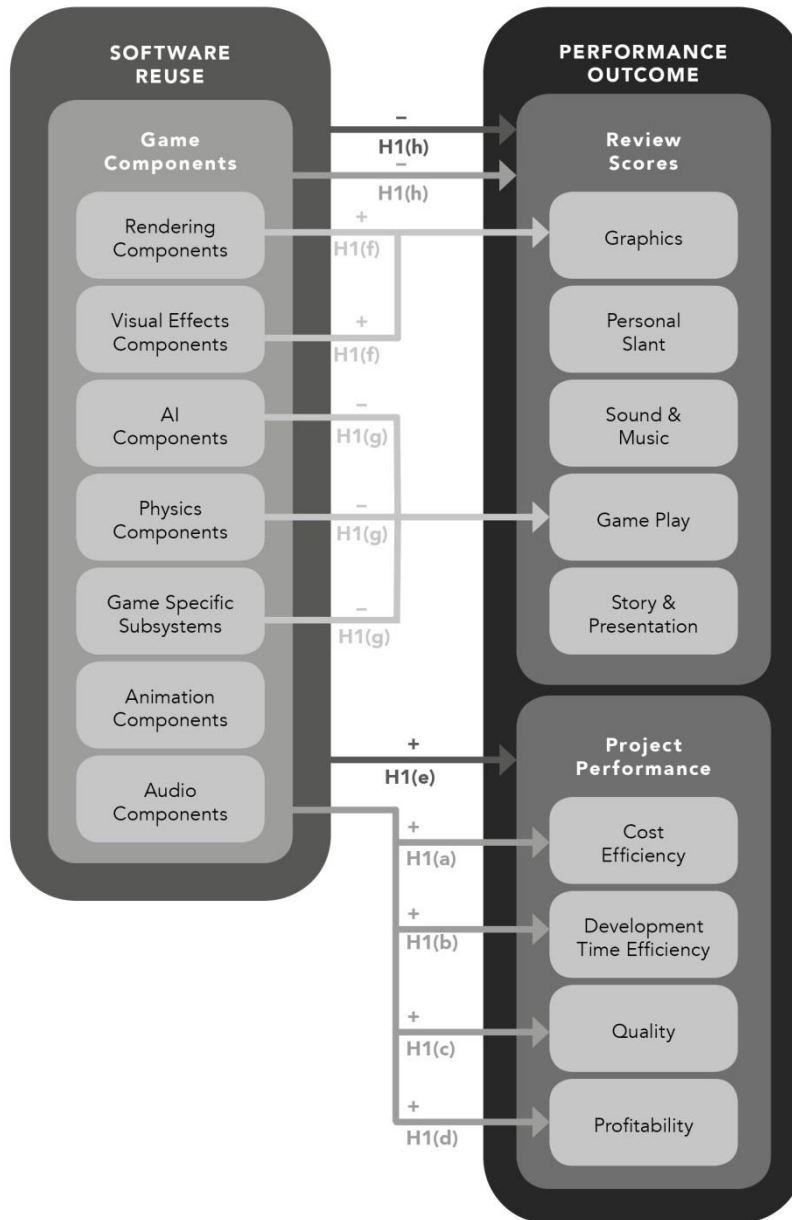
*H3 (a): An increase in the degree of external reuse has a positive effect on development time efficiency.*

*H3 (b): An increase in the degree of external reuse has a positive effect on cost efficiency.*

It is also hypothesized that too much (unmodified) external reuse, could negatively affect the novelty of a game, negatively affecting review scores for the same reasons as discussed in previous chapter. We therefore propose the following hypothesis related to external software reuse and review scores.

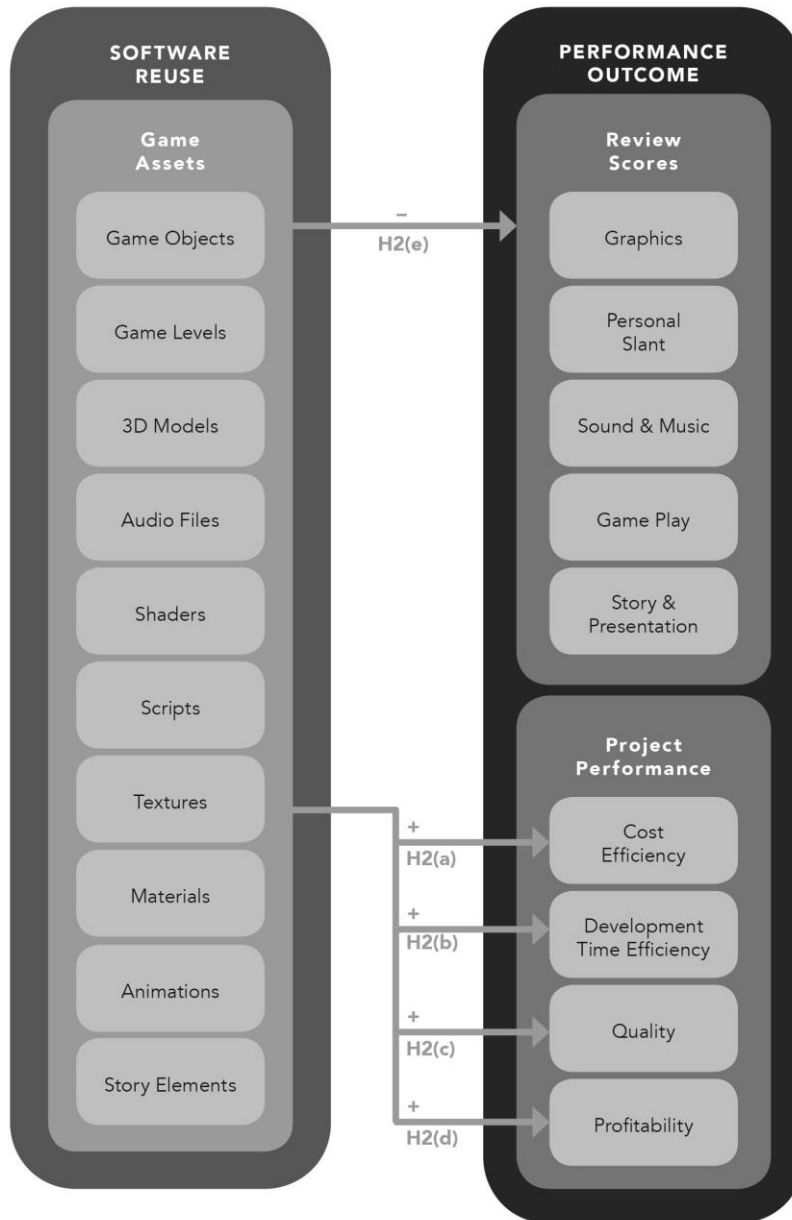
*H3 (c): An increase in the degree of external reuse has a negative effect on review scores.*

**FIGURE 2**



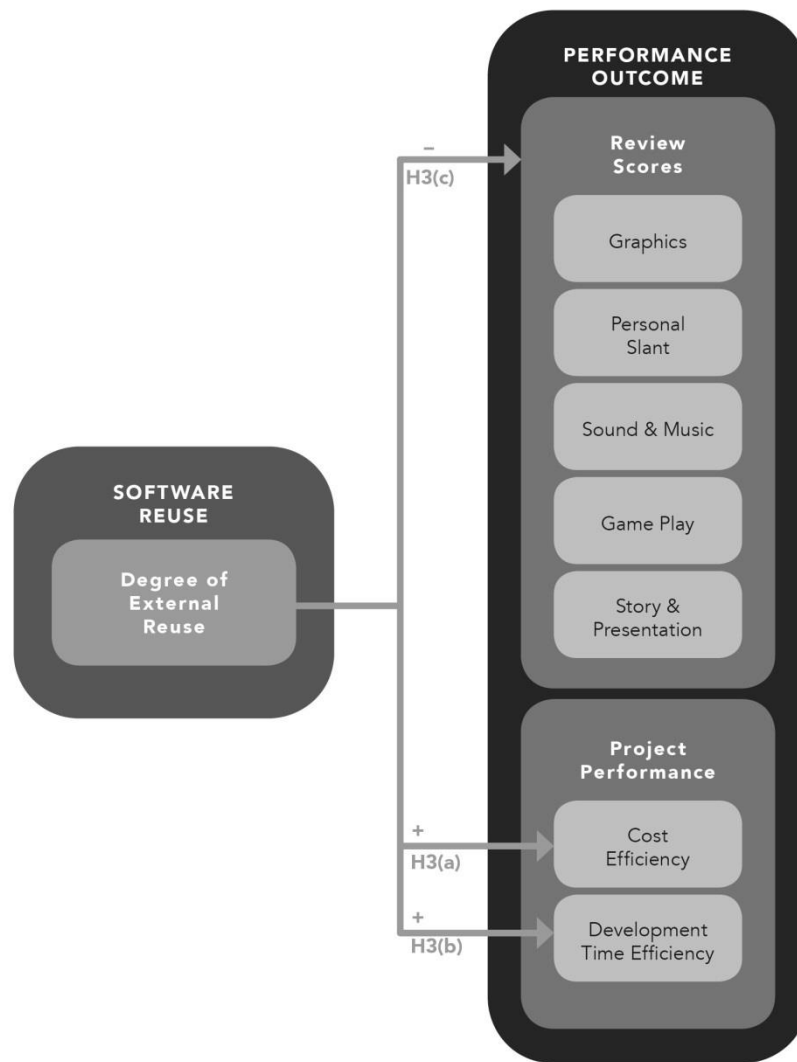
**Figure 2. Hypothesized conceptual model of the relationship between Game components reuse, review scores and project performance.**

**FIGURE 3**



**Figure 3. Hypothesized conceptual model of the relationship between Game Assets reuse, review scores and project performance.**

**FIGURE 4**



**Figure 4. Hypothesized conceptual model of the relationship between the degree of external reuse and Review scores, Cost efficiency and Development time efficiency.**



## **4. Methods**

In the paragraphs below literature around software reuse, areas of reuse, reuse strategies and reuse in computer games is reviewed. Hypotheses are then developed how software reuse in video games affects project and game performance.

### **4.1 Sample and research methods**

In February 2015 a random sample of 124 games were targeted for this study that were published during 2010 – 2014 on any game platform (pc and consoles). The game titles and developer names used in this study were randomly collected from Mobygames.com, an online database listing various information about videogames such as reviews, credits and game company information. Technical Directors and Creative directors and people holding similar senior technical and creative functions at game companies with different firm sizes were extracted out of the credits list of the games. We then collected the person's email addresses via the Internet manually from sources such as Linked-in, the company's website, or any personal sites. Unfortunately this proved a real time-consuming task and it turned out that many personal email addresses were unfindable.

The people we did find an email address from were asked to fill in a questionnaire about the specific game they had worked on. The unit of analysis in this study are video games, the subjects for the study are game developers that have worked on a particular game.

The survey was eventually sent out to 211 personal email addresses of people that worked on a game in our sample. If we found email addresses of people working on the same game we included these in our mailing to increase the chance of a valid response for a game. A total of 27 email addresses bounced and 13 were sent to non-personal email addresses but to the info email addresses instead.

Prior to the survey we did an extensive literature review on the concepts of software reuse and product modularity and did an initial exploratory field study through several interviews with game and software developers. Our objective of these interviews was to understand possible areas of software reuse and how reuse is done, identifying challenges and benefits of software reuse.

Five interviews were also held in a semi-structured way to understand the strategic choices from the development team's perspective and how they affected product and project performance (Appendix A).

The result of our literature review and interviews provided insight in the following questions:

- How can software reuse be defined?
- What are potential areas of software reuse in games?
- What are the positive and negative effects of software reuse on different software development economics?
- What are the main decision making aspects when deciding between reuse or building something new?

The results of the literature review and exploratory study was then used to formulate the research model, hypotheses and our survey questionnaire.

## **4.2 Response**

Out of 124 games approached, 16 people that had also worked on 16 different games completed the survey. Since 27 email bounced and 13 were sent to non-personal info addresses the survey's response rate is  $16 / 82 = 19.5\%$ . The full survey is included in Appendix B.

The Likert scale questions for the degree of reusable Game Components, Game Assets and project performance had no missing values. There were some values missing about the development budget (3 missing), used game engine name (2 missing), and game size in terms of code due to lack of knowledge on this question or non-disclosure on this subject. Overall, the few missing values did not cause a problem and contained sufficient data for further analysis. No entire cases were excluded for doing analyses related to Project performance. However three cases were left out when analyzing the Game performance measures because for these games there was missing review data.

## **4.3 Operationalization**

### **Variables and measurement**

We developed 7-point bipolar Likert-type questions for our study variables. For each item the scale ranges from 1 "Strongly Disagree" and 7 "Strongly Agree"). Item ratings were summarized to form an average (overall) rating scale for the independent and dependent variables consisting of multiple Likert items. Where possible we used existing items or question format from existing studies.

Based on our exploratory study and literature review we identified the following independent variables, dependent variables and control variables:

#### **4.3.1 Independent variables:**

While software reuse can occur in many levels of granularity such as a few lines of code, methods, component, classes or whole systems in this study we frame our investigation mainly around the reuse of Game Components and Game Assets. These parts should be more easily identifiable across different games over abstractions such as procedures, number of classes, modules, lines of code, functions and have proven to be distinct parts that together make up a video game as explained in Chapter 3.3 – 3.9. The degree of software reuse in this study is thereof measured in terms of:

- Degree of reuse of used Game Components & Game Assets:

Game Components:

Artificial components, Rendering components, Physics components, Animation Components, Visual effects Components, Audio components, Game specific subsystems. Adopted from (Gregory, 2009, p. 29), ("Game Systems | HeroEngine," 2012). The reuse scale ranges from (1) No reuse at all - (7) Full reuse.

#### Game Assets:

Game objects, game levels, 3d Models, audio files, shaders, scripts, textures, materials, animations and story elements. Mostly adopted from ("Unity3d Asset Store," 2015). The reuse scale ranges from (1) No reuse at all - (7) Full reuse.

- Degree of external software reuse – The extent to which developers have used external components via Open-Source & COTS-Components. (Philip A. Bernstein, 1993). The reuse scale ranges from (1) Fully internal - (7) Fully external.

#### 4.3.2 Dependent variables

The items for the dimensions of the dependent variable measuring project performance (profitability, development time efficiency, cost efficiency and product quality) were adapted from existing literature and measured using a 7-point bipolar Likert scale. Reviews scores were collected from an online database.

We identified the following dependent variables:

- Game performance – Review scores were collected from Mobygames.com. A combined overall rating from different game review sites and target platforms is summarized for different game scoring criteria (Gameplay, Graphics, Personal slant, sound & music, Story & presentation. The scale ranges from (1) very poor - (5) very good.
- Project performance – We operationalized this variable by measuring multiple project performance criteria. Whether the project was completed in a time efficient manner, Items were adapted from (Kessler, 1999), whether the project was cost efficient, adapted from (R. G. Cooper & Kleinschmidt, 1987), whether the project met quality goals adapted from (Atuahene-Gima, 2003); (Sahay & Riley, 2003); (Mohagheghi et al., 2004) and met profitability goals, adapted from (Song & Parry, 1997). The scale ranges from (1) strongly disagree - (7) strongly agree.

#### 4.3.3 Control variables

Three control variables were included in the data analysis.

Because systematic software reuse was found to be significantly related to lower problem density (defects, faults or errors), lower rework effort and increased apparent productivity in earlier studies (Lim, 1994); (Mohagheghi et al., 2004); (Selby, 2005); (Thomas et al., 1997) we included this variable as our first control variable. Software reuse is most effective when it is planned as part of an organization-wide reuse program. A reuse program involves the creation of reusable assets and the adaptation of development processes to incorporate these assets in new game titles (Sommerville, 2011, p. 427). Having a low or high degree of systematic reuse process in the firm can potentially influence our performance outcomes and thus we included this control variable in the data analysis.

Another variable that could potentially confound our study results is the used Game engine type: Using an internal or external middleware (COTS) game engine may have a different effect on the measures Project performance variables ("Gamasutra: Mark DeLoura's Blog - The Engine Survey: Technology Results," 2009). Because there could be

a difference between games using an internal or external middleware game engine in the degree of software reuse and project performance outcomes this control variable was also included in the study.

Lastly, we controlled for the development Team size because larger firms may have other heterogeneous firm resources that could significantly contribute to product and project performance outcomes (Barney, 1991). For example, larger teams could have more experience, financial resources or larger R&D capacity available that could influence the product and project performance outcome.

- Systematic reuse process – Whether the firm has a structured, systematic reuse process for reuse that is applied and integrated in the firms development process, uses databases listing standard components and has flexible means for combining components through standard interfaces among modules to achieve substantial benefits in productivity, quality and business performance. Measures were adapted from (Sametinger, 1997); (Taivalsaari, 1993); (Worren, Moore, & Cardona, 2002); (Tiwana, 2008). The scale ranges from (1) strongly disagree - (7) strongly agree.
- Game engine type – Whether the game engine used was a proprietary game engine or a middleware (COTS) game engine.
- Team size - Total number of full time developers that have worked on the game. We differentiated between small (<55 FTE) and large sized development teams (>55 FTE) as the measure of team size. For this we looked at the median value of the development team. Team size values lower than the median were recoded into the low team size category and values higher than the median were recoded into the high nominal category. The median value group was recoded to the same value of the smallest frequency, e.g. the lower frequency of the low or high category.

#### **4.4 Instrument validation**

A pilot study was conducted by distributing the preliminary survey to a game developer working at a large game studio in the Netherlands and also two professional staff members of the University of Twente in the Netherlands. The contacted game developer was asked to examine with his colleagues at the game studio whether the preliminary questionnaire captured the measured constructs well and whether the questionnaire was clear and understandable to them. Based on received feedback of the pilot minor adjustments were made in the instrument before sending out the full survey. Content validity was tested by defining the topic of concern, describing items to be scaled, developing scales to be used and using a test panel of experts to maximize the quality of the construct (D. R. Cooper & Schindler, 2014).

Measurement reliability is the extent to which a set of measurements is free from random error variance. In more practical terms, reliability refers to the consistency of a set of measures. Cronbach's alpha (Cronbach, 1951) provides an estimate of reliability by assessing the internal consistency of a set of items in a scale or test. Cronbach's alpha are widely used in business research (Chau, 1999) with reflective models for reliability assessment and is based on correlations among the indicators that compromise a measure with higher correlations among the indicators associated with high alpha coefficients. Cronbach's alpha were calculated for all constructs and dimensions in the conceptual model. The Cronbach alpha's alpha values for degree of

Component reuse ( $\alpha = 0.898$ ), Asset reuse ( $\alpha = 0.952$ ), project performance (table 7) and the control variable of Systematic reuse ( $\alpha = 0.838$ ) all exceeded the suggested value of 0.70 standard advocated by (Cohen, 2003) in empirical research and thus the measures can be considered reliable.

Construct	Dimension	Cronbach's alpha
<b>Project Performance</b>	Profitability	0.926
	Development time efficiency	0.850
	Cost efficiency	0.861
	Product quality	0.869
<b>Control variable</b>	Systematic reuse process	0.838

**Table 7. Cronbach's alpha scores of the project performance dimensions**

## 5. Results

### 5.1 Analysis

We started the analysis by testing for normality. Visual inspection of the data and Shaphiro Wilk test indicated that all outcome variables regarding Project Performance and Game Performance in the study were approximately normally distributed (Appendix D). Inspection of the independent variables indicated that all independent variables relating to asset reuse were approximately normally distributed but the sample data for the overall component reuse, specific components reuse, specific assets reuse and external reuse were not normally distributed but slightly skewed.

Because different people with different functional backgrounds have responded we first tested whether there were significant differences between the multivariate means of the different populations (respondents with either a Technical background or a Creative background). It is therefore important to verify if the groups did not give significantly different answers. A MANOVA test (multivariate analysis of variance) was executed to test for homogeneity and it was concluded that there is no significant difference in the answers for the two different backgrounds. Wilks  $\lambda = .023$ ,  $F(14,1) = 3.07$ ,  $p = .422$ , partial  $\eta^2 = .997$ .

We used Pearson product-moment correlation coefficient (Pearson  $r$ ) to test for correlations between all study variables. The Pearson product-moment correlation coefficient is a measure of the strength of the linear relationship between two variables and therefore used for testing our hypotheses. We also visually inspected and described the data when there was no linear statistical relationships associated between the variables to explain the exact relationship using a polynomial term—a quadratic (squared) or cubic (cubed) term- that turns a linear regression model into a curve as means of further exploration.

Additionally we did an analysis of variance (ANOVA) to check whether a low/high degree of reuse affects game and project performance to complement our managerial recommendations. For this we looked at the median score of the average (overall) score of the reuse variables. Scores lower than the median were recoded into the low reuse category and scores higher than the median were recoded into the high nominal category. The median value group was recoded to the same value of the smallest frequency, e.g. the lower frequency of the low or high categories. This method preserves as much of the actual data's variance as possible within the reduced two value response data set. To ensure our data set met the ANOVA assumptions we examined the data for linearity, homogeneity of variance and multi-collinearity.

## 5.2 Results

The descriptive statistics and the correlation matrix for the variables are shown in Table 8.

Table 8. Descriptive statistics and correlation coefficients for the project performance variables.

(N=16)	Mean	Std. Deviation	1	2	3	4	5	6	7	8	9	10	11	12
1. Overall degree of software reuse	4.56	1.46	1.00											
2. Overall reuse estimation system components	4.25	1.39	.88**	1.00										
3. Overall reuse estimation game assets	2.94	1.77	.53*	.41	1.00									
4. Primary source of the total reuse portion	2.56	1.26	.40	.29	.37	1.00								
5. Specific Components	4.47	1.51	.86**	.95**	.39	.24	1.00							
6. Specific Assets	2.94	1.65	.45	.30	.95**	.41	.27	1.00						
7. Profitability	3.59	1.78	.11	.20	.06	-.26	.13	.04	1.00					
8. Development time efficiency	4.06	2.06	.33	.46	.34	.02	.46	.34	-.03	1.00				
9. Cost efficiency	4.58	1.97	.49	.57*	.16	.18	.56*	.17	-.27	.82**	1.00			
10. Product Quality	4.97	1.57	-.19	-.26	.14	-.40	-.21	.16	.22	.00	-.05	1.00		
11. Systematic reuse process	4.86	1.18	.62*	.54*	.18	.22	.58*	.16	-.16	.29	.69*	-.04	1.00	
12. Team size	86.81	102.11	.06	-.19	.08	-.05	.26	.08	.54*	.16	-.17	-.25	-.05	1.00

\*, Correlation is significant at the 0.05 level (2-tailed).

\*\*, Correlation is significant at the 0.01 level (2-tailed).

Examination of the means for Overall degree of software reuse and Overall estimated Components reuse in our study have an average score of 4.56 and 4.25 which is close to middle of the scale. The Overall reuse estimation for the Game Assets was 2.94. These overall means closely match the means of the of the specific Components and Assets scale that were averaged being 4.47 for Components and also 2.94 for Assets. The mean of the primary source of the reuse proportion scale was 2.56 indicating slightly more internal reuse than external reuse. The means of the dependent variables for Development time efficiency, Cost efficiency and Quality were slightly higher than the middle of the scale ranging from 4.06 – 4.97. Profitability was slightly lower than the middle of the scale with an average score of 3.59. Examination of the correlations between the variables indicate that the Overall degree of reuse is highly positively correlated with the degree of System component reuse measured by an Overall reuse estimation of System components and the 7 averaged Specific components in the study. The same applies for the degree of Asset reuse. The degree of Component reuse both for the overall measure and the 7 averaged specific components measures were significantly positively correlated with Cost efficiency and Development time efficiency was also significantly positively correlated with the Cost efficiency. Table 8 shows that Systematic reuse process was significantly positively correlated with the overall degree of software reuse, overall- and specific components, and Cost efficiency and Team size was significantly positively correlated with profitability.

The descriptive statistics and the correlation matrix for the Game Scores are shown in Table 9.

Table 9. Descriptive statistics and correlation coefficients for the game performance variables.

(N=13)	Mean	Std. Deviation	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1. Overall degree of software reuse	4.77	1.30	1.00														
2. Overall reuse estimation system components	4.46	1.33	.84**	1.00													
3. Overall reuse estimation game assets	3.00	1.78	.40	.28	1.00												
4. Primary source of the total reuse portion	2.77	1.30	.31	.16	.36	1.00											
5. Specific Components	4.70	1.40	.82**	.94**	.24	.07	1.00										
6. Specific Assets	2.95	1.71	.34	.18	.95**	.43	.14	1.00									
7. Acting	3.82	.58	-.60**	-.33	.02	-.06	-.28	.02	1.00								
8. AI	3.60	.51	-.34	-.11	-.11	-.55	.07	-.23	.60*	1.00							
9. Gameplay	3.71	.58	-.48	-.26	-.16	-.66*	-.17	-.27	.56*	.88**	1.00						
10. Graphics	3.81	.64	-.61*	-.39	-.14	-.50	-.29	-.25	.69**	.78**	.90**	1.00					
11. Personalized	3.58	.67	-.49	-.26	.01	-.43	-.12	-.10	.72**	.84**	.88**	.93**	1.00				
12. Sound & Music	3.76	.53	-.63*	-.37	-.09	-.29	-.37	-.12	.87**	.88**	.81**	.82**	.82**	1.00			
13. Story & Presentation	3.49	.70	-.58*	-.39	.02	-.52	-.27	-.03	.77**	.78**	.87**	.92**	.95**	.86**	1.00		
14. Overall score	3.68	.56	-.60**	-.35	-.04	-.47	-.24	-.13	.82**	.86**	.92**	.95**	.96**	.91**	.97**	1.00	
15. Systematic reuse process	4.86	1.18	.46	.37	-.02	.06	.41	.02	-.52	-.49	-.64*	-.56*	-.47	-.68*	-.47	-.60*	1.0
16. Team size	86.81	102.11	-.04	.11	.08	-.18	.20	.10	.55	.42	.34	.39	.36	.40	.45	.44	-.20

\*, Correlation is significant at the 0.05 level (2-tailed).

\*\*, Correlation is significant at the 0.01 level (2-tailed).

Examination for the means for the dependent variables for the Game scores have a score less than 4.0 which is more than the middle of the Game score scale. The means of the dependent variable for Overall degree of software reuse and Overall reuse of system components were higher than the middle of the scale. An examination of the correlations between the variables indicate that Overall degree of reuse is significantly negatively correlated with Acting, Graphics, Sound and Music, Story and presentation and Overall game score. Table 9 shows that more external reuse was negatively correlated with Gameplay score. A systematic reuse process significantly was negatively correlated with Gameplay, Graphics, Sound & Music and Overall score.

### 5.2.1 Specific components reuse on project performance

Hypothesis 1 (a) – To examine whether an increase in the degree of component reuse has a positive effect on Cost efficiency Pearson r was used. Table 8 shows that there is a statistically significant positive correlation ( $r=.57$ ,  $p<0.05$ ) between the Overall components degree level and Cost efficiency. The averaged Specific components measure shows a statistically significant correlation positive as well ( $r=.56$ ,  $p<0.05$ ). Games with a higher level of overall reuse score better on Cost efficiency. Hypotheses 1(a) is there for supported in our model.

Hypothesis 1 (b) – Whether an increase in the degree of component reuse has a positive effect on Development time efficiency of the game was not supported as Table 8 shows that there is no statistically negative correlation between the Overall degree of components reuse and Development time efficiency. However we can't reject the null hypotheses yet until we examine the correlations between the Specific components attributes and Development time efficiency in more detail.

Hypotheses 1 (c-d) – Whether an increase in the degree of component reuse has a positive effect on product Quality and Profitability were also not supported in the analysis of Table 8 by looking at the overall reuse and Overall degree of components reuse measure. However again we can't reject the null hypotheses yet until we examine the correlations between the Specific components attributes and Quality and Profitability in Table 10.

Table 10. Descriptive statistics and correlation coefficients for the project performance variables.

(N=16)	Mean	Std. Deviation	1	2	3	4	5	6	7	8	9	10	11
1. Artificial intelligence components	2.81	1.87	1.00										
2. Rendering components	5.44	1.86	.35	1.00									
3. Physics components	5.00	2.31	.43	.56*	1.00								
4. Animation components	4.75	2.21	.23	.63**	.52*	1.00							
5. Visual effects components	4.88	2.03	.06	.63**	.60**	.72**	1.00						
6. Audio components	5.31	1.99	.18	.70**	.67**	.69**	.92**	1.00					
7. Game specific subsystems	3.13	1.75	.38	.54*	.02	.41	.44	.54*	1.00				
8. Profitability	3.59	1.78	.05	-.17	.09	.35	.18	.08	.04	1.00			
9. Development time efficiency	4.06	2.06	.27	.42	.13	.58*	.24	.39	.41	-.03	1.00		
10. Cost efficiency	4.58	1.97	.33	.67**	.21	.58*	.21	.41	.56*	-.27	.82**	1.00	
11. Product Quality	4.97	1.57	-.39	-.04	-.34	.07	-.11	-.23	-.08	.22	.00	-.05	1.00

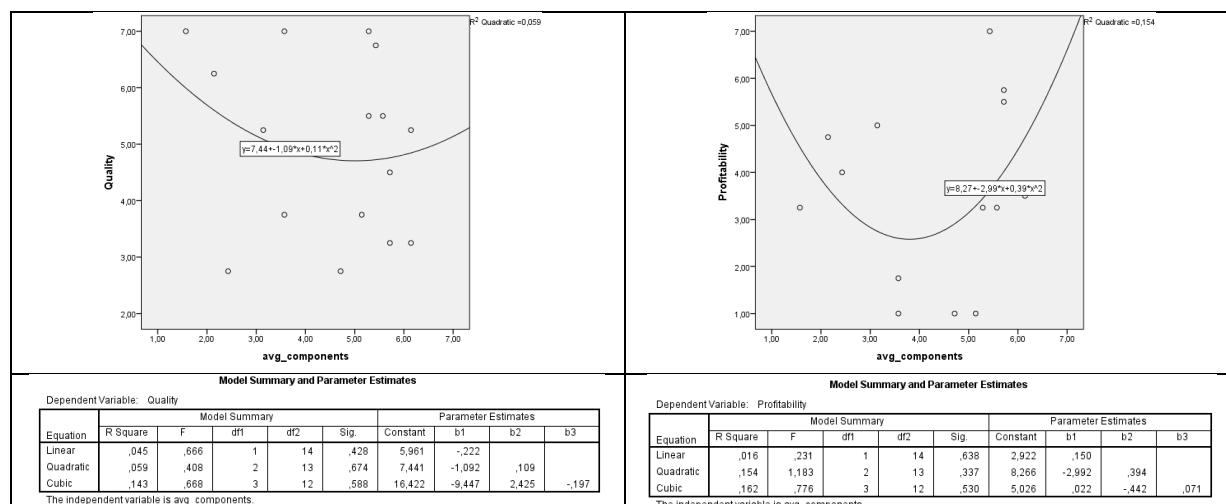
\*, Correlation is significant at the 0.05 level (2-tailed).

\*\*, Correlation is significant at the 0.01 level (2-tailed).

Table 10 supports the analysis in Table 8 with respect to the correlation between the degree of Component reuse and Cost efficiency. The Rendering and Animation components and Game specific subsystems have a significant positive relationship with Cost efficiency. The Rendering components show a statistically significant positive correlation with Cost efficiency ( $r=.67$   $p<0.01$ ). The Animation components show both a statistical positive correlation with Cost efficiency ( $r=.58$ ,  $p<0.05$ ) and Development time efficiency ( $r=.58$ ,  $p<0.05$ ). Therefore games with a higher level of Animation

components reuse score better on Cost efficiency and Development time efficiency. Hypothesis 1 (b) is there for supported in our model.

The results in Table 10 shows no statistical linear relationships between the degree of Specific component reuse and Product quality and Specific component reuse and Profitability. Therefore, hypotheses (1 c-d) were not supported. Further analysis using a polynomial term– a quadratic (squared) or cubic (cubed) term - which turns a linear regression model into a curve shows a slight curvilinear relationship between Specific components reuse and Quality.



The curvilinear relationship did not fit the data better than a linear equation ( $p=.428$ ) between the degree of Specific components reuse and Quality as can be seen in Table and Appendix C. A stronger curvilinear relationship is found between the degree of Specific components reuse and profitability. The results show that a quadratic equation ( $p=.337$ ) fits the data better than a linear equation ( $p=.638$ ). As the degree of specific components reuse increases, profitability decreases but only up to a certain point where we can see that as the degree of Specific components reuse increases, further profitability increases, leading to a clear U shape in the right plot above.

Lastly, hypothesis 1 (e) – Whether an increase in the degree of Overall reuse has a positive effect on Project performance was not supported as Table 8 shows that there are no statistically significant positive correlations between the Overall degree of software reuse and the four performance variables Cost efficiency, Development time efficiency, Quality and Profitability.



## 5.2.2 Specific Components reuse on Game Performance

Table 11. Descriptive statistics and correlation coefficients for the game performance variables.

(N=13)	Mean	Std. Deviation	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1. Overall degree of software reuse	4.77	1.30	1.00													
2. Overall reuse estimation system components	4.46	1.33	.84**	1.00												
3. Overall reuse estimation game assets	3.00	1.78	.40	.28	1.00											
4. Primary source of the total reuse portion	2.77	1.30	.31	.16	.36	1.00										
5. Specific Components	4.70	1.40	.82**	.94**	.24	.07	1.00									
6. Specific Assets	2.95	1.71	.34	.18	.95**	.43	.14	1.00								
7. Acting	3.82	.58	-.60*	-.33	.02	-.06	.02	.02	1.00							
8. AI	3.60	.51	-.34	-.11	-.11	-.55	.07	-.23	.89*	1.00						
9. Gameplay	3.71	.58	-.48	-.26	-.16	-.66*	-.17	-.27	.58*	.88**	1.00					
10. Graphics	3.81	.64	-.81*	-.39	-.14	-.50	-.29	-.25	.89**	.78**	.90**	1.00				
11. Personal slant	3.58	.67	-.49	-.26	.01	-.43	-.12	-.10	.72**	.84**	.88**	.93**	1.00			
12. Sound & Music	3.76	.53	-.63*	-.37	-.09	-.29	-.37	-.12	.87**	.68**	.81**	.82**	.82**	1.00		
13. Story & Presentation	3.49	.70	-.58*	-.39	.02	-.52	-.27	-.03	.77**	.78**	.87**	.92**	.95**	.86**	1.00	
14. Overall score	3.68	.55	-.60*	-.35	-.04	-.47	-.24	-.13	.82**	.86**	.92**	.95**	.96**	.91**	.97**	1.00

\*, Correlation is significant at the 0.05 level (2-tailed).

\*\*, Correlation is significant at the 0.01 level (2-tailed).

Hypothesis 1 (f) examines whether an increase in the degree of reuse of the Rendering components and the Visual Effects components has a positive effect on Graphics score. Table 11 shows a statistical significant negative correlation between Overall degree of software reuse and Graphics, Sound & Music, Story and presentation and the Overall game score ( $r = -.61$ ,  $p < 0.05$  and  $r = -.63$ ,  $p < 0.05$  and  $r = -.58$ ,  $p < 0.05$  and  $r = -.60$ ,  $p < 0.05$ ). However on the Overall component and averaged Specific components level there is no significant correlation with any of the Game performance variables. Again, we can't reject the null hypotheses yet until we examine the correlations between the Specific components attributes and the attributes of the Game performance scores.

Table 12. Descriptive statistics and correlation coefficients for the Game performance variables.

(N=13)	Mean	Std. Deviation	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1. Artificial intelligence components	3.23	1.83	1.00														
2. Rendering components	5.62	1.85	.32	1.00													
3. Physics components	5.23	2.28	.43	.48	1.00												
4. Animation components	4.92	2.18	.19	.53	.44	1.00											
5. Visual effects components	5.00	2.00	.00	.54	.51	.65*	1.00										
6. Audio components	5.46	1.76	.14	.67*	.55	.66*	.95**	1.00									
7. Game specific subsystems	3.46	1.76	.22	.52	-.15	.38	.43	.57*	1.00								
8. Acting	3.82	.58	.02	-.39	.24	.01	-.27	-.38	-.81**	1.00							
9. AI	3.60	.51	-.08	.06	.46	.10	.10	.09	-.53	.60*	1.00						
10. Gameplay	3.71	.58	-.21	-.21	.19	.01	-.08	-.12	-.58*	.56*	.88**	1.00					
11. Graphics	3.81	.64	-.29	-.31	.14	-.08	-.11	-.23	-.71**	.69**	.78**	.90**	1.00				
12. Personal slant	3.58	.67	-.25	-.10	.24	.13	-.01	-.10	-.65*	.72**	.84**	.88**	.93**	1.00			
13. Sound & Music	3.76	.53	-.08	-.42	.12	.00	-.35	-.44	-.82**	.67**	.68*	.81**	.82**	.82**	1.00		
14. Story & Presentation	3.49	.70	-.27	-.29	.06	.04	-.15	-.23	-.64*	.77**	.78**	.87**	.92**	.95**	.86**	1.00	
15. Overall score	3.68	.55	-.18	-.28	.20	.01	-.14	-.24	-.74**	.82**	.86**	.92**	.95**	.96**	.91**	.97**	1.00

\*, Correlation is significant at the 0.05 level (2-tailed).

\*\*, Correlation is significant at the 0.01 level (2-tailed).

Table 12 shows no significant statistical positive relationship between the degree of reuse of the Rendering components and Graphics score nor the Visual effects components and Graphics score. In fact, this relationship was slightly negative for the Rendering components and Visual effects components on Graphics score. Hypotheses 1 (e) is therefore rejected in our model. Further analysis indicated that these specific relations appeared not to be curvilinear and a linear relationship model best fitted the data (Appendix E).

Hypothesis 1 (g) – Whether an increase in the degree of reuse of AI components, Physics and Game specific subsystems has a negative effect on Gameplay score was supported for the effect of Game Specific subsystems on Gameplay scores, showing a statistically significant negative correlation ( $r = -.58$ ,  $p < 0.05$ ). In addition, Game Specific subsystems also show a statistical significant negative correlation with Acting, Graphics, Personal slant, Sound and Music, Story and Presentation and Overall score ( $r = -.81$ ,  $p < 0.01$  and  $r = -.71$ ,  $p < 0.01$  and  $r = -.65$ ,  $p < 0.05$  and  $r = -.82$ ,  $p < 0.01$  and  $r = -.64$ ,

$p < 0.05$  and  $r = -.74$ ,  $p < 0.01$ ). AI components however, was not significantly correlated with Gameplay score.

Lastly, Hypothesis 1 (h): Whether an increase in the Overall degree of software reuse and Overall degree of component reuse has a negative effect on Review scores was supported for the effect of Overall degree of software reuse on Review scores. Table 11 shows a statistical significant negative correlation between Overall degree of software reuse and Graphics, Sound & Music, Story and presentation and the Overall game score ( $r = -.61$ ,  $p < 0.05$  and  $r = -.63$ ,  $p < 0.05$  and  $r = -.58$ ,  $p < 0.05$  and  $r = -.60$ ,  $p < 0.05$ ). On the Overall component and averaged Specific components level there is no significant correlation with any of the Game performance variables.

### 5.2.3 Specific Game Assets on Project Performance

Table 13. Descriptive statistics and correlation coefficients for the Project performance variables.

(N=16)	Mean	Std. Deviation	1	2	3	4	5	6	7	8	9	10
1. Overall degree of software reuse	4.56	1.46	1.00									
2. Overall reuse estimation system components	4.25	1.39	.88**	1.00								
3. Overall reuse estimation game assets	2.94	1.77	.53*	.41	1.00							
4. Primary source of the total reuse portion	2.56	1.26	.40	.29	.37	1.00						
5. Specific Components	4.47	1.51	.86**	.96**	.39	.24	1.00					
6. Specific Assets	2.94	1.65	.45	.30	.96**	.41	.27	1.00				
7. Profitability	3.69	1.78	.11	.20	.06	-.26	.13	.04	1.00			
8. Development time efficiency	4.06	2.06	.33	.46	.34	.02	.46	.34	-.03	1.00		
9. Cost efficiency	4.58	1.97	.49	.57*	.16	.18	.56*	.17	-.27	.82**	1.00	
10. Product Quality	4.97	1.57	-.19	-.26	.14	-.40	-.21	.16	.22	.00	-.05	1.00

\*, Correlation is significant at the 0.05 level (2-tailed).

\*\*, Correlation is significant at the 0.01 level (2-tailed).

Hypotheses 2 (a-c) – Whether an increase in the degree of asset reuse has a positive effect on Cost efficiency, Development time efficiency and Profitability was not supported in the analysis of Table 13. Table 13 shows that there is no statistically significant correlation found on the Overall asset degree reuse level and averaged Specific game assets level on Development time efficiency and Cost efficiency.

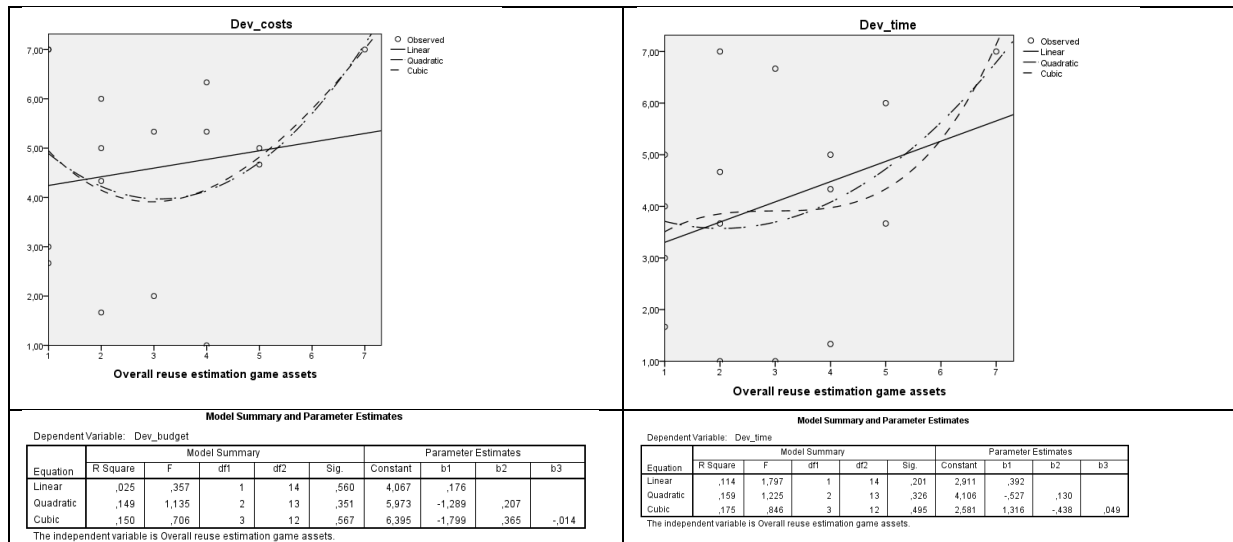
Table 14. Descriptive statistics and correlation coefficients for the Project performance variables.

(N=16)	Mean	Std. Deviation	1	2	3	4	5	6	7	8	9	9	10	11	12	13
1. Game objects	3.06	2.05	1.00													
2. Game levels	2.13	1.71	.63**	1.00												
3. (3D) Models	2.81	2.23	.81**	.83**	1.00											
4. Audio files	2.75	1.91	.67**	.78**	.64**	1.00										
5. Shaders	4.13	1.93	.37**	.58**	.50*	.41	1.00									
6. Scripts	3.00	1.93	.72**	.73**	.87**	.54*	.23	1.00								
7. Textures	3.13	2.22	.79**	.73**	.91**	.67**	.56*	.67**	1.00							
8. Materials	3.13	2.22	.67**	.70**	.80**	.56**	.65**	.58*	.92**	1.00						
9. Animations	3.13	2.25	.85**	.64**	.80**	.64**	.50*	.60*	.93**	.84**	1.00					
10. Story elements	2.13	1.78	.34	.61*	.34	.61*	.15	.46	.28	.20	.31	1.00				
11. Profitability	3.69	1.78	.32	-.06	.14	-.31	-.09	.17	.06	.03	.20	-.22	1.00			
12. Development time efficiency	4.06	2.06	.46	.42	.16	.57*	.18	.14	.11	.08	.27	.43	-.03	1.00		
13. Costs efficiency	4.58	1.97	.27	.30	.07	.44	.21	-.01	-.01	-.06	.04	.24	-.27	.82**	1.00	
14. Product Quality	4.97	1.57	.13	.26	.28	-.12	-.17	.40	.12	.07	.06	.35	.22	.00	-.05	1.00

\*, Correlation is significant at the 0.05 level (2-tailed).

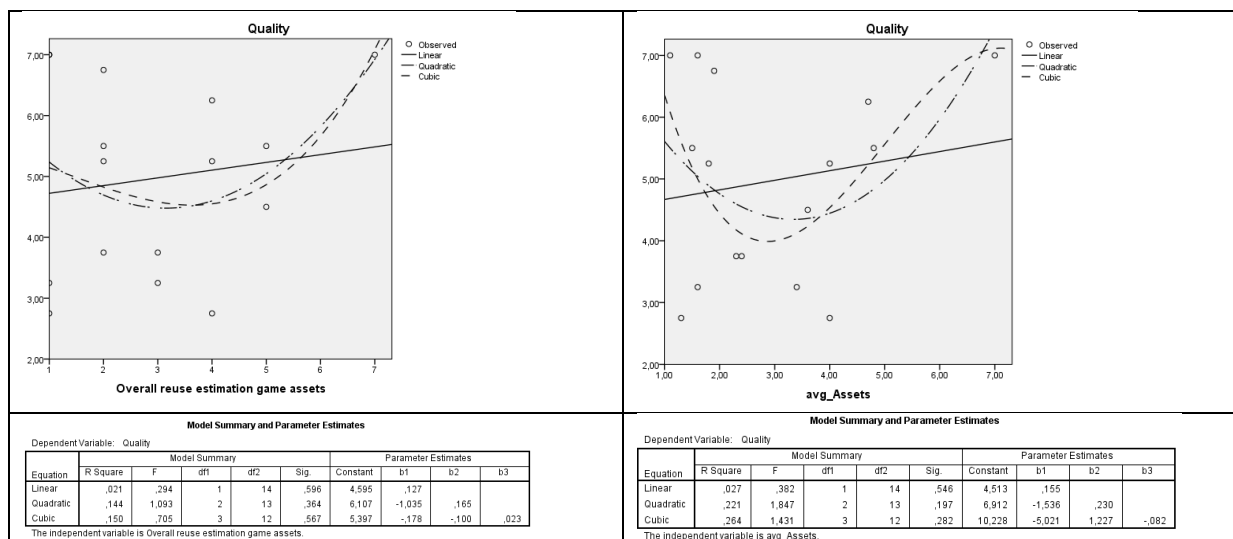
\*\*, Correlation is significant at the 0.01 level (2-tailed).

Further examination of the correlations between the Specific game assets and Project performance measures in Table 14 shows that only Audio files were significantly correlated with Development time efficiency ( $r = .57$ ,  $p < 0.05$ ). Therefore H2 (b) was supported for the degree of audio files reuse. Further analysis using a polynomial term shows a slight curvilinear relationship between Overall Asset reuse and Cost efficiency as can be seen on the next page.



The results show that a quadratic equation ( $p = .351$ ) fits the data slightly better than a linear equation ( $p = .560$ ) while this was not the case for Overall reuse of Assets and Development time efficiency.

Hypothesis 2 (d) - An increase in the degree of asset reuse has a positive effect on product quality was also rejected as Table 11 shows no statistically significant positive correlation on the Overall asset degree reuse level and Specific assets level on Product quality. Further analysis using a polynomial term shows a curvilinear relationship between Overall reuse of Assets and Quality and Specific reuse of Assets and Quality. The quadratic equation ( $p = .364$ ) fits the data slightly better than a linear equation ( $p = .596$ ). A stronger curvilinear relationship is found between the degree of Specific Assets reuse and Quality: The quadratic equation ( $p = .179$ ) fits the data better than a linear equation ( $p = .546$ ).



As the degree of Overall Game Assets reuse increases, Quality decreases but only up to a certain point where we can see that as the degree of Overall Game Assets reuse increases, Quality increases further, leading to a clear U shape in the right plot above. The same holds for the Specific assets and Quality.

## 5.2.4 Specific Game Assets on Game performance

Table 15. Descriptive statistics and correlation coefficients for the Game performance variables.

(N=13)	Mean	Std. Deviation	1	2	3	4	5	6	7	8	9	10
1. Overall reuse estimation game assets	3.00	1.78	1.00									
2. Assets	2.95	1.71	.95**	1.00								
3. Acting	3.82	.58	.02	.02	1.00							
4. AI	3.60	.51	-.11	-.23	.60*	1.00						
5. Gameplay	3.71	.58	-.16	-.27	.56*	.88**	1.00					
6. Graphics	3.81	.64	-.14	-.25	.69**	.78**	.90**	1.00				
7. Personal slant	3.58	.67	.01	-.10	.72**	.84**	.88**	.93**	1.00			
8. Sound & Music	3.76	.53	-.09	-.12	.87**	.68*	.81**	.82**	.82**	1.00		
9. Story & Presentation	3.49	.70	.02	-.03	.77**	.78**	.87**	.92**	.95**	.86**	1.00	
10. Overall score	3.68	.55	-.04	-.13	.82**	.86**	.92**	.95**	.96**	.91**	.97**	1.00

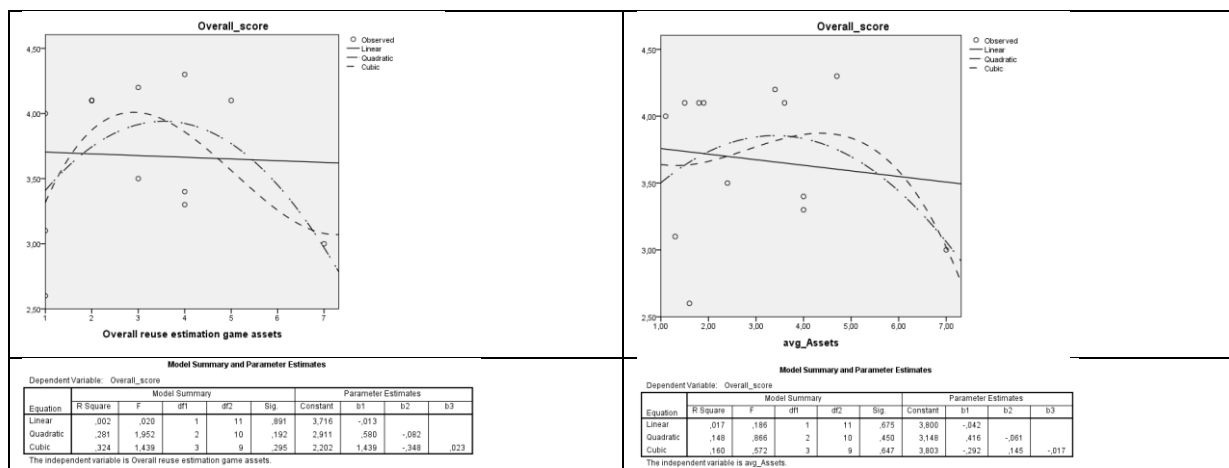
\*, Correlation is significant at the 0.05 level (2-tailed).  
 \*\*, Correlation is significant at the 0.01 level (2-tailed).

Hypothesis (2e) - Whether an increase in the degree of asset reuse has a negative effect on Review scores was not supported in the analysis of Table 15. Table 15 shows that there is no statistically negative significant correlation found on the Overall assets, Averaged specific assets and Game scores. Examination at the individual Specific Assets level on Game scores in Table 16 confirms that there is no statistical relationships found between the degree of asset reuse and Game performance. Hypothesis 2 (d) is there for not supported in our model.

Table 16. Descriptive statistics and correlation coefficients for the Game performance variables.

	Mean	Std. Deviation	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1. Game objects	3.31	2.10	1.00																	
2. Game levels	1.92	1.68	.73**	1.00																
3. (3D) Models	2.77	2.05	.85**	.85**	1.00															
4. Audio files	2.62	1.94	.81**	.79**	.77**	1.00														
5. Shaders	4.15	1.99	.31	.53	.42	.36	1.00													
6. Scripts	3.00	2.04	.72**	.74**	.88**	.59*	.14	1.00												
7. Textures	3.23	2.17	.77**	.77**	.90**	.80**	.49	.82**	1.00											
8. Materials	3.23	2.17	.63	.73**	.75**	.66*	.61*	.51	.89**	1.00										
9. Animations	3.38	2.33	.83*	.74**	.86**	.77**	.47	.58*	.96**	.84**	1.00									
10. Story elements	1.85	1.68	.73**	.96**	.81**	.85**	.41	.76**	.70**	.58*	.66*	1.00								
11. Acting	3.82	.58	.08	-.20	.07	-.21	-.13	.08	.15	.27	.23	-.31	1.00							
12. AI	3.60	.51	-.12	-.30	-.19	-.43	-.24	-.16	-.13	-.13	-.01	-.36	.60*	1.00						
13. Gameplay	3.71	.58	-.20	-.30	-.21	-.45	-.27	-.18	-.18	-.19	-.05	-.38	.56*	.88**	1.00					
14. Graphics	3.81	.64	-.23	-.30	-.20	-.52	-.15	-.15	-.16	-.01	-.08	-.45	.69**	.79**	.90**	1.00				
15. Personal slant	3.58	.67	-.08	-.16	-.05	-.34	-.06	-.02	-.04	.07	.04	-.28	.72**	.84**	.88**	.93**	1.00			
16. Sound & Music	3.76	.53	-.08	-.26	-.09	-.28	-.24	-.09	.00	.06	.13	-.35	.87**	.68*	.81**	.82**	.82**	1.00		
17. Story Presentation	3.49	.70	.01	-.08	.06	-.30	-.20	.16	.01	.08	.11	-.20	.77**	.79**	.87**	.92**	.95**	.86**	1.00	
18. Overall score	3.68	.55	-.08	-.23	-.06	-.38	-.19	-.04	-.02	.05	.08	-.35	.82**	.86**	.92**	.95**	.96**	.91**	.97**	1.00

\*, Correlation is significant at the 0.05 level (2-tailed).  
 \*\*, Correlation is significant at the 0.01 level (2-tailed).



Further analysis using a polynomial term shows a slight curvilinear relationship between Overall reuse of Assets reuse and Quality and Specific reuse of Assets reuse and Game score. The results show that a quadratic equation ( $p = .337$ ) fits the data better than a linear equation ( $p = .179$ ) for the relationship between Overall Assets and Game score and the Specific assets and Game score. As the degree of Overall Game Assets reuse increases, Overall Game score increases but only up to a certain point where we can see that as the degree of Overall Game Assets reuse increases, Overall Game score

decreases, leading to a clear inverted-U shape. The same holds for the Specific assets and Game score.

## 5.2.5 Internal and External software reuse.

Table 17. Descriptive statistics and correlation coefficients for the Game performance variables.

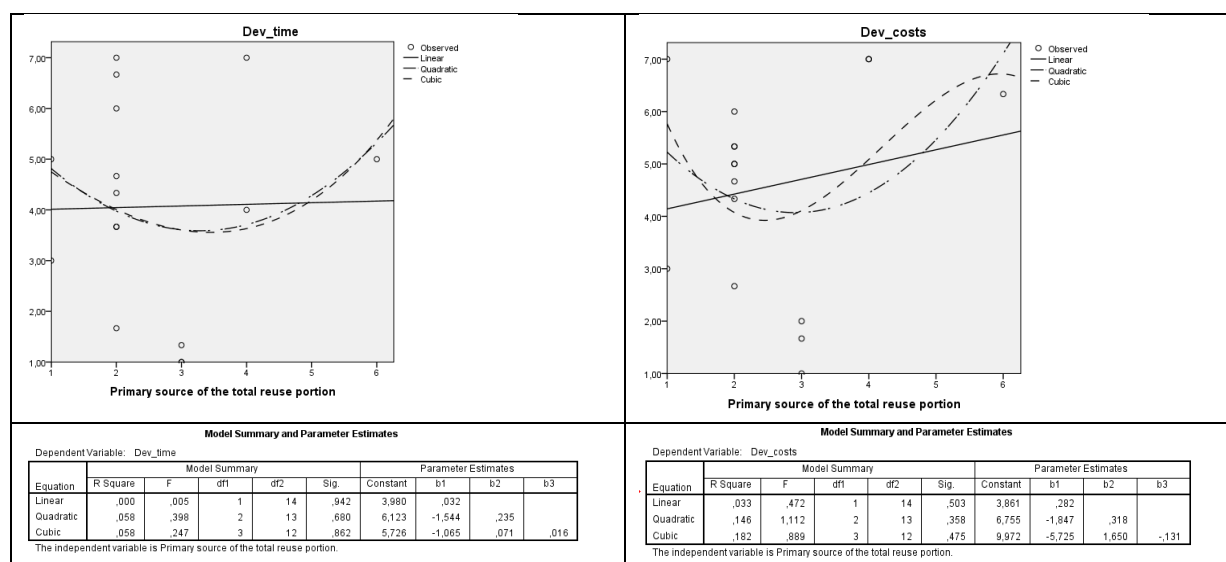
N=16	Mean	Std. Deviation	1	2	3	4	5
1. Primary source of the total reuse portion	2.56	1.26	1.00				
2. Profitability	3.59	1.78	-.26	1.00			
3. Development time efficiency	4.06	2.06	.02	-.03	1.00		
4. Cost efficiency	4.58	1.97	.18	-.27	.82**	1.00	
5. Product Quality	4.97	1.57	-.40	.22	.00	-.05	1.00

\*. Correlation is significant at the 0.05 level (2-tailed).

\*\*. Correlation is significant at the 0.01 level (2-tailed).

Hypotheses 3 (a-b) examines whether an increase in the degree of external reuse has a positive effect on Development time efficiency and Cost efficiency.

Table 17 shows no significant negative relation between the degree of External reuse and Development time efficiency and Cost efficiency. Hypotheses 3 (a-b) is therefore not supported in our model. Table 17 also shows that the degree of external reuse correlated negatively with Profitability and Quality. However this relation was not statically significant.



As the degree of External reuse increases, Cost efficiency decreases but only up to a certain point where we can see that as the degree of External reuse increases, Cost efficiency increases causing the inverted-U shape.

Table 18. Descriptive statistics and correlation coefficients for the Game performance variables.

N=13	Mean	Std. Deviation	1	2	3	4	5	6	7	8	9
1. Primary source of the total reuse portion	2,77	1,30	1,00								
2. Acting	3,82	,58	-,06	1,00							
3. AI	3,60	,51	-,55	,60*	1,00						
4. Gameplay	3,71	,58	-,66*	,56*	,88**	1,00					
5. Graphics	3,81	,64	-,50	,69**	,78**	,90**	1,00				
6. Personal slant	3,58	,67	-,43	,72**	,84**	,88**	,93**	1,00			
7. Sound & Music	3,76	,53	-,29	,87**	,68**	,81**	,82**	,82**	1,00		
8. Story & Presentation	3,49	,70	-,52	,77**	,78**	,87**	,92**	,95**	,86**	1,00	
9. Overall score	3,68	,55	-,47	,82**	,86**	,92**	,95**	,96**	,91**	,97**	1,00

\*, Correlation is significant at the 0.05 level (2-tailed).

\*\*, Correlation is significant at the 0.01 level (2-tailed).

Table 18 considers the relationship between External reuse and Game performance. The Overall degree of External reuse correlated negatively with Review scores, Gameplay score showing a statistically significant negative relationship ( $r = -.66$ ,  $p < 0.05$ ). Hypothesis 3 (c): Which posit that an increase in the degree of external reuse has a negative effect on Review scores is therefore confirmed.

### 5.2.6 Low vs. high reuse and the effect on project and game performance.

ANOVA was used to understand if there are any real differences in Project and Game performance based on the amount of reuse. We differentiated between a low and high reuse mode and tested the direct effect on the Project and Game performance variables. Our ANOVA results (Appendix F) show that Overall low/high reuse did not significantly differ from each other when considered jointly on the Project performance variables. This means that the groups of low overall reuse, and high overall reuse did not score significantly different on project performance, Wilk's  $\lambda = .379$ ,  $F(4,11)=1.16$ ,  $p = .38$ , partial  $\eta^2 = .30$ . At the univariate level, the separate ANOVA's for each dependent variable also showed no statistically significant results.

Further analysis indicate a significant difference between low/high reuse of Overall components when considered jointly on the Project Performance variables, Wilk's  $\lambda = .390$ ,  $F(4,11) = 4.29$ ,  $p = .025$ , partial  $\eta^2 = .610$ . A separate ANOVA at the univariate level was conducted for each dependent variable, with each ANOVA evaluated at the 0.025 level. An alpha correction is made to account for the multiple ANOVA's being run (using the Bonferroni correction) thus we accept statistical significance at  $p < .025$ .

There was a significant difference between low/high Overall component reuse on Development time efficiency,  $F(1,14)=5.59$ ,  $p = .033$ , partial  $\eta^2 = .285$  at the 0.05 alpha level, with high reuse scoring higher ( $M = 5.00$ ) than low reuse ( $M = 2.857$ ). However since we only accept statistical significance at  $p < .025$  using the Bonferroni adjusted level of alpha this result was not truly significant.

There was also a significant difference between low/high Overall component reuse on Cost efficiency,  $F(1,14)=7.00$ ,  $p = .019$ , partial  $\eta^2 = .333$ . Also, since this  $p$  value is lower than  $p < .025$  we can say that a low/high Overall component reuse differ on Cost efficiency with high reuse scoring higher ( $M = 5.56$ ) than low reuse ( $M = 3.33$ ).

Further examination of the other dependent variables indicated that there were no significant differences between low and high Overall component reuse on Profitability and Quality. There was also no significant difference between low/high reuse of Overall game assets when considered jointly on the Project Performance variables, Wilk's  $\lambda = .895$ ,  $F(4,11) = .32$ ,  $p = .858$ , partial  $\eta^2 = .105$ .

At the Specific components level, there was a significant difference between low/high reuse when considered jointly on the Project Performance variables, Wilk's  $\lambda = .356$ ,  $F(4,11) = 4.99$ ,  $p = .015$ , partial  $\eta^2 = .64$ . A separate ANOVA was conducted for each dependent variable, with each ANOVA evaluated at the 0.025 level. There was no significant difference between low/high specific component reuse on profitability,  $F(1,14) = 4.89$ ,  $p = .044$ , partial  $\eta^2 = .259$ , with high reuse scoring higher ( $M = 4.47$ ) than low reuse ( $M = 2.72$ ). Further examination of the other dependent variables indicated that there were no significant differences between low and high reuse of the Specific components on Development time efficiency, Cost efficiency and Quality. There was also no significant difference between low/high reuse of Specific assets when considered jointly on the Project Performance variables, Wilk's  $\lambda = .895$ ,  $F(4,11) = .32$ ,  $p = .858$ , partial  $\eta^2 = .105$ .

There was also no significant difference between internal/external reuse when considered jointly on the Project Performance variables, Wilk's  $\lambda = .830$ ,  $F(4,11) = .56$ ,  $p = .694$ , partial  $\eta^2 = .170$ . At the univariate level, the separate ANOVA's for each dependent variable also showed no statistically significant results. There was also no significant difference between internal/external reuse groups when considered jointly on the Game Performance variables, Wilk's  $\lambda = .498$ ,  $F(8,4) = .503$ ,  $p = .810$ , partial  $\eta^2 = .502$ . At the univariate level, the separate ANOVA's for each dependent variable showed no statistically significant results.

At the Game performance level, there was no significant difference between Overall low/high reuse when considered jointly on the Game Performance variables, Wilk's  $\lambda = .432$ ,  $F(8,4) = .66$ ,  $p = .715$ , partial  $\eta^2 = .59$ . At the univariate level, the separate ANOVA's for each dependent variable showed no statistically significant results.

Neither was there a significant difference between low/high Overall components reuse when considered jointly on the Game Performance variables, Wilk's  $\lambda = .543$ ,  $F(8,4) = .421$ ,  $p = .862$ , partial  $\eta^2 = .457$ . Also at the univariate level, the separate ANOVA's for each dependent variable showed no statistically significant results.

Lastly, ANOVA results showed no significant difference between low/high Overall assets reuse when considered jointly on the Game Performance variables, Wilk's  $\lambda = .417$ ,  $F(8,4) = .700$ ,  $p = .691$ , partial  $\eta^2 = .583$ . Also at the univariate level, the separate ANOVA's for each dependent variable showed no statistically significant results.

At the Specific assets level there was no significant difference between low/high reuse when considered jointly on the Game Performance variables. Wilk's  $\lambda = .417$ ,  $F(8,4) = .700$ ,  $p = .691$ , partial  $\eta^2 = .583$ . Also at the univariate level, the separate ANOVA's for each dependent variable showed no statistically significant results. There was no significant difference between low/high reuse when considered jointly on the Project Performance variables. Wilk's  $\lambda = .895$ ,  $F(4,11) = .32$ ,  $p = .858$ , partial  $\eta^2 = .105$ .

### 5.2.7 Control variables

We also tested whether existence of Low / high degree of Systematic reuse in the firm, using an Middleware / Internal game engine or and having a small ( $n = 8$ ) or large development team ( $n = 8$ ) would have a different effect on the amount of Reuse and the Product and Project performance outcomes using ANOVA. Using a Middleware / Internal game engine and having a small / large team size reported the same level of effects on the amount of Reuse and the Product and Project performance outcomes and these categories were not statistically different from each other.

Employing a high level of Systematic reuse process in the firm did show significant differences in performance outcomes. The analysis show that a Low/high systematic reuse process did differ on Game Scores with a Low systematic reuse process scoring significantly higher than a High systematic reuse process on all the Game performance variables except for the Acting review score. In all these cases the p value is lower than  $p < .025$ , thus we accept a true significant difference in Game performance between the two groups. This result is interesting for developers because although a high level of Systematic reuse is positively related to Cost efficiency (Table 8) it does not necessary result in a 'good' game as indicated by the lower Review scores in table 19. Firms employing a high level of systematic reuse scored significantly lower on AI, Gameplay, Graphics, Personal Slant, Sound & Music, Story & Presentation and overall Game score when if they employed a low level of Systematic reuse process.

Table 19 shows also that there was a nearly statistically significant difference in Overall degree of software reuse, but as we only accept statistical significance at  $p < .025$  using the Bonferroni adjusted level of alpha this result was not truly significant.

	Low Systematic reuse (n= 7)		High Systematic reuse (n=9)		Sig.	Partial $n^2$
	Mean	SD	Mean	SD		
<b>Degree of reuse</b>						
Overall degree of software reuse	3.71	1.38	5.22	1.20	.035	.280
Overall reuse estimation components	3.71	1.604	4.67	1.12	.183	.123
Overall reuse estimation Game Assets	2.71	1.38	3.11	2.09	.672	.013
Specific components	3.92	1.76	4.90	1.21	.204	.113
Specific Asset	2.70	1.22	3.12	1.97	.628	.017
<b>Project Performance</b>						
Profitability	4.07	.67	3.22	.59	.361	.60
Development time efficiency	4.10	.80	4.04	.71	.957	.000
Cost efficiency	3.81	.72	5.19	.64	.174	.128
Quality	5.07	.61	4.89	.54	.826	.004
<b>Game Performance</b>	Low Systematic reuse (n= 5)		High Systematic reuse (n= 8)		Sig.	Partial $n^2$
	Mean	SD	Mean	SD		
Acting	4.16	.24	3.61	.19	.097	.230
AI	4.02	.18	3.34	.14	<b>.012</b>	.454
Gameplay	4.26	.17	3.36	.13	<b>.002</b>	.615
Graphics	4.36	.21	3.46	.17	<b>.006</b>	.507
Personal Slant	4.12	.23	3.24	.18	<b>.012</b>	.448
Sound & Music	4.18	.29	3.50	.15	<b>.016</b>	.422
Story & Presentation	4.02	.26	3.16	.20	<b>.024</b>	.385
Overall Score	4.16	.18	3.38	.14	<b>.006</b>	.515

**Table 19. Low systematic reuse vs. High systematic reuse.**



## 6. Discussion

### 6.1 Overview

This study investigated the effects of software reuse on different game performance outcomes and project performance outcomes. Specifically, we examined the relationships between different levels of software reuse including 7 specific Game Components and 10 specific Game Assets on project outcomes (Cost efficiency, Development time efficiency, Quality and Profitability) and Game performance outcomes (Review scores).

We started the study with an extensive literature review on the concepts of software reuse and product modularity and did an initial exploratory field study through several interviews with software and game developers. The outcome of this exploratory study led to the development of a questionnaire, development of hypotheses and the identification of variables and research model.

Our research model was empirically tested with data from a random sample of 124 games that were targeted for this study and that were published during 2010 – 2014 on any game platform (pc and consoles).

Based on the hypotheses that were tested in our study we conclude that:

- Games with a higher level of Overall component reuse tend to score better on Cost efficiency Hypothesis 1 (a) is there for supported in our model. There is a statistically significant positive correlation ( $r=.57$ ,  $p<0.05$ ) between the Overall components degree level and Cost efficiency. The averaged Specific components measure shows a statistically significant positive correlation as well ( $r =.56$ ,  $p<0.05$ ). This result imply that higher levels of cost efficiency can be achieved when reusing major parts of components.
- Hypothesis 1 (b) - Whether an increase in the degree of component reuse has a positive effect on Development time efficiency is supported. Looking at the specific components level, Rendering components show only a statistically significant positive correlation with Cost efficiency ( $r= .67$   $p<0.01$ ). The Animation components however shows both a statistical positive correlation with Cost efficiency ( $r= 0.58$ ,  $p<0.05$ ) and Development time efficiency ( $r= 0.58$ ,  $p<0.05$ ). This finding suggest that these specific components in particular are worth investigating for managers to improve Development time efficiency or Cost efficiency.
- Hypotheses (1 c-d) are not supported: An increase in the degree of Overall component reuse and Specific component reuse does not have a positive effect on product Quality and Profitability. A curvilinear relationship is found between the degree of Specific components reuse and profitability. The results show that a quadratic equation ( $p= .337$ ) fits the data better than a linear equation ( $p = .638$ ). As the degree of specific components reuse increases, profitability decreases but only up to a certain point where we can see that as the degree of Specific components reuse increases, further profitability increases. This finding suggests that Specific components should either have a minimum level of reuse or be reused in full for maximum profitability.
- Hypothesis 1 (e) – Whether an increase in the degree of Overall software reuse has a positive effect on Project performance was not supported as Table 8 shows that there are no statistically significant positive correlations between the Overall

degree of software reuse and the four project performance variables Cost efficiency, Development time efficiency, Quality and Profitability. This finding underscores the importance of investigating other granularities of software reuse, such as game Components and Game Assets and not just overall software reuse.

- Hypothesis 1 (f) examines whether an increase in the degree of reuse of the Rendering components and the Visual Effects components has a positive effect on Graphics score and is rejected. We found a statistically significant negative correlation between Overall degree of software reuse and Graphics, Sound & Music, Story and presentation and the Overall game score ( $r = -.61, p < 0.05$  and  $r = -.63, p < 0.05$  and  $r = -.58, p < 0.05$  and  $r = -.60, p < 0.05$ ). However on the Overall, Averaged Specific- and individual Specific components level there is no significant correlation with any of the Game performance variables. This finding suggests that Specific components should either have a minimum level of reuse or be reused in full for maximum profitability.
- Hypothesis 1 (g) – Whether an increase in the degree of reuse of AI components, Physics and Game specific subsystems has a negative effect on Gameplay score is supported for the effect of Game Specific subsystems on Gameplay scores. We found a significant negative correlation between the degree of Game Specific subsystems reuse and Gameplay scores ( $r = -.58, p < 0.05$ ). In addition, Game Specific subsystems also show a statistical significant negative correlation between with Acting, Graphics, Personal slant, Sound and Music, Story and Presentation and Overall score. The results imply that more reuse of these specific components in particular works negatively towards review scores and addresses that tailoring Game specific components to the game is important to achieve higher review scores.
- Hypothesis 1 (h) - Whether an increase in the Overall degree of software reuse and increase in the Overall degree of component reuse has a negative effect on Review scores was only supported for the effect of Overall degree of software reuse on Review scores. A significant negative correlation is found between the Overall degree of software reuse and Graphics, Sound & Music, Story and presentation and the Overall game score ( $r = -.61, p < 0.05$  and  $r = -.63, p < 0.05$  and  $r = -.58, p < 0.05$  and  $r = -.60, p < 0.05$ ). This implies that a high degree of software reuse in general and in different levels of granularity can have a negative effect on review scores and should be something to constantly consider while applying software reuse.
- Hypotheses 2 (a-c) – whether an increase in the degree of asset reuse has a positive effect on Cost efficiency, Development time efficiency and Profitability is not supported for the Overall asset degree reuse level and averaged Specific game assets level on Development time efficiency and Cost efficiency. Further analysis using a polynomial term shows a slight curvilinear relationship between Overall reuse of Assets reuse and Cost efficiency. A quadratic equation ( $p = .351$ ) fits the data slightly better than a linear equation ( $p = .560$ ) while this was not the case for Overall reuse of Assets and Development time efficiency.
- Hypothesis 2 (b) is supported for the degree of audio files reuse. Examination of the correlations between the Specific game assets and Project performance measures in Table 14 shows that only Audio files were significantly correlated with Development time efficiency ( $r = .57, p < 0.05$ ). This result is somewhat counterintuitive as we expected more game assets types with a higher level of reuse would significantly lead to better Development time efficiency. Game

objects ( $r = .46$ ) and Game levels ( $r = 0.42$ ) were also positively related to Development time efficiency but the results were not significant.

- Hypothesis 2 (d) - Whether an increase in the degree of asset reuse has a positive effect on product Quality is also rejected. Further analysis using a polynomial term shows a curvilinear relationship between Overall reuse of Assets reuse and Quality and the Specific reuse of Assets reuse and Quality. The quadratic equation ( $p = .364$ ) fits the data slightly better than a linear equation ( $p = .596$ ). A stronger curvilinear relationship is found between the degree of Specific Assets reuse and Quality and the quadratic equation ( $p = .179$ ) fits the data better than a linear equation ( $p = .546$ ). This finding suggests that for a maximum level of Quality, assets should have a very low level of reuse or be reused in full for maximum Quality.
- Hypothesis 2 (e) - Whether an increase in the degree of asset reuse has a negative effect on Review scores is not supported. Examination at the Overall and individual Specific Assets level on Game scores confirms that there is no statistical linear relationship found between the degree of asset reuse and Game performance. A quadratic equation ( $p = .337$ ) fits the data better than a linear equation ( $p = .179$ ) for the relationship between Overall Assets and Overall Game score and the Specific assets and Game score. The result imply that a minimum level asset reuse is accepted and has a positively effect on Game score but after a certain point, too much reuse without modification will negatively affect Game score. It is therefore important to try not to reuse Game Assets in full and introduce enough modification in the Game Assets.
- Hypotheses 3 (a-b) - Whether an increase in the degree of external reuse has a positive effect on Development time efficiency and Cost efficiency is not supported. There is no significant negative relation between the degree of External reuse and Development time efficiency and Costs efficiency. The overall degree of External reuse correlated negatively with Review scores, Gameplay score showing a statistically significant negative relationship ( $r = -.66$ ,  $p < 0.05$ ). This finding suggest that more external reuse does not introduce benefits in project performance over internal reuse, in fact external reuse correlated slightly negatively with Quality ( $r = -.40$ ) and profitability ( $r = -.26$ ). Potential costs may lie in the possibility of still having to adapt the external software and integrating it into the product under development depending on the requirements. If adapting the externally reused artifact is not thoroughly executed this could negatively impact Quality.
- Hypothesis 3 (c): An increase in the degree of external reuse has a negative effect on review scores is confirmed. On the Overall software reuse level, External reuse is found to correlate negatively with all Review score criteria and Gameplay score showing a statistically significant negative relationship ( $r = -.66$ ,  $p < 0.05$ ). This finding implies that a higher level of external reuse works negatively towards review scores. Again, if the externally reused artifact is not thoroughly adapted, modified and integrated into the game this could negatively impact the quality and lead to lower review scores. Also any external reused artifact might prove more effective if it is well documented, generalized and already of high quality.

In sum, the results from our analysis indicate that there are significant statistical correlations between the factors of software reuse and project and game performances outcomes. The study also shows there is a significant difference between the Low and High reuse modes of Overall components and Project Performance variables. Firms applying a high mode of Overall component reuse scored better on Development time efficiency and Cost efficiency. In this study we found no statistical significant differences in software reuse and project performance when using a Middleware or Internal game engine or having a small or large development team.

The overall degree of external reuse correlated negatively with profitability and quality, however this relationship was not significant. The overall degree of External reuse also correlated negatively with Review scores, Gameplay score showing a statistically significant negative relationship.

Lastly, it was concluded that employing a high level of Systematic reuse process resulted in significant differences in performance outcomes. The study results show that a Low/high systematic reuse process differ on Game Scores with a Low systematic reuse process scoring higher than a High systematic reuse process on AI, Gameplay, Graphics, Personal Slant, Sound & Music, Story & Presentation and overall Game score.

## **6.2 Theoretical implications**

The main contribution of this study is that it provides one of the first empirical investigation of the effects of different degrees and forms of reuse on project and game performance outcomes. From a theoretical perspective this study supports that software reuse has a positive impact on Development time efficiency and Cost efficiency in a game development context. Games with a higher level of Overall reuse tend to score better on Cost efficiency. These results are consistent with other software reuse-oriented studies that examined the effect of software reuse on project performance (V.R. Basili et al., 1996); (de O. Melo et al., 2013); (W. Frakes & Terry, 1996); (Ajila & Wu, 2007) and component based development (Schach, 2011, p. 594).

There are very few empirical studies and conclusions drawn about the application of software reuse within the context of the gaming industry and the effect on product- and project performance. To our knowledge, no earlier study explored facets of software reuse commonly seen in game development and analyzed their effects on different game performance outcomes and project performance outcomes. Our study is thus able to extend and contribute to our current knowledge already known from software development articles and books covering game development aspects such as programming (Rollings & Morris, 2004); (Gregory, 2009); (Schmidt et al., 2007) project-management and game design (McGuire & Chadwicke Jenkins, 2008).

In addition, we think we have added value by studying various game engine architectures and identifying specific components or specialized game domains that we found present across several game engine architectures. These components can be seen as parts of a software system that are identifiable and reusable. While traditional software studies mainly focused on measuring abstract forms of software reuse such as lines of code, function points, procedures, classes and design patterns, we specifically targeted our study around the concepts of Game Components and Game Assets. These specific software parts, often functioning as executable units of independent production, acquisition, and deployment can be composed into a functioning game system. We think

it should give game developers more practical value to research these particular domains typically seen in game development over the generally known software abstractions to further base their engine-technology and component sourcing strategies upon. Furthermore, our study contributes reliable and valid measures for software reuse constructs related specific for the creation of video games that should be useful for future research on game development.

Our results extend the findings by (V.R. Basili et al., 1996); (W. Frakes & Terry, 1996); (Ajila & Wu, 2007) where the effects of software reuse on productivity was examined and where it was concluded that software reuse has a strong impact on productivity, product quality and defect density. In line with their findings, our study showed that game development companies applying a high mode of Overall component reuse scored better on Development time efficiency and Cost efficiency. Surprisingly our findings do challenge these authors' findings regarding their conclusions about Product quality. In our study Overall reuse, Component and Asset reuse did not lead to higher levels of product quality, in fact a higher reuse level turns out to work many times slightly negative towards quality. This contradicts current findings in the literature regarding software reuse and the effect on quality. This finding is unsuspected as literature argues that reusing high quality artifacts that have been tested should deliver higher levels of quality, then when building it from scratch. We measured product quality partly as the perceived quality by the respondents as we could not exactly measure quality in terms of real defect density (number of faults and errors / software size) due to practical reasons and time limitations but which the other studies did do. In a future study we could try to add and operationalize these quality in terms of defect density like other studies did and verify if it introduces different results in the level of quality.

In our study Systematic reuse was found to be positively related to Project performance in terms of Cost efficiency and these findings are in line with various studies on software reuse (W. Frakes & Terry, 1996); (Ezran et al., 2002); (Victor R. Basili, 1990); W. B. Frakes & Isoda, 1994). To achieve significant payoffs a reuse program must be systematic (W. B. Frakes & Isoda, 1994) and organizations implementing systematic software reuse programs must be able to measure their progress and identify the most effective reuse strategies. Our study thus confirms and underscores the importance of having a structured, systematic reuse process that is applied and integrated in the firms development process, using databases to list standard components and having flexible means for combining components through standard interfaces among modules can help game development companies to achieve substantial benefits in Cost efficiency and Development time efficiency.

Although a high level of Systematic reuse was found to be positively related to Project performance in terms of Cost efficiency it does not necessary result in a 'good' game product as indicated by the lower Review scores. Firms employing a high level of systematic reuse scored significantly lower on AI, Gameplay, Graphics, Personal Slant, Sound & Music, Story & Presentation and overall Game score when if they employed a low level of Systematic reuse process. These findings are in line with our earlier expectations about using unmodified software in games. ("Visceral Games Speaks Out on Battlefield Hardline Re-Using Battlefield 4 Assets," 2014). If certain Game Components and Game Assets make it unmodified into the game this could negatively affect the game scores, for example too much reuse of visible elements could make the levels look generic and less diverse or negatively affect the users experience, ultimately negatively affecting review scores.

Because a Low systematic reuse process scores significantly higher than a High systematic reuse process on all the Game performance variables except for the Acting review score it is reasonable to think that other factors such as the amount of specific tweaking, optimizing or the way Game Assets and Components are integrated is also important for good review scores or higher product quality. This area offers new research possibilities to be explored in a future study on game development.

### **6.3 Managerial implications**

Our study results could be of great advantage to different entities like independent developers, large game development studios or game publishers that are considering making a game or are in the process of making a game.

Direct implications for game developers and game directors are that they not only need to pay attention the specific components and assets they are reusing but also to the degree they are reusing it. Our findings show that there are significant statistical correlations between the different Overall reuse and Specific reuse factors and project and game performances outcomes. Looking at the specific components level, Rendering components show a statistically significant positive correlation with Cost efficiency. The Animation components shows both a significant statistical positive correlation with Cost efficiency and Development time efficiency.

There is a statistical significant negative correlation between Overall degree of software reuse and Graphics, Sound & Music, Story and presentation and the Overall game score. The results also shows there is a significant difference between the Low and High reuse modes of Overall components and Project Performance variables. Firms applying a high mode of Overall component reuse scored better on Development time efficiency and Cost efficiency. Game Assets are important as well for managerial investigation and considerations. The overall degree of external reuse correlated negatively with profitability and quality, however this relationship was not significant. The overall degree of External reuse also correlated negatively with Review scores, Gameplay score showing a statistically significant negative relationship. Game developers and game directors should consider said effects cautiously during development, e.g. during architectural design and while sourcing specific components or assets for their game.

Another implication is that managers need to find a certain balance in the levels of reuse, as too much reuse can negatively affect project and performance outcomes. Our study showed for example for Asset reuse that as the degree of Overall Game Assets reuse increases, Overall Game score increases but only up to a certain point where we can see that as the degree of Overall Game Assets reuse increases, Overall Game score decreases. The same is concluded for the Specific assets and Game score.

What this implicates is that a too high level of reuse can negatively impact Game score. Game developers should therefor wisely consider what assets to reuse, update or rebuild from scratch. If certain assets make it unmodified into the game this could negatively affect the review scores of the game. The game design workflow may be further improved to allow maximum efficiency and flexibility for level designers and artist. Artists should have a workflow in which they are able to create unique assets or groups of assets in an easy and efficient way which level designers should then easily be able to grab, use or recombine into new assets while adding the right amount of changes to each asset.

Also when the Overall Game Assets reuse increases, Quality decreases but only up to a certain point where we can see that as the degree of Overall Game Assets reuse increases, Quality increases further, the same holds for the Specific assets and Quality. Too much reuse however, - as we see with firms that employ a high level of systematic reuse - could eventually negatively impact Game performance leading to lower Game scores. Game developers should thus be aware that a reuse strategy that is focused on reusing software where possible should be generally considered as bad practice. In sum, game developers, Technical- and Art directors should wisely consider and interpreted our study results and analyze and compare their own specific reuse choices and the effects on development time and development costs and Game score. By implementing a systematic reuse process they can achieve substantial benefits in Cost efficiency but they must keep in mind that although a high level of Systematic reuse is positively related to better Cost efficiency it does not necessary result in a 'good' game. Firms employing a high level of systematic reuse scored significantly lower on AI, Gameplay, Graphics, Personal Slant, Sound & Music, Story & Presentation and overall Game score when if they employed a low level of Systematic reuse process. Game developers should therefore find a balance in where and where not to follow a systematic reuse process in the different stages of game development and game design which should then be further integrated into the software development process.

In addition to having a systematic reuse process, component reuse can help to achieve significant higher levels of Cost efficiency. We found no statistically significant positive correlations between the Overall degree of software reuse and the four project performance variables Cost efficiency, Development time efficiency, Quality and Profitability. The result imply that applying a systematic component strategy that includes a high level of reuse of the Rendering components, Animation components and game specific subsystems can help firms to achieve higher levels of Cost efficiency. Also, as the reuse of Game specific components was significantly negatively correlated with review scores it underscores the importance and need of tailoring Game specific components to fit the game right in order to achieve higher review scores.

Understanding above reuse choices and their effect on product performance could help game developers to put emphasize on the right management efforts and financial resources in the different stages of software development. For example, the results can help managers deciding whether it is worth investing in particular game components such as the Rendering components, Animation components and Game specific subsystems or new game development methodologies in system- and game design to improve development time efficiency, cost efficiency or game quality.

## **6.4 Limitations and suggestions for future research**

A limitation of our study is that the measures for the independent variables and dependent variables project performances were based on the perceptions of the respondents and were single sourced. The answers are thus limited by the person's truthfulness and estimate accuracy that the person makes in answering the questions. This means their answers could be biased even though senior developers and managers should be able to oversee different development aspects fairly well. However our extensive literature review, exploratory field study through several interviews with game and software developers, pilot study of the preliminary survey and reliability analysis should provide some assurance of the instrument being able to capture useful measures. In a next study we could include both lead artists, lead developers and project managers answering only specific parts of the survey, now the complete survey was based on the perceptions of just one person with either a technical or less technical background. A homogeneity test of variance however pointed out that these two groups with different specialism did not give statistically different answers.

Another limitation is the very small sample size in this study. While this study hypothesized and found several associations between the variables of software reuse and project- and game performance variables, results need to be interpreted cautious due to small sample size and therefore a larger confirmatory study is very much required. A larger sample size provides more precise results because it has a small standard error leading to narrow 95% confidence intervals which gives us a more precise estimate of the effect and firm conclusions. With smaller sample sizes, such as in this study it is harder to distinguish between a real effect and random variation due to a large standard error and wide 95% confidence interval. Therefore we get imprecise estimates of the effects and thus less firm conclusions can be drawn. A future larger study, should provide us more reliable results and should be executed as a follow up study. This study should therefore be replicated in the gaming industry and academia to confirm the results and enhance and refine our research model. Replication of the study in the gaming industry can help managers to further investigating investments in certain game components to improve project and game performance. Replication in academia can help to compare reuse methods against the traditional forms of software reuse methods such as those that are mentioned in Chapter 2.2. Replication can also help to increase the study's external validity. Because this study primarily focused on reuse concepts specifically used in game development, external validity beyond the software and gaming industry is considered low and other reuse concepts may apply for different industries.

An interesting topic for future research would be to differentiate between different game engines such as games made in Unreal Engine or Unity3d or a proprietary game engine. This study did not differentiate between popular game engines as our study had too few respondents in the same game engine category. Knowing which game engine was associated with a better project or game performance would be interesting for developers as choosing a game engine is one of the first steps in creating a video game.

Another interesting research choice would be to differentiate between different game type such as Remakes, Sequels and Ports of games and analyze the degree of reuse and effects on game and project performance. It would be reasonable interesting as well to know the proportion of Game Components and Game Assets that were sourced externally. Using middleware components and assets seems like a new trend in game development, especially for smaller game studios as they use to have smaller teams and



less resources that big game development studios have. This study did not distinguish between the two on a component and asset level.

Because a Low systematic reuse process scores significantly higher than a High systematic reuse process on all the Game performance variables except for the Acting review score concepts such as Tweaking, Optimizing and the way how specific Game Assets and Components are being integrated could be further explored as this could potentially give further insight why a high degree of systematic reuse scores lower on review scores. This area offers new research possibilities to be explored in future studies on game development.

In spite of the aforementioned limitations, we think this study adds value to existing literature about videogame development and software reuse in a Gaming industry context. It emphasizes that reuse in general, component reuse and specific reuse can affect project- and game performance.

## Appendix A: Interview guide

Name	Function	Date	Company
Michael Angelo Groeneveld	Senior software developer	03/07/14	El-Nino B.V.
Teun Lassche	Senior software developer	05/07/14	T.H. Lassche Webdevelopment
Pieter van den Bosch	Jr. Software developer	20/07/14	Magdeveloper.com
Lee Bamber	Senior software developer	07/08/14	The game creators Ltd.
Hans Wichman	Tutor, independent game developer	15/08/14	Inner Drive Studios

### Main Questions:

- What are potential areas of software reuse in your company?
- What are the positive and negative effects of software reuse on different software development economics?
- What are the main decision making aspects when deciding between reuse or building something new?
- What are the general implementing problems/challenges developers face when implementing a middle ware component / external component.
- What concepts of programming/tweaking/tricks are involved in optimizing performance.

### **Interview: Lee Bamber, Co-founder The Game creators, 07/10/14**

Q1: How do you optimize a third party middle-ware software such as an AI or Physics system to best fit your game, if any? What are the general implementing problems and challenges developers face when implementing a middle ware component as, for example an AI component?

LEE: All the middle-ware we use (open source and free) comes with full source code so we can step through the alien code at the lowest level and study the performance metrics. By creating timers in different sections of the code, we can measure how long each section takes to execute and identify areas that are consuming an unusually large portion of the overall run-time. As to what constitutes 'unusually large' is down to experience and the budget we work with which is about 16 milliseconds per game cycle. The biggest hurdle to implementing middle ware is the availability of good documentation, examples and availability of experiential advice from say a forum. If you have those, the implementation is smooth.

Q2: What are the main decision making aspects when deciding between reuse of components or own development? When do you decide to reuse a component instead of developing it yourself for your FPS reloaded product?

LEE: Ten years ago I would have argued that everything be written yourself, keep the I.P and you know exactly where to go if you need to fix something. These days I would say there are some middle ware choices that will accelerate overall development without sacrificing creative or legal freedoms, nor incur a significant financial burden. The decision boils down to whether it's quicker/cheaper to buy-in/license the code rather than put someone in front of a PC and ask them to create one from scratch.

Referring to: ""We will always be looking at clever tricks and tweaks to increase performance further. ""

Q3: How did you manage to get the 30% performance increase for FPS reloaded?  
What concepts of programming/tweaking/tricks are involved? (So we can do more research on these concepts/principles.)

LEE: The 30% was gained by thoroughly breaking down the modules used in the game cycle and deciding how many milliseconds to give to each one to get a better FPS score. Through measuring all these components we discovered the AI system was eating quite a lot of processing, which turned out to be a series of features the final engine did not use, so by removing them from the AI sub-system we gained extra speed. We also introduced code in various parts of the renderer so that we only render what is absolutely needed to be on the screen rather than everything that was in the game. Culling objects from reflective water, from behind buildings, from a distance too far to see the object, by hiding them from the render step we tax the GPU less and gain more frame rate as a result. The process of getting more performance is exactly the same now. We take what we have and we apply measurements to all the known modules and decide which is the next 'most expensive and unusually large' consumption and see if we can trim it down a touch.

### **Interview: Hans Wichman, Senior software developer, Inner Drive studios, 15/08/14**

Q1.What are potential areas of software reuse in your company?

Mainly Game Components and Assets from the Unity 3d asset store, music and texture libraries, also scripts and algorithms.

Q2.What are the positive and negative effects of software reuse on different software development economics in your company ?

By reusing components both internally or externally that are properly documented or have examples we are able to improve quality and potentially lower development costs compared when building the same thing from scratch.

Q3. What are the main decision making aspects when deciding between reuse or building something new?

Can we build it ourselves? Do we have the resources: Time, budget , required knowledge to build the component. Is there something commercially available that matches most of the requirements we need for the game?

Q4. What are the general implementing problems/challenges developers face when implementing a middle ware component or external component

Good developer examples and good documentation are key in integrating a component. For some problems there are just no standard, easy solutions thus there can be need in creating something yourself for a particular problem.

Q5. What concepts of programming/tweaking/tricks are involved in optimizing the performance of your software.

Timing the software, CPU-profiling, there are also automated tools which allows us to step through the program to find any performance bottlenecks.

**Interview: Michael Angelo Groeneveld, Senior software developer El-Nino B.V, 03/07/14**

Q1.What are potential areas of software reuse in your company?

There are many based on the nature of the product / system to be built. We try to reuse as much as possible to cut down time and effort. We have a large base software that we use for most of our systems. This base software is regularly updated to improve the stability and performance of the implemented systems.

Q2.What are the positive and negative effects of software reuse on different software development economics in your company ?

We don't reuse software enough. But the advantages we already notice is that development time has reduced drastically when it comes to creating admin screens. We are trying to make similar steps when it comes to frontend development.

Q3. What are the main decision making aspects when deciding between reuse or building something new?

Usually time and cost. Standard functionality already present in the base system is always used.

Q4. What are the general implementing problems/challenges developers face when implementing a middle ware component or external component

Usually bad documentation, bugs in code made by external companies / programmers. Slow connection times and functional limitations of external software.

Q5. What concepts of programming/tweaking/tricks are involved in optimizing the performance of your software.

Code re-formatting, caching mechanisms, gulp, server clusters, database technology and offloading processes using frontend technologies like angularjs and backbone.

**Interview: Teun Lassche, Senior software developer, T.H. Lassche Webdevelopment, 05/07/14**

Q1.What are potential areas of software reuse in your company?

Mostly in backend components, we use auto generate software to generate the reusable parts of our software.

Q2.What are the positive and negative effects of software reuse on different software development economics in your company ?

Development time reduces, however it may come with a lower amount of innovation. By reusing the software the revenue generated for the customer will increase in lower time.

Q3. What are the main decision making aspects when deciding between reuse or building something new?

Costs.

Q4. What are the general implementing problems/challenges developers face when implementing a middle ware component or external component

Low code quality or ambiguities in the core itself. Lots of components are not actively maintained anymore. Most of the components fit 90% of the functionality but miss the other essential 10% and are therefore only partly usable.

Q5. What concepts of programming/tweaking/tricks are involved in optimizing the performance of your software.

Caching, key value storages. Making heavy tasks a-sync so end user doesn't notice increased loading times. Furthermore, by writing efficient code.

**Interview: Pieter van den Bosch, Jr. software developer, Magdeveloper.com, 20/07/14**

Q1.What are potential areas of software reuse in your company?

We mainly use software development tools that require a yearly license. There are several categories of which we use software. One aspect is the reuse of certain architecture and to reuse some modules throughout software development.

General license reuse/updates

Component reuse e.g. modules in different webshops but same architecture.

Reuse software for promotions

Requiring a yearly license often forces us to think in long term.

Q2.What are the positive and negative effects of software reuse on different software development economics in your company ?

We use several reuse methodologies that help us to be more agile in the building processes leading in better cost control and faster development time. As explained earlier we sometimes are dependent in yearly perpetual licenses that forces us to think long term. When we purchase a license we often ask ourselves is this the best software that we want to use for a year?

Q3. What are the main decision making aspects when deciding between reuse or building something new?

The main aspects are the timeframe, budget and speed (can we build it ourselves) of the project.

Q4. What are the general implementing problems/challenges developers face when implementing a middle ware component or external component

Sometimes the component lacks technical documentation this forces us to investigate more if the component does not work out of the box correctly which can happen with software. A challenge also is with third party software that your employee has to remember how and what specific part the external component may be good for.

Q5. What concepts of programming/tweaking/tricks are involved in optimizing the performance of your software.

We mainly do stress tests and trial and error tests. One big advantage is that we closely work with the customer who also tests our software. Our general idea is that software development is never finished so most of the time we provide continuous support for the customer allowing to work according mile stones. During all our development we closely work with programmers who know the iterative concept of building on top of components they or other have created. Due to objective programming and community support we know the market changes and can quickly adapt to changes that may be required in our software.

## Appendix B: Survey questionnaire

# STUDY ON GAME DEVELOPMENT

0%  100%

### PART I: PRODUCT DESIGN

The purpose of this section is to gain insight in the development strategies you followed in the selected game development project. Strategies include for example modularity, reuse and tweaking efforts.

Synonyms that may be used in your company for 'modules' include for example assets, components, classes, or other chunks of software.

**1. Please indicate how strongly you disagree or agree with each of the following statements.** Your answer can be given on a scale ranging from 1 "strongly disagree" thru 7 "strongly agree".

	1-Strongly disagree	2	3	4	5	6	7-Strongly agree
The game was decomposed into separate modules	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
We stressed to reuse software when possible	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
We relied heavily on a common architecture that was used across previous games	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Our software development process was built around a core set of reusable software components as the foundation for our game	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Our development team precisely followed a software reuse process, which is defined and integrated with the organization's software development process	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Our development team used databases listing standard components and their interface specifications	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
We relied on stable, well-defined interfaces among software modules	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

**2. Please indicate to what degree you disagree or agree with the following statements.**

For the selected game we developed there was a complete set of design rules available that fully described the following categories of design information:

	1-Strongly disagree	2	3	4	5	6	7-Strongly agree
<b>The architecture</b> (i.e., what subsystems will be part of the architecture system, and what the roles of the subsystems will be in the architecture system)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<b>The interfaces among the different software subsystems</b> (i.e., detailed descriptions of how the different subsystems will interact, including how they will fit together, connect, communicate and so forth).	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<b>Integrations protocols</b> (i.e. procedures that will allow designers to assemble the architecture system)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<b>Testing standards</b> (i.e., standards that will allow designers to determine how well the architecture system works, whether a particular (sub-)system conforms to the design rules, and how one version of a subsystem performs relative to another)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

## PART II: SOFTWARE DEVELOPMENT AND OPTIMIZATIONS

Sometimes software subsystems require extensive integration and optimization efforts for them to work together. The development team will be forced to modify standard subsystems and interfaces towards the specific requirements of a particular project.

Synonyms that may be used in your company for 'modules' include for example assets, components, classes, or other chunks of software.

### 1. For the selected game development project, please indicate to what extent:

	1 - No extent	2	3	4 - Moderate extent	5	6	7 - Full extent
... modules required extensive interaction for them to work together	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... the development team optimized modules to work with each other	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... the development team modified modules particularly for this game	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... reused modules needed to be modified towards the functional requirements of this game	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... reused modules required extensive integration efforts	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... interfaces among the modules made the modules less recombining in other games	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... design changes of modules resulted in changes of other modules	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... the development team put effort in specifying interfaces between modules	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... modifications made to modules in this game have no value for future games	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... design specifications of individual modules affected the selection criteria of other modules	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

### 2. For the selected game development project, please indicate to what extent:

	1 - No extent	2	3	4 - Moderate extent	5	6	7 - Full extent
... the development team put effort in finding small bugs/errors throughout the system	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... the development team used systematic procedures for fine-tuning models	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... the development team used automatic tweaking utilities	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... the development team made modifications to reused components	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... the development team spent effort in selecting the right modules for this game	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
... the development team tried to gain insight into performance bottlenecks and make adjustments	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

### PART III: SOFTWARE REUSE INFORMATION

This part of the survey is used to gain insight in the degree of reuse of software in the development of this game. Software reuse is the process whereby software systems are created from existing software rather than building them from scratch. Furthermore we ask questions about the source of these reused pieces of software (e.g. internal libraries or Commercially of the Shelf Components) and we zoom in on the reuse of specific system components and assets.

1. Please indicate the overall degree of software reuse throughout the development of the selected game:

	1 - No reuse at all	2	3	4 - Half reuse / half new content	5	6	7 - Full reuse
Your answer:	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

2. Please indicate the primary source of the total reuse portion:

	1 - Fully internal	2	3	4 - Half internal / half external	5	6	7 - Fully external
Your answer:	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

? - Internal sources such as company's internal libraries  
 - External sources such as Commercial of the Shelf (COTS) or Open Source Software (OSS)

3. Please indicate the degree of software reuse for each of the following system components:

	1 - No reuse at all	2	3	4 - Half reuse & half new content	5	6	7 - Full reuse
Artificial intelligence components	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Rendering components	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Physics components	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Animation components	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Visual effects components	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Audio components	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Game specific subsystems	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Overall reuse estimation system components	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

4. Please indicate the degree of software reuse for each of the following game assets:

	1 - No reuse at all	2	3	4 - Half reuse & half new content	5	6	7 - Full reuse
Game objects	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Game levels	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
(3D) Models	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Audio files	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Shaders	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Scripts	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Textures	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Materials	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Animations	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Story elements	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Overall reuse estimation game assets	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



#### PART IV: PROJECT PERFORMANCE

The purpose of this section is to get deeper insight in the performance of this game in terms of costs, development speed and quality.

1. Please indicate how strongly you disagree or agree with each of the following statements:

	1-Strongly disagree	2	3	4	5	6	7-Strongly agree
The game was a financial success	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Relative to our other projects, this game exceeded profitability	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Relative to our competing games, this game exceeded profitability	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The game exceeded profitability objectives	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

2. Please indicate how strongly you disagree or agree with each of the following statements:

	1-Strongly disagree	2	3	4	5	6	7-Strongly agree
This project was developed and launched faster than our major competitors	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
This project was completed in less time than what was considered normal and customary for our industry	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
This game was launched on or ahead of the original schedule developed at initial project go-ahead	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

3. Please indicate how strongly you disagree or agree with each of the following statements:

	1-Strongly disagree	2	3	4	5	6	7-Strongly agree
This project was launched within or under the original budget	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
This project came in at or below cost estimate for development	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
This project was less costly than considered normal and customary in our industry	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

4. Please indicate how strongly you disagree or agree with each of the following statements:  
(at the time of release)

	1-Strongly disagree	2	3	4	5	6	7-Strongly agree
The product performance met the requirements of customers	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The product met quality standards expected by the customers	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
On average, the number of defects (i.e. errors and bugs) identified after launch was very low	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
On average, customers perceive our game to perform better than other comparable games on the marketplace	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

5. Please indicate how strongly you disagree or agree with each of the following statements:

	1-Strongly disagree	2	3	4	5	6	7-Strongly agree
Investments made in developing the modules and architecture of this game are of great value for successive games	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
This game's architecture and modules function as a platform for derivative games	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
A large part of the costs made for developing this game were to be returned in future games	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

## PART V: GENERAL PROJECT INFORMATION

*The purpose of this final section is to insight in the primary data of this development project.*

1. What was the approximate number of full time employees included in the development team?

*Only numbers may be entered in this field.*

2. Please indicate the approximate size of this game according to the total lines of programming code:

3. How would you indicate the overall level of experience of this development team?

	1 - Beginner	2	3	4 - Intermediate	5	6	7 - Expert
Overall experience level development team	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

4. Please indicate how strongly you disagree or agree with the following statement:

	1-Strongly disagree	2	3	4	5	6	7-Strongly agree
Relative to other game development projects the marketing resources devoted to this project were high	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

5. What was approximately the total estimated dollar cost of this game development project:

6. Please name the game engine used in this project (optional question):

We would like to reach as many respondents as possible. Perhaps you are willing to provide us with the contact details of other game development project leaders (located in your firm or other firms).

	Name:	E-mail address:	Game developed:
Person 1:	<input type="text"/>	<input type="text"/>	<input type="text"/>
Person 2:	<input type="text"/>	<input type="text"/>	<input type="text"/>
Person 3:	<input type="text"/>	<input type="text"/>	<input type="text"/>

## Appendix C: Reliability analysis

**Reliability Statistics**

Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	N of Items
,898	,905	8

**Summary Item Statistics**

	Mean	Minimum	Maximum	Range	Maximum / Minimum	Variance	N of Items
Item Means	4,445	2,813	5,438	2,625	1,933	,967	8

**Inter-Item Correlation Matrix**

	Artificial intelligence components	Rendering components	Physics components	Animation components	Visual effects components	Audio components	Game specific subsystems	Overall reuse estimation system components
Artificial intelligence components	1,000	,351	,432	,230	,064	,178	,375	,583
Rendering components	,351	1,000	,558	,629	,633	,699	,536	,754
Physics components	,432	,558	1,000	,523	,598	,667	,017	,644
Animation components	,230	,629	,523	1,000	,722	,687	,407	,826
Visual effects components	,064	,633	,598	,722	1,000	,918	,437	,768
Audio components	,178	,699	,667	,687	,918	1,000	,544	,789
Game specific subsystems	,375	,536	,017	,407	,437	,544	1,000	,645
Overall reuse estimation system components	,583	,754	,644	,826	,768	,789	,645	1,000

**Item-Total Statistics**

	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Squared Multiple Correlation	Cronbach's Alpha if Item Deleted
Artificial intelligence components	32,75	122,200	,372	,840	,912
Rendering components	30,13	107,850	,770	,632	,877
Physics components	30,56	105,862	,628	,881	,892
Animation components	30,81	103,096	,739	,841	,880
Visual effects components	30,69	104,762	,775	,923	,876
Audio components	30,25	102,867	,848	,935	,869
Game specific subsystems	32,44	118,796	,505	,874	,900
Overall reuse estimation system components	31,31	111,162	,954	,968	,870

Reliability Statistics		
Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	N of Items
,952	,952	11

**Summary Item Statistics**

	Mean	Minimum	Maximum	Range	Maximum / Minimum	Variance	N of Items
Item Means	2,938	2,125	4,125	2,000	1,941	,291	11

**Inter-Item Correlation Matrix**

	Game objects	Game levels	(3D) Models	Audio files	Shaders	Scripts	Textures	Materials	Animations	Story elements	Overall reuse estimation game assets
Game objects	1,000	,627	,806	,667	,369	,724	,791	,673	,853	,344	,847
Game levels	,627	1,000	,830	,785	,582	,727	,735	,700	,638	,607	,819
(3D) Models	,806	,830	1,000	,645	,503	,867	,909	,801	,804	,342	,877
Audio files	,667	,785	,645	1,000	,406	,541	,667	,557	,643	,615	,763
Shaders	,369	,582	,503	,406	1,000	,233	,558	,651	,504	,150	,608
Scripts	,724	,727	,867	,541	,233	1,000	,669	,576	,599	,464	,663
Textures	,791	,735	,909	,667	,558	,669	1,000	,919	,933	,282	,937
Materials	,673	,700	,801	,557	,651	,576	,919	1,000	,840	,198	,886
Animations	,853	,638	,804	,643	,504	,599	,933	,840	1,000	,312	,891
Story elements	,344	,607	,342	,615	,150	,464	,282	,198	,312	1,000	,383
Overall reuse estimation game assets	,847	,819	,877	,763	,608	,663	,937	,886	,891	,383	1,000

**Item-Total Statistics**

	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Squared Multiple Correlation	Cronbach's Alpha if Item Deleted
Game objects	29,25	270,867	,822	,934	,946
Game levels	30,19	278,696	,857	,947	,946
(3D) Models	29,50	260,000	,912	,987	,943
Audio files	29,56	278,663	,753	,821	,949
Shaders	28,19	291,096	,542	,641	,956
Scripts	29,31	279,429	,733	,925	,949
Textures	29,19	259,763	,921	,991	,942
Materials	29,19	265,096	,838	,935	,946
Animations	29,19	262,296	,868	,957	,944
Story elements	30,19	301,229	,421	,702	,959
Overall reuse estimation game assets	29,38	271,717	,954	,965	,942

### Reliability Statistics

Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	N of Items
,971	,972	8

### Summary Item Statistics

	Mean	Minimum	Maximum	Range	Maximum / Minimum	Variance	N of Items
Item Means	3,636	3,458	3,758	,300	1,087	,012	8

### Item-Total Statistics

	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Squared Multiple Correlation	Cronbach's Alpha if Item Deleted
Acting	25,333	15,933	,717	.	,976
AI	25,550	16,179	,806	.	,972
Gameplay	25,417	15,242	,884	.	,967
Graphics	25,333	14,641	,921	.	,965
Personal Slant	25,525	14,144	,954	.	,963
Sound & music	25,392	15,754	,879	.	,968
Story & Presentation	25,633	13,981	,963	.	,963
Overall score	25,458	15,024	,999	.	,961

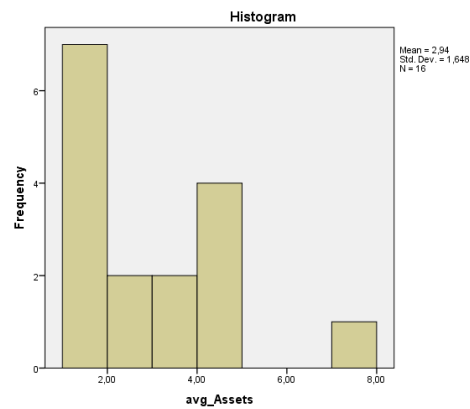
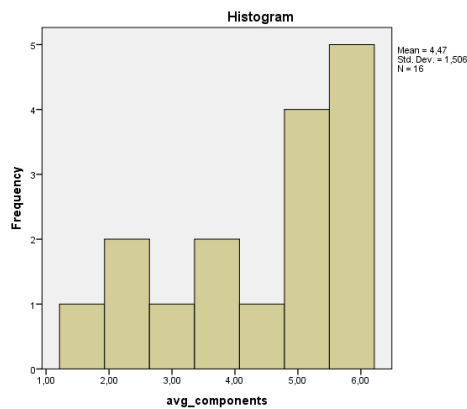
## Appendix D: Normality tests

### Independent variables

Tests of Normality

	Kolmogorov-Smirnov <sup>a</sup>			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AVG_Components	,234	16	,019	,878	16	<b>,04</b>
AVG_Assets	,190	16	,124	,891	16	,06

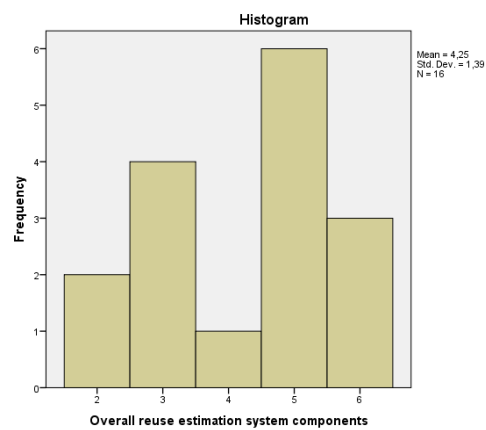
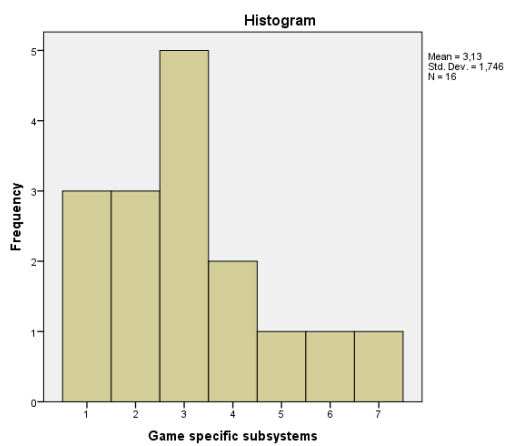
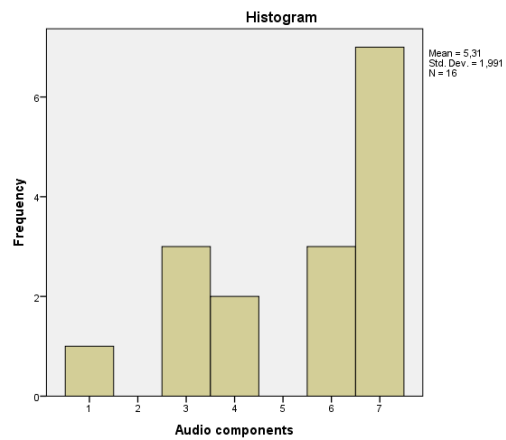
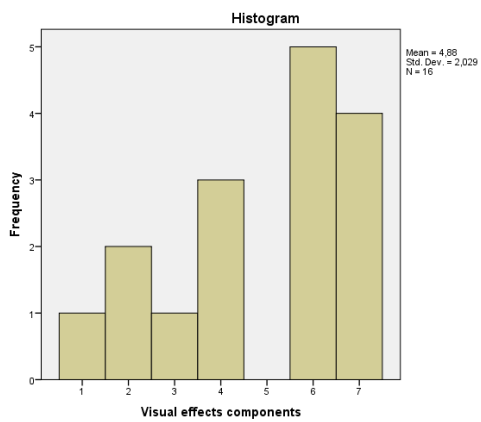
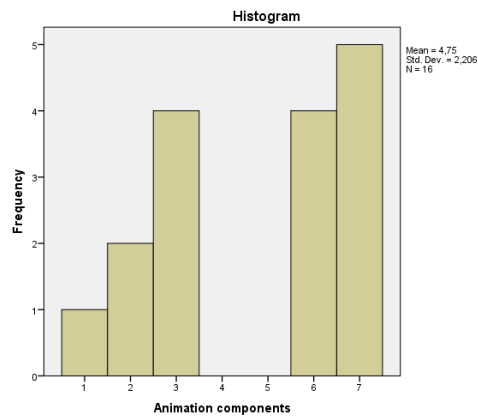
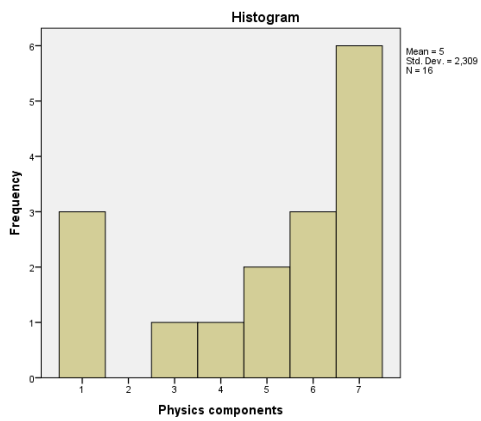
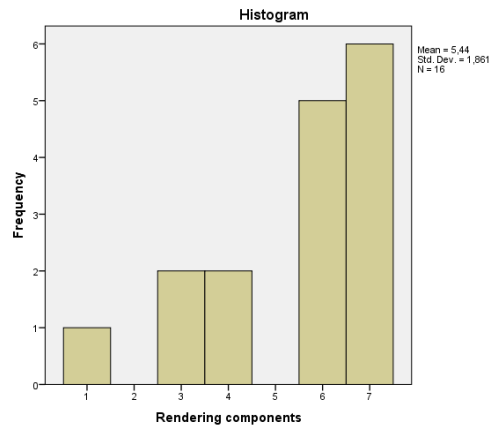
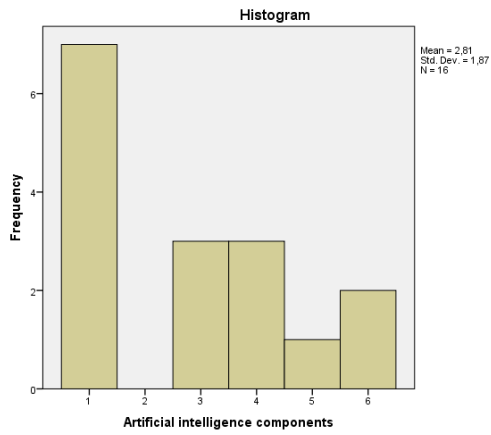
a. Lilliefors Significance Correction



Tests of Normality

	Kolmogorov-Smirnov <sup>a</sup>			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Artificial intelligence components	,27	16,00	,00	,84	16,00	<b>,01</b>
Rendering components	,31	16,00	,00	,81	16,00	<b>,00</b>
Physics components	,23	16,00	,02	,80	16,00	<b>,00</b>
Animation components	,28	16,00	,00	,83	16,00	<b>,01</b>
Visual effects components	,27	16,00	,00	,87	16,00	<b>,03</b>
Audio components	,26	16,00	,00	,81	16,00	<b>,00</b>
Game specific subsystems	,22	16,00	,04	,91	16,00	,13
Overall reuse estimation system components	,27	16,00	,00	,87	16,00	<b>,03</b>

a. Lilliefors Significance Correction



Tests of Normality						
	Kolmogorov-Smirnov <sup>a</sup>			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Game objects	,218	16	,041	,860	16	<b>,019</b>
Game levels	,279	16	,002	,708	16	<b>,000</b>
(3D) Models	,230	16	,024	,787	16	<b>,002</b>
Audio files	,215	16	,047	,848	16	<b>,013</b>
Shaders	,210	16	,058	,897	16	,073
Scripts	,260	16	,005	,878	16	<b>,036</b>
Textures	,269	16	,003	,828	16	<b>,007</b>
Materials	,206	16	,068	,839	16	<b>,010</b>
Animations	,265	16	,004	,823	16	<b>,006</b>
Story elements	,298	16	,000	,704	16	<b>,000</b>
Overall reuse estimation game assets	,202	16	,080	,903	16	,088

a. Lilliefors Significance Correction

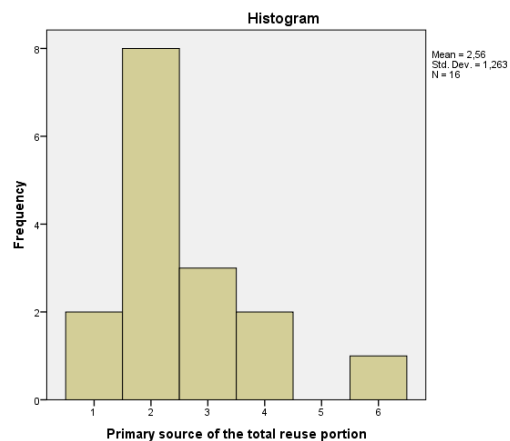
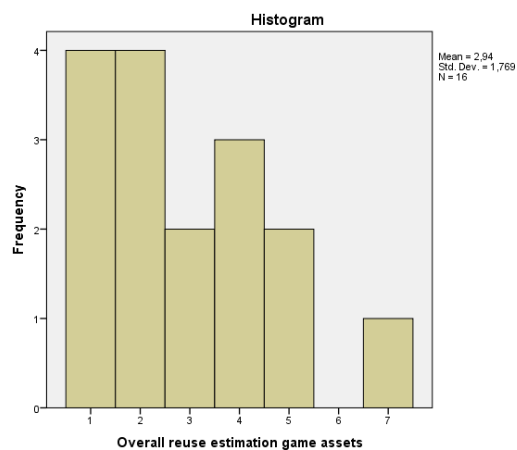
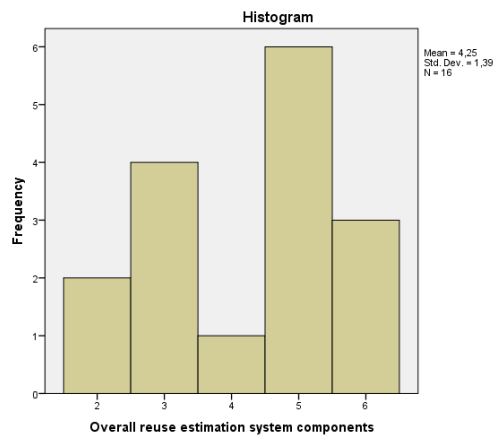
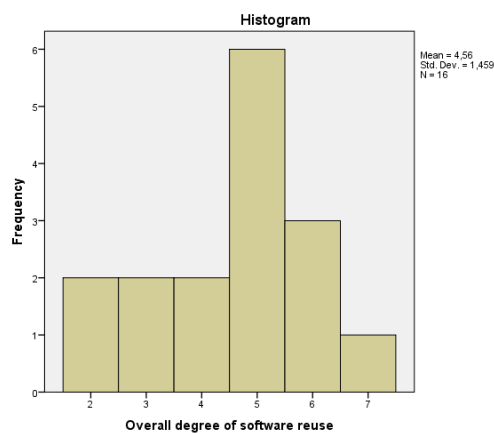


## Reuse levels

### Tests of Normality

	Kolmogorov-Smirnov <sup>a</sup>			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Overall degree of software reuse	,24	16,00	,01	,92	16,00	,18
Overall reuse estimation system components	,27	16,00	,00	,87	16,00	<b>,03</b>
Overall reuse estimation game assets	,20	16,00	,08	,90	16,00	,09
Primary source of the total reuse portion	,30	16,00	,00	,83	16,00	<b>,01</b>

a. Lilliefors Significance Correction



## Dependent Variables:

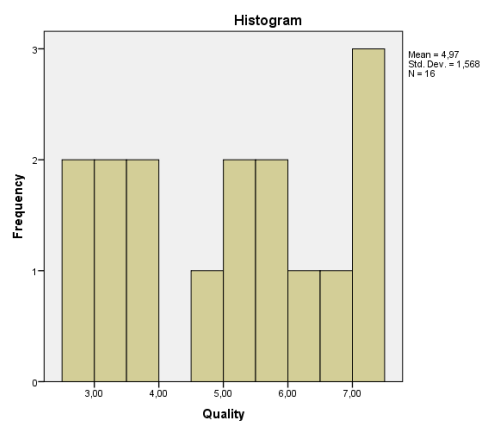
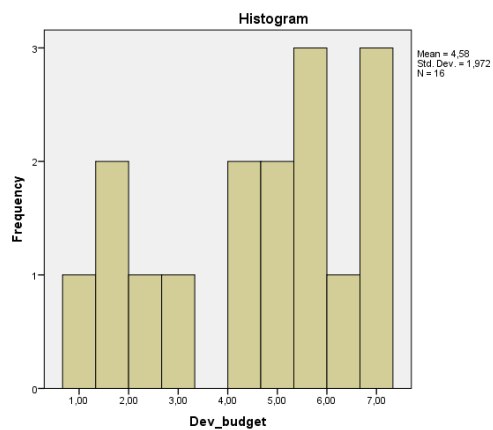
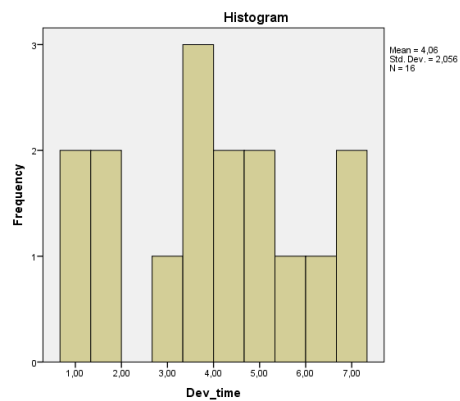
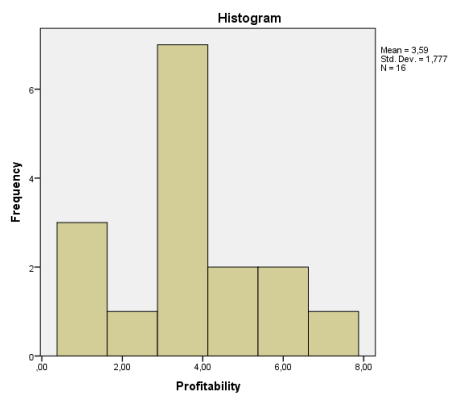
### Project Performance

Tests of Normality

	Kolmogorov-Smirnov <sup>a</sup>			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Profitability	,173	16	,200 <sup>*</sup>	,944	16	,41
Devtime	,128	16	,200 <sup>*</sup>	,931	16	,25
Dev_budget	,146	16	,200 <sup>*</sup>	,922	16	,18
Quality	,157	16	,200 <sup>*</sup>	,903	16	,09

\*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction



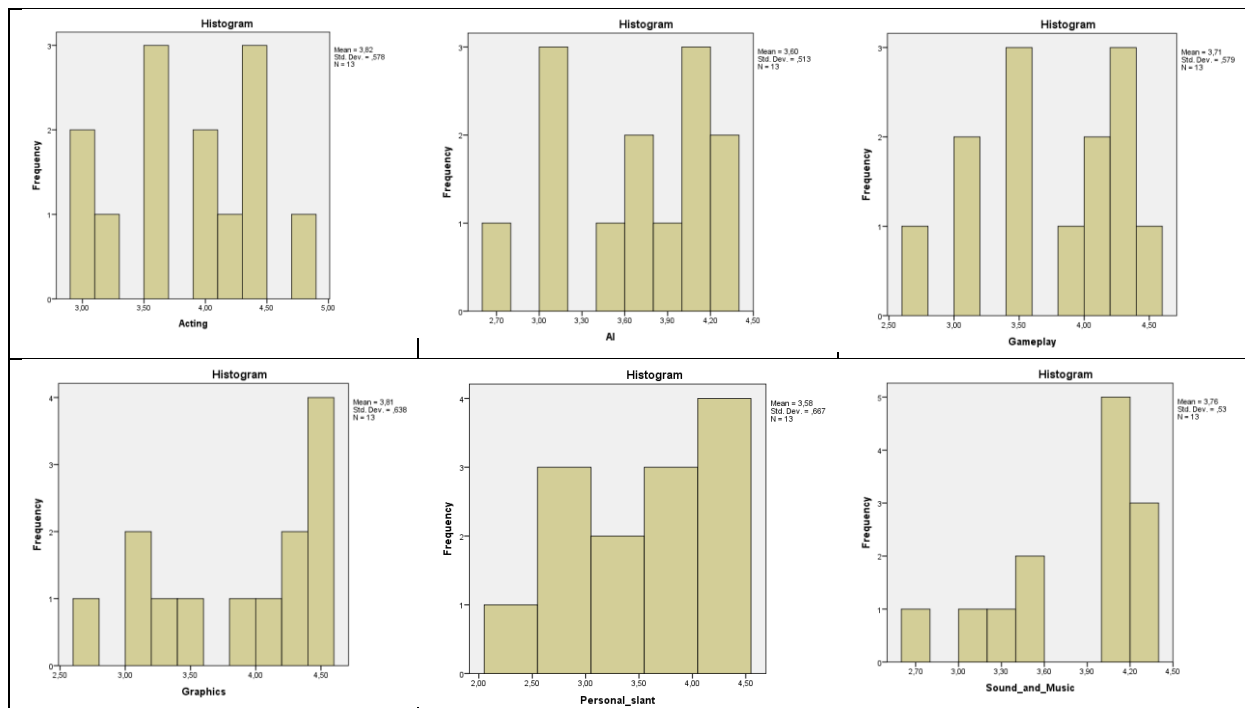
## Review Scores

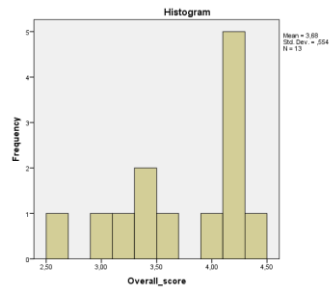
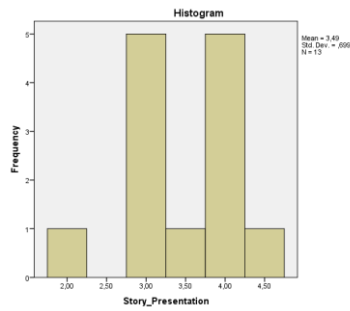
### Tests of Normality

	Kolmogorov-Smirnov <sup>a</sup>			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Acting	,19	12,00	,20 <sup>*</sup>	,92	12,00	,27
AI	,21	12,00	,16	,90	12,00	,15
Gameplay	,15	12,00	,20 <sup>*</sup>	,94	12,00	,48
Graphics	,17	12,00	,20 <sup>*</sup>	,90	12,00	,17
Personal Slant	,23	12,00	,07	,92	12,00	,28
Sound & music	,22	12,00	,12	,91	12,00	,25
Story & Presentation	,18	12,00	,20 <sup>*</sup>	,92	12,00	,28
Overall score	,22	12,00	,13	,91	12,00	,23

\*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction





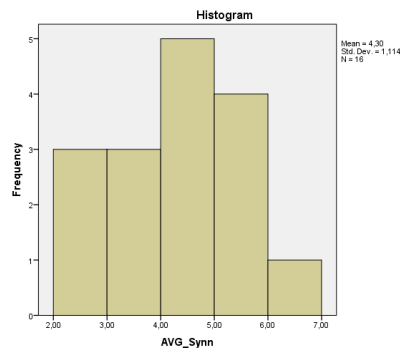
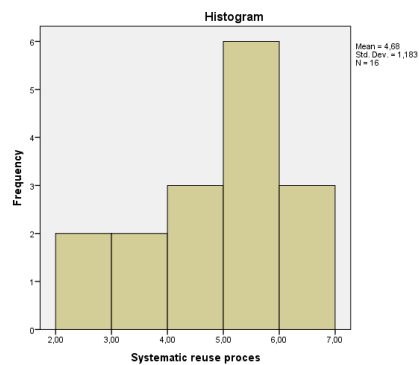
## Control Variables

Tests of Normality

	Kolmogorov-Smirnov <sup>a</sup>			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Systematic reuse process	,170	16	,200 <sup>*</sup>	,930	16	,248
Synergistic Specificity	,106	16	,200 <sup>*</sup>	,978	16	,941

\*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

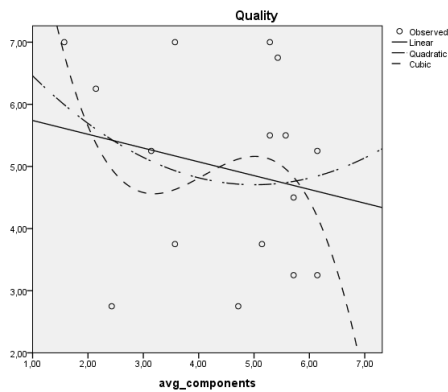


## Appendix E: Curvilinear relationships testing

Model Summary and Parameter Estimates

Dependent Variable: Quality									
Equation	Model Summary					Parameter Estimates			
	R Square	F	df1	df2	Sig.	Constant	b1	b2	b3
Linear	,045	,666	1	14	,428	5,961	-,222		
Quadratic	,059	,408	2	13	,674	7,441	-1,092	,109	
Cubic	,143	,668	3	12	,588	16,422	-9,447	2,425	-,197

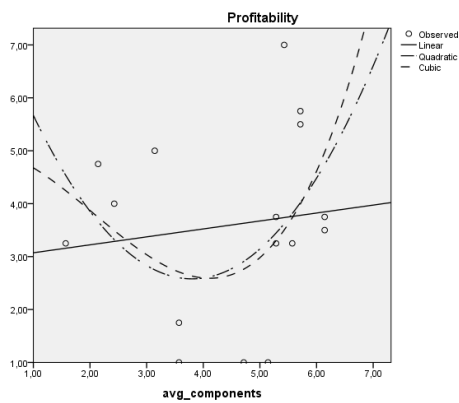
The independent variable is avg\_components.



Model Summary and Parameter Estimates

Dependent Variable: Profitability									
Equation	Model Summary					Parameter Estimates			
	R Square	F	df1	df2	Sig.	Constant	b1	b2	b3
Linear	,016	,231	1	14	,638	2,922	,150		
Quadratic	,154	1,183	2	13	,337	8,266	-2,992	,394	
Cubic	,162	,776	3	12	,530	5,026	,022	-,442	,071

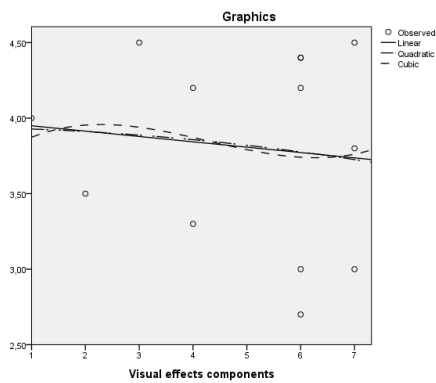
The independent variable is avg\_components.



Model Summary and Parameter Estimates

Dependent Variable: Graphics									
Equation	Model Summary					Parameter Estimates			
	R Square	F	df1	df2	Sig.	Constant	b1	b2	b3
Linear	,012	,137	1	11	,718	3,985	-,035		
Quadratic	,013	,064	2	10	,939	3,938	-,006	-,003	
Cubic	,016	,050	3	9	,984	3,655	,300	-,089	,007

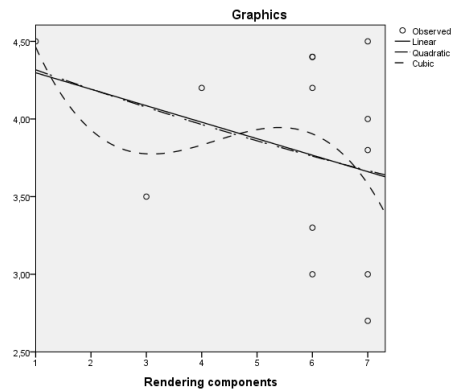
The independent variable is Visual effects components.



Model Summary and Parameter Estimates

Dependent Variable: Graphics									
Equation	Model Summary					Parameter Estimates			
	R Square	F	df1	df2	Sig.	Constant	b1	b2	b3
Linear	,095	1,151	1	11	,306	4,404	-,106		
Quadratic	,095	,525	2	10	,607	4,447	-,133	,003	
Cubic	,150	,529	3	9	,673	5,545	-1,412	,358	-,028

The independent variable is Rendering components.

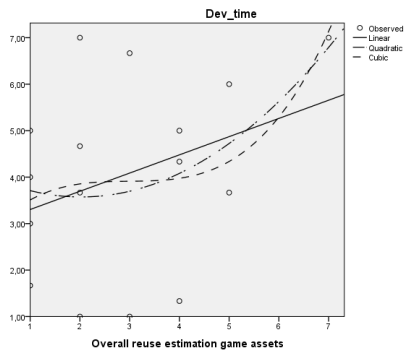


Model Summary and Parameter Estimates

Dependent Variable: Dev\_time

Equation	Model Summary					Parameter Estimates			
	R Square	F	df1	df2	Sig.	Constant	b1	b2	b3
Linear	,114	1,797	1	14	,201	2,911	,392		
Quadratic	,159	1,225	2	13	,326	4,106	-.527	,130	
Cubic	,175	,846	3	12	,495	2,581	1,316	-.438	,049

The independent variable is Overall reuse estimation game assets.

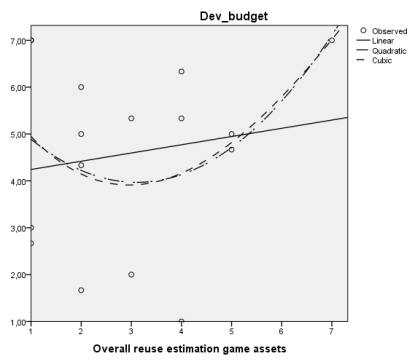


Model Summary and Parameter Estimates

Dependent Variable: Dev\_budget

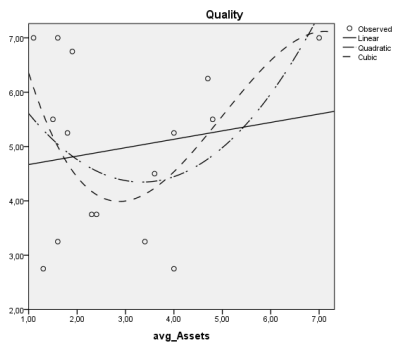
Equation	Model Summary					Parameter Estimates			
	R Square	F	df1	df2	Sig.	Constant	b1	b2	b3
Linear	,025	,357	1	14	,560	4,067	,176		
Quadratic	,149	1,135	2	13	,351	5,973	-1,289	,207	
Cubic	,150	,706	3	12	,567	6,395	-1,799	,365	-.014

The independent variable is Overall reuse estimation game assets.



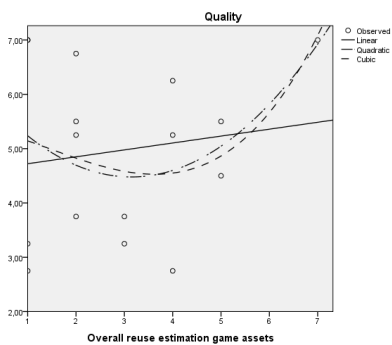
Model Summary and Parameter Estimates									
Dependent Variable: Quality									
Model Summary					Parameter Estimates				
Equation	R Square	F	df1	df2	Sig.	Constant	b1	b2	b3
Linear	,027	,382	1	14	,546	4,513	,155		
Quadratic	,221	1,847	2	13	,197	6,912	-1,536	,230	
Cubic	,264	1,431	3	12	,282	10,228	-5,021	1,227	-.082

The independent variable is avg\_Assets.



Model Summary and Parameter Estimates									
Dependent Variable: Quality									
Model Summary					Parameter Estimates				
Equation	R Square	F	df1	df2	Sig.	Constant	b1	b2	b3
Linear	,021	,294	1	14	,596	4,595	,127		
Quadratic	,144	1,093	2	13	,364	6,107	-1,035	,165	
Cubic	,150	,705	3	12	,567	5,397	-.178	-.100	,023

The independent variable is Overall reuse estimation game assets.



## Appendix F: ANOVA Results

### Overall reuse level on Project Performance

		Statistics			
		Overall degree of software reuse	Overall reuse estimation system components	Overall reuse estimation game assets	Primary source of the total reuse portion
N	Valid	16	16	16	16
	Missing	0	0	0	0
Mean		4,56	4,25	2,94	2,56
Median		<b>5,00</b>	<b>5,00</b>	<b>2,50</b>	<b>2,00</b>
Mode		5	5	1 <sup>a</sup>	2
Std. Deviation		1,459	1,390	1,769	1,263
Variance		2,129	1,933	3,129	1,596

a. Multiple modes exist. The smallest value is shown

		Statistics	
		avg_components	avg_Assets
N	Valid	16	16
	Missing	0	0
Mean		4,4732	2,9375
Median		<b>5,2143</b>	<b>2,3500</b>
Mode		3,57 <sup>a</sup>	1,60 <sup>a</sup>
Std. Deviation		1,50619	1,64838
Variance		2,269	2,717

a. Multiple modes exist. The smallest value is shown



### Between-Subjects Factors

	Value Label	N
Reuse Level	1,00	Low Reuse
	2,00	High Reuse
		6
		10

### Descriptive Statistics

	Reuse Level	Mean	Std. Deviation	N
Profitability	Low Reuse	3,2917	1,62340	6
	High Reuse	3,7750	1,92372	10
	Total	3,5938	1,77688	16
Dev_time	Low Reuse	3,1667	2,38281	6
	High Reuse	4,6000	1,74129	10
	Total	4,0625	2,05559	16
Dev_budget	Low Reuse	3,5556	2,40986	6
	High Reuse	5,2000	1,45890	10
	Total	4,5833	1,97203	16
Quality	Low Reuse	5,3333	1,76541	6
	High Reuse	4,7500	1,49071	10
	Total	4,9688	1,56758	16

### Box's Test of Equality of Covariance Matrices<sup>a</sup>

Box's M	24,854
F	1,582
df1	10
df2	506,163
Sig.	,108

Tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups.

a. Design: Intercept + Reuse\_level

**Multivariate Tests<sup>a</sup>**

Effect		Value	F	Hypothesis df	Error df	Sig.	Partial Eta Squared
Intercept	Pillai's Trace	,957	61,805 <sup>b</sup>	4,000	11,000	,000	,957
	Wilks' Lambda	,043	61,805 <sup>b</sup>	4,000	11,000	,000	,957
	Hotelling's Trace	22,475	61,805 <sup>b</sup>	4,000	11,000	,000	,957
	Roy's Largest Root	22,475	61,805 <sup>b</sup>	4,000	11,000	,000	,957
Reuse_level	Pillai's Trace	,297	1,162 <sup>b</sup>	4,000	11,000	,379	,297
	Wilks' Lambda	,703	1,162 <sup>b</sup>	4,000	11,000	,379	,297
	Hotelling's Trace	,423	1,162 <sup>b</sup>	4,000	11,000	,379	,297
	Roy's Largest Root	,423	1,162 <sup>b</sup>	4,000	11,000	,379	,297

a. Design: Intercept + Reuse\_level

b. Exact statistic

**Levene's Test of Equality of Error Variances<sup>a</sup>**

	F	df1	df2	Sig.
Profitability	,026	1	14	,873
Dev_time	1,115	1	14	,309
Dev_budget	3,230	1	14	,094
Quality	,184	1	14	,674

Tests the null hypothesis that the error variance of the dependent variable is equal across groups.

a. Design: Intercept + Reuse\_level

**Reuse Level**

Dependent Variable	Reuse Level	Mean	Std. Error	95% Confidence Interval	
				Lower Bound	Upper Bound
Profitability	Low Reuse	3,292	,744	1,696	4,887
	High Reuse	3,775	,576	2,539	5,011
Dev_time	Low Reuse	3,167	,814	1,421	4,913
	High Reuse	4,600	,631	3,247	5,953
Dev_budget	Low Reuse	3,556	,757	1,931	5,180
	High Reuse	5,200	,587	3,942	6,458
Quality	Low Reuse	5,333	,651	3,937	6,729
	High Reuse	4,750	,504	3,669	5,831

## Specific Components on Project Performance

### Between-Subjects Factors

		Value Label	N
Components grouped	1,00	Low reuse	8
	2,00	High reuse	8

### Descriptive Statistics

	Components grouped	Mean	Std. Deviation	N
Profitability	Low reuse	2,7188	1,73430	8
	High reuse	4,4688	1,41697	8
	Total	3,5938	1,77688	16
Dev_time	Low reuse	3,1250	2,28131	8
	High reuse	5,0000	1,35693	8
	Total	4,0625	2,05559	16
Dev_budget	Low reuse	3,7083	2,35997	8
	High reuse	5,4583	1,00692	8
	Total	4,5833	1,97203	16
Quality	Low reuse	4,8125	1,79657	8
	High reuse	5,1250	1,40789	8
	Total	4,9688	1,56758	16

### Box's Test of Equality of Covariance

#### Matrices<sup>a</sup>

Box's M	9,766
F	,666
df1	10
df2	937,052
Sig.	,756

Tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups.

a. Design: Intercept + Components\_grouped

**Multivariate Tests<sup>a</sup>**

Effect		Value	F	Hypothesis df	Error df	Sig.	Partial Eta Squared
Intercept	Pillai's Trace	,980	131,905 <sup>b</sup>	4,000	11,000	,000	,980
	Wilks' Lambda	,020	131,905 <sup>b</sup>	4,000	11,000	,000	,980
	Hotelling's Trace	47,966	131,905 <sup>b</sup>	4,000	11,000	,000	,980
	Roy's Largest Root	47,966	131,905 <sup>b</sup>	4,000	11,000	,000	,980
Components_grouped	Pillai's Trace	,644	4,985 <sup>b</sup>	4,000	11,000	,015	,644
	Wilks' Lambda	,356	4,985 <sup>b</sup>	4,000	11,000	<b>,015</b>	,644
	Hotelling's Trace	1,813	4,985 <sup>b</sup>	4,000	11,000	,015	,644
	Roy's Largest Root	1,813	4,985 <sup>b</sup>	4,000	11,000	,015	,644

a. Design: Intercept + Components\_grouped

b. Exact statistic

**Levene's Test of Equality of Error Variances<sup>a</sup>**

	F	df1	df2	Sig.
Profitability	1,243	1	14	,284
Dev_time	3,272	1	14	,092
Dev_costs	12,030	1	14	,004
Quality	1,670	1	14	,217

Tests the null hypothesis that the error variance of the dependent variable is equal across groups.

a. Design: Intercept + Components\_grouped

**Tests of Between-Subjects Effects**

Source	Dependent Variable	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	Profitability	12,250 <sup>a</sup>	1	12,250	4,885	,044	,259
	Dev_time	14,062 <sup>b</sup>	1	14,062	3,992	,066	,222
	Dev_budget	12,250 <sup>c</sup>	1	12,250	3,722	,074	,210
	Quality	,391 <sup>d</sup>	1	,391	,150	,704	,011
Intercept	Profitability	206,641	1	206,641	82,399	,000	,855
	Dev_time	264,063	1	264,063	74,958	,000	,843
	Dev_budget	336,111	1	336,111	102,110	,000	,879
	Quality	395,016	1	395,016	151,643	,000	,915
Components_grouped	Profitability	12,250	1	12,250	4,885	,044	,259
	Dev_time	14,063	1	14,063	3,992	,066	,222
	Dev_budget	12,250	1	12,250	3,722	,074	,210
	Quality	,391	1	,391	,150	,704	,011
Error	Profitability	35,109	14	2,508			
	Dev_time	49,319	14	3,523			
	Dev_budget	46,083	14	3,292			
	Quality	36,469	14	2,605			
Total	Profitability	254,000	16				
	Dev_time	327,444	16				
	Dev_budget	394,444	16				
	Quality	431,875	16				
Corrected Total	Profitability	47,359	15				
	Dev_time	63,382	15				
	Dev_budget	58,333	15				
	Quality	36,859	15				

**Components grouped**

Dependent Variable	Components grouped	Mean	Std. Error	95% Confidence Interval	
				Lower Bound	Upper Bound
Profitability	Low reuse	2,719	,560	1,518	3,920
	High reuse	4,469	,560	3,268	5,670
Dev_time	Low reuse	3,125	,664	1,702	4,548
	High reuse	5,000	,664	3,577	6,423
Dev_budget	Low reuse	3,708	,641	2,333	5,084
	High reuse	5,458	,641	4,083	6,834
Quality	Low reuse	4,813	,571	3,589	6,036
	High reuse	5,125	,571	3,901	6,349

I perform a curvilinear regression analysis in SPSS. Specifically, I test a quadratic effect (one bend in the regression line) using a hierarchical multiple regression approach. I point out the key to the analysis, which is the F change value associated with the squared independent variable. I discuss the beta weights and how they are not particularly interpretable. I also discuss multicollinearity and why it is not a problem in the nonlinear regression case. I also show how to do the nonlinear analysis using a second approach in SPSS which gives more useful scatter plots in the nonlinear regression case

## Specific Assets on Project Performance

**Between-Subjects Factors**

		Value Label	N
avg_ass_gr	1,00	low	8
	2,00	high	8

**Descriptive Statistics**

	avg_ass_gr	Mean	Std. Deviation	N
Profitability	low	3,6250	1,85646	8
	high	3,5625	1,82125	8
	Total	3,5938	1,77688	16
Dev_time	low	3,7500	1,90863	8
	high	4,3750	2,27783	8
	Total	4,0625	2,05559	16
Dev_budget	low	4,5833	2,02171	8
	high	4,5833	2,06059	8
	Total	4,5833	1,97203	16
Quality	low	5,1563	1,72657	8
	high	4,7813	1,48467	8
	Total	4,9688	1,56758	16

**Box's Test of Equality  
of Covariance  
Matrices<sup>a</sup>**

Box's M	8,792
F	,600
df1	10
df2	937,052
Sig.	,815

Tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups.

a. Design: Intercept + Assets\_grouped

**Multivariate Tests<sup>a</sup>**

Effect		Value	F	Hypothesis df	Error df	Sig.	Partial Eta Squared
Intercept	Pillai's Trace	,959	64,332 <sup>b</sup>	4,000	11,000	,000	,959
	Wilks' Lambda	,041	64,332 <sup>b</sup>	4,000	11,000	,000	,959
	Hotelling's Trace	23,394	64,332 <sup>b</sup>	4,000	11,000	,000	,959
	Roy's Largest Root	23,394	64,332 <sup>b</sup>	4,000	11,000	,000	,959
avg_ass_gr	Pillai's Trace	,105	,321 <sup>b</sup>	4,000	11,000	,858	,105
	Wilks' Lambda	,895	,321 <sup>b</sup>	4,000	11,000	,858	,105
	Hotelling's Trace	,117	,321 <sup>b</sup>	4,000	11,000	,858	,105
	Roy's Largest Root	,117	,321 <sup>b</sup>	4,000	11,000	,858	,105

a. Design: Intercept + avg\_ass\_gr

b. Exact statistic

**Levene's Test of Equality of Error Variances<sup>a</sup>**

	F	df1	df2	Sig.
Profitability	,012	1	14	,915
Dev_time	,392	1	14	,541
Dev_budget	,051	1	14	,824
Quality	,309	1	14	,587

Tests the null hypothesis that the error variance of the dependent variable is equal across groups.

a. Design: Intercept + Assets\_grouped

**Tests of Between-Subjects Effects**

Source	Dependent Variable	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	Profitability	,016 <sup>a</sup>	1	,016	,005	,947	,000
	Dev_time	1,562 <sup>b</sup>	1	1,562	,354	,561	,025
	Dev_budget	,000 <sup>c</sup>	1	,000	,000	1,000	,000
	Quality	,563 <sup>d</sup>	1	,563	,217	,649	,015
Intercept	Profitability	206,641	1	206,641	61,106	,000	,814
	Dev_time	264,063	1	264,063	59,801	,000	,810
	Dev_budget	336,111	1	336,111	80,667	,000	,852
	Quality	395,016	1	395,016	152,361	,000	,916
Assets_grouped	Profitability	,016	1	,016	,005	,947	,000
	Dev_time	1,562	1	1,562	,354	,561	,025
	Dev_budget	,000	1	,000	,000	1,000	,000
	Quality	,563	1	,563	,217	,649	,015
Error	Profitability	47,344	14	3,382			
	Dev_time	61,819	14	4,416			
	Dev_budget	58,333	14	4,167			
	Quality	36,297	14	2,593			
Total	Profitability	254,000	16				
	Dev_time	327,444	16				
	Dev_budget	394,444	16				
	Quality	431,875	16				
Corrected Total	Profitability	47,359	15				
	Dev_time	63,382	15				
	Dev_budget	58,333	15				
	Quality	36,859	15				

a. R Squared = ,000 (Adjusted R Squared = -,071)

b. R Squared = ,025 (Adjusted R Squared = -,045)

c. R Squared = ,000 (Adjusted R Squared = -,071)

d. R Squared = ,015 (Adjusted R Squared = -,055)

**Assets grouped**

Dependent Variable	Assets grouped	Mean	Std. Error	95% Confidence Interval	
				Lower Bound	Upper Bound
Profitability	1,00	3,625	,650	2,231	5,019
	2,00	3,563	,650	2,168	4,957
Dev_time	1,00	3,750	,743	2,157	5,343
	2,00	4,375	,743	2,782	5,968
Dev_budget	1,00	4,583	,722	3,035	6,131
	2,00	4,583	,722	3,035	6,131
Quality	1,00	5,156	,569	3,935	6,377
	2,00	4,781	,569	3,560	6,002



## Overall degree of game components on Project Performance

**Between-Subjects Factors**

		Value Label	N
Overall_comp_gr	1,00	low	7
	2,00	high	9

**Descriptive Statistics**

	Overall_comp_gr	Mean	Std. Deviation	N
Profitability	low	2,9643	1,71652	7
	high	4,0833	1,75891	9
	Total	3,5938	1,77688	16
Dev_time	low	2,8571	2,32425	7
	high	5,0000	1,26930	9
	Total	4,0625	2,05559	16
Dev_budget	low	3,3333	2,27710	7
	high	5,5556	,98601	9
	Total	4,5833	1,97203	16
Quality	low	5,1071	1,71912	7
	high	4,8611	1,53659	9
	Total	4,9688	1,56758	16

**Box's Test of Equality of Covariance Matrices<sup>a</sup>**

Box's M	8,826
F	,593
df1	10
df2	786,795
Sig.	,821

Tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups.

a. Design: Intercept + Overall\_comp\_gr

**Multivariate Tests<sup>a</sup>**

Effect		Value	F	Hypothesis df	Error df	Sig.	Partial Eta Squared
Intercept	Pillai's Trace	,973	97,737 <sup>b</sup>	4,000	11,000	,000	,973
	Wilks' Lambda	,027	97,737 <sup>b</sup>	4,000	11,000	,000	,973
	Hotelling's Trace	35,541	97,737 <sup>b</sup>	4,000	11,000	,000	,973
	Roy's Largest Root	35,541	97,737 <sup>b</sup>	4,000	11,000	,000	,973
Overall_comp _gr	Pillai's Trace	,610	4,293 <sup>b</sup>	4,000	11,000	,025	,610
	Wilks' Lambda	,390	4,293 <sup>b</sup>	4,000	11,000	,025	,610
	Hotelling's Trace	1,561	4,293 <sup>b</sup>	4,000	11,000	,025	,610
	Roy's Largest Root	1,561	4,293 <sup>b</sup>	4,000	11,000	,025	,610

a. Design: Intercept + Overall\_comp\_gr

b. Exact statistic

**Levene's Test of Equality of Error Variances<sup>a</sup>**

	F	df1	df2	Sig.
Profitability	,090	1	14	,768
Dev_time	2,936	1	14	,109
Dev_budget	5,458	1	14	,035
Quality	,247	1	14	,627

Tests the null hypothesis that the error variance of the dependent variable is equal across groups.

a. Design: Intercept + Overall\_comp\_gr

**Tests of Between-Subjects Effects**

Source	Dependent Variable	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	Profitability	4,931 <sup>a</sup>	1	4,931	1,627	,223	,104
	Dev_time	18,080 <sup>b</sup>	1	18,080	5,588	,033	,285
	Dev_budget	19,444 <sup>c</sup>	1	19,444	7,000	,019	,333
	Quality	,238 <sup>d</sup>	1	,238	,091	,767	,006
Intercept	Profitability	195,571	1	195,571	64,532	,000	,822
	Dev_time	243,080	1	243,080	75,122	,000	,843
	Dev_budget	311,111	1	311,111	112,000	,000	,889
	Quality	391,254	1	391,254	149,574	,000	,914
Overall_comp_gr	Profitability	4,931	1	4,931	1,627	,223	,104
	Dev_time	18,080	1	18,080	5,588	,033	,285
	Dev_budget	19,444	1	19,444	7,000	,019	,333
	Quality	,238	1	,238	,091	,767	,006
Error	Profitability	42,429	14	3,031			
	Dev_time	45,302	14	3,236			
	Dev_budget	38,889	14	2,778			
	Quality	36,621	14	2,616			
Total	Profitability	254,000	16				
	Dev_time	327,444	16				
	Dev_budget	394,444	16				
	Quality	431,875	16				
Corrected Total	Profitability	47,359	15				
	Dev_time	63,382	15				
	Dev_budget	58,333	15				
	Quality	36,859	15				

a. R Squared = ,104 (Adjusted R Squared = ,040)

b. R Squared = ,285 (Adjusted R Squared = ,234)

**Overall\_comp\_gr**

Dependent Variable	Overall_comp_gr	Mean	Std. Error	95% Confidence Interval	
				Lower Bound	Upper Bound
Profitability	low	2,964	,658	1,553	4,376
	high	4,083	,580	2,839	5,328
Dev_time	low	2,857	,680	1,399	4,315
	high	5,000	,600	3,714	6,286
Dev_budget	low	3,333	,630	1,982	4,684
	high	5,556	,556	4,364	6,747
Quality	low	5,107	,611	3,796	6,418
	high	4,861	,539	3,705	6,017

## Overall degree of Game Assets on Project Performance

**Between-Subjects Factors**

		Value Label	N
avg_ass_gr	1,00	low	8
	2,00	high	8

**Descriptive Statistics**

	avg_ass_gr	Mean	Std. Deviation	N
Profitability	low	3,6250	1,85646	8
	high	3,5625	1,82125	8
	Total	3,5938	1,77688	16
Dev_time	low	3,7500	1,90863	8
	high	4,3750	2,27783	8
	Total	4,0625	2,05559	16
Dev_budget	low	4,5833	2,02171	8
	high	4,5833	2,06059	8
	Total	4,5833	1,97203	16
Quality	low	5,1563	1,72657	8
	high	4,7813	1,48467	8
	Total	4,9688	1,56758	16

**Box's Test of Equality  
of Covariance  
Matrices<sup>a</sup>**

Box's M	8,792
F	,600
df1	10
df2	937,052
Sig.	,815

Tests the null hypothesis that the observed covariance matrices of the dependent variables are equal across groups.

a. Design: Intercept + avg\_ass\_gr

**Multivariate Tests<sup>a</sup>**

Effect		Value	F	Hypothesis df	Error df	Sig.	Partial Eta Squared
Intercept	Pillai's Trace	,959	64,332 <sup>b</sup>	4,000	11,000	,000	,959
	Wilks' Lambda	,041	64,332 <sup>b</sup>	4,000	11,000	,000	,959
	Hotelling's Trace	23,394	64,332 <sup>b</sup>	4,000	11,000	,000	,959
	Roy's Largest Root	23,394	64,332 <sup>b</sup>	4,000	11,000	,000	,959
avg_ass_gr	Pillai's Trace	,105	,321 <sup>b</sup>	4,000	11,000	,858	,105
	Wilks' Lambda	,895	,321 <sup>b</sup>	4,000	11,000	,858	,105
	Hotelling's Trace	,117	,321 <sup>b</sup>	4,000	11,000	,858	,105
	Roy's Largest Root	,117	,321 <sup>b</sup>	4,000	11,000	,858	,105

a. Design: Intercept + avg\_ass\_gr

b. Exact statistic

**Levene's Test of Equality of Error Variances<sup>a</sup>**

	F	df1	df2	Sig.
Profitability	,012	1	14	,915
Dev_time	,392	1	14	,541
Dev_budget	,051	1	14	,824
Quality	,309	1	14	,587

Tests the null hypothesis that the error variance of the dependent variable is equal across groups.

a. Design: Intercept + avg\_ass\_gr

**Tests of Between-Subjects Effects**

Source	Dependent Variable	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	Profitability	,016 <sup>a</sup>	1	,016	,005	,947	,000
	Dev_time	1,562 <sup>b</sup>	1	1,562	,354	,561	,025
	Dev_budget	,000 <sup>c</sup>	1	,000	,000	1,000	,000
	Quality	,563 <sup>d</sup>	1	,563	,217	,649	,015
Intercept	Profitability	206,641	1	206,641	61,106	,000	,814
	Dev_time	264,063	1	264,063	59,801	,000	,810
	Dev_budget	336,111	1	336,111	80,667	,000	,852
	Quality	395,016	1	395,016	152,361	,000	,916
avg_ass_gr	Profitability	,016	1	,016	,005	,947	,000
	Dev_time	1,562	1	1,562	,354	,561	,025
	Dev_budget	,000	1	,000	,000	1,000	,000
	Quality	,563	1	,563	,217	,649	,015
Error	Profitability	47,344	14	3,382			
	Dev_time	61,819	14	4,416			
	Dev_budget	58,333	14	4,167			
	Quality	36,297	14	2,593			
Total	Profitability	254,000	16				
	Dev_time	327,444	16				
	Dev_budget	394,444	16				
	Quality	431,875	16				
Corrected Total	Profitability	47,359	15				
	Dev_time	63,382	15				
	Dev_budget	58,333	15				
	Quality	36,859	15				

a. R Squared = ,000 (Adjusted R Squared = -,071)

b. R Squared = ,025 (Adjusted R Squared = -,045)

c. R Squared = ,000 (Adjusted R Squared = -,071)

d. R Squared = ,015 (Adjusted R Squared = -,055)

avg\_ass\_gr

Dependent Variable	avg_ass_gr	Mean	Std. Error	95% Confidence Interval	
				Lower Bound	Upper Bound
Profitability	low	3,625	,650	2,231	5,019
	high	3,563	,650	2,168	4,957
Dev_time	low	3,750	,743	2,157	5,343
	high	4,375	,743	2,782	5,968
Dev_budget	low	4,583	,722	3,035	6,131
	high	4,583	,722	3,035	6,131
Quality	low	5,156	,569	3,935	6,377
	high	4,781	,569	3,560	6,002

## References:

100 Most Popular Game Engines - Mod DB. (2015, July 31). Retrieved August 5, 2015, from <http://www.moddb.com/engines/top>

Ajila, S. A., & Wu, D. (2007). Empirical study of the effects of open source adoption on software development economics. *Journal of Systems and Software*, 80(9), 1517–1529.  
doi:10.1016/j.jss.2007.01.011

Atuahene-Gima, K. (2003). The Effects of Centrifugal and Centripetal Forces on Product Development Speed and Quality: How Does Problem Solving Matter? *Academy of Management Journal*, 46(3), 359–373. doi:10.2307/30040629

Baldassarre, M. T., Bianchi, A., Caivano, D., & Visaggio, G. (2005). An industrial case study on reuse oriented development (pp. 283–292). IEEE. doi:10.1109/ICSM.2005.20

Baldwin, C. Y., & Clark, K. B. (1997). Managing in an age of modularity. *Harvard Business Review*, 75(5), 84–93.

Barney, J. (1991). Firm Resources and Sustained Competitive Advantage. *Journal of Management*, 17, 99–120.

Barns, B. H., & Bollinger, T. B. (1991). Making reuse cost-effective. *IEEE Software*, 8(1), 13–24.  
doi:10.1109/52.62928

Basili, V. R. (1990). Viewing maintenance as reuse-oriented software development. *IEEE Software*, 7(1), 19–25. doi:10.1109/52.43045

Basili, V. R. (1990). Viewing maintenance as reuse-oriented software development. *Software, IEEE*, 7(1), 19–25.

Basili, V. R., Briand, L. C., & Melo, W. L. (1996). How reuse influences productivity in object-oriented systems. *Communications of the ACM*, 39(10), 104–116.

Basili, V. R., & Rombach, H. D. (1988). The TAME project: towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6), 758–773.  
doi:10.1109/32.6156

- Biggerstaff, T. J., & Perlis, A. J. (Eds.). (1989). *Software reusability*. New York, N.Y. : Reading, Mass: ACM Press ; Addison-Wesley.
- Chau, P. Y. K. (1999). On the use of construct reliability in MIS research: a meta-analysis. *Information & Management*, 35(4), 217–227. doi:10.1016/S0378-7206(98)00089-5
- Cohen, J. (Ed.). (2003). *Applied multiple regression/correlation analysis for the behavioral sciences* (3rd ed.). Mahwah, N.J: L. Erlbaum Associates.
- Cooper, D. R., & Schindler, P. S. (2014). *Business research methods* (12th ed.). New York: McGraw-Hill.
- Cooper, R. G., & Kleinschmidt, E. J. (1987). New Products: What Separates Winners from Losers? *Journal of Product Innovation Management*, 4(3), 169–184. doi:10.1111/1540-5885.430169
- Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika*, 16(3), 297–334. doi:10.1007/BF02310555
- de O. Melo, C., S. Cruzes, D., Kon, F., & Conradi, R. (2013). Interpretative case studies on agile team productivity and management. *Information and Software Technology*, 55(2), 412–427. doi:10.1016/j.infsof.2012.09.004
- Ezran, M., Morisio, M., & Tully, C. J. (2002). *Practical Software Reuse*. London: Springer London : Imprint : Springer. Retrieved from <http://dx.doi.org/10.1007/978-1-4471-0141-3>
- Forecast: Video Game Ecosystem, Worldwide, 4Q13. (2015, January 15). Retrieved January 15, 2015, from <https://www.gartner.com/doc/2606315/forecast-video-game-ecosystem-worldwide>
- Frakes, W. B., & Isoda, S. (1994). Success factors of systematic reuse. *IEEE Software*, 11(5), 14–19. doi:10.1109/52.311045
- Frakes, W., & Terry, C. (1996). Software reuse: metrics and models. *ACM Computing Surveys (CSUR)*, 28(2), 415–435.
- Gamasutra: Mark DeLoura's Blog - The Engine Survey: Technology Results. (2009, March 16). Retrieved April 19, 2015, from



- [http://www.gamasutra.com/blogs/MarkDeLoura/20090316/903/The\\_Engine\\_Survey\\_Technology\\_Results.php](http://www.gamasutra.com/blogs/MarkDeLoura/20090316/903/The_Engine_Survey_Technology_Results.php)
- Games to cost \$60m, says Ubisoft boss - Eurogamer.net. (2009, June 16). Retrieved November 21, 2014, from <http://www.eurogamer.net/articles/games-to-cost-USD60m-says-ubisoft-boss>
- Game Systems | HeroEngine. (2012). Retrieved March 19, 2014, from <http://www.heroengine.com/heroengine/game-systems/>
- Gregory, J. (2009). *Game Engine Architecture*. Taylor & Francis.
- ISO/IEC Standard for Systems and Software Engineering - Recommended Practice for Architectural Description of Software-Intensive Systems. (2007). *ISO/IEC 42010 IEEE Standard 1471-2000*. doi:10.1109/IEEESTD.2007.386501
- Jones, T. C. (1984). Reusability in Programming: A Survey of the State of the Art. *Software Engineering, IEEE Transactions on*, (5), 488–494.
- Kessler, E. (1999). Speeding up the pace of new product development. *Journal of Product Innovation Management*, 16(3), 231–247. doi:10.1016/S0737-6782(98)00048-4
- Lim, W. C. (1994). Effects of reuse on quality, productivity, and economics. *IEEE Software*, 11(5), 23–30. doi:10.1109/52.311048
- Marketplace - UE4 Marketplace. (2015). Retrieved August 6, 2015, from <https://www.unrealengine.com/marketplace>
- Matt Chat 99: Duke Nukem with Scott Miller - YouTube. (2014, October 12). Retrieved December 10, 2013, from [http://www.youtube.com/watch?v=n\\_i3-aoHnww](http://www.youtube.com/watch?v=n_i3-aoHnww)
- McGuire, M., & Chadwicke Jenkins, O. (2008). *Creating Games: Mechanics, Content, and Technology*. A K Peters/CRC Press. Retrieved from [http://www.amazon.com/gp/product/1568813058/ref=as\\_li\\_tf\\_tl?ie=UTF8&tag=casueffe06-20&linkCode=as2&camp=1789&creative=9325&creativeASIN=1568813058](http://www.amazon.com/gp/product/1568813058/ref=as_li_tf_tl?ie=UTF8&tag=casueffe06-20&linkCode=as2&camp=1789&creative=9325&creativeASIN=1568813058)
- Miguel, P. A. C. (2005). Modularity in product development: a literature review towards a research agenda. *Product: Management & Development*, 3(2), 165–174.

- Mohagheghi, P., & Conradi, R. (2007). Quality, productivity and economic benefits of software reuse: a review of industrial studies. *Empirical Software Engineering*, 12(5), 471–516.  
doi:10.1007/s10664-007-9040-x
- Mohagheghi, P., Conradi, R., Killi, O. M., & Schwarz, H. (2004). An empirical study of software reuse vs. defect-density and stability (pp. 282–291). IEEE Comput. Soc.  
doi:10.1109/ICSE.2004.1317450
- Morisio, M., Romano, D., & Stamelos, I. (2002). Quality, productivity, and learning in framework-based development: an exploratory case study. *IEEE Transactions on Software Engineering*, 28(9), 876–888. doi:10.1109/TSE.2002.1033227
- Morris, D., Donnelly, T., & Donnelly, T. (2004). Supplier parks in the automotive industry. *Supply Chain Management: An International Journal*, 9(2), 129–133.  
doi:10.1108/13598540410527024
- Philip A. Bernstein. (1993). *Middleware An Architecture for Distributed System Services*. Digital Equipment Corporation 1993.
- Rollings, A., & Morris, D. (2004). *Game architecture and design: a new edition*. Indianapolis, Ind.: New Riders.
- Sahay, A., & Riley, D. (2003). The Role of Resource Access, Market Considerations, and the Nature of Innovation in Pursuit of Standards in the New Product Development Process. *Journal of Product Innovation Management*, 20(5), 338–355. doi:10.1111/1540-5885.00033
- Sametinger, J. (1997). *Software engineering with reusable components*. Springer.
- Sanchez, R., & Mahoney, J. T. (1996). Modularity, flexibility, and knowledge management in product and organizational design. *Strategic Management Journal*, 17, 63–76.
- Schach, S. R. (2011). *Object-oriented and classical software engineering*. New York: McGraw-Hill.
- Schilling, M. A. (2000). Toward a general modular systems theory and its application to interfirm product modularity. *Academy of Management Review*, 25(2), 312–334.

- Schmidt, H. G., Crnkovic, I., & Heineman, G. T. (2007). *Component-Based Software Engineering: 10th International Symposium, CBSE 2007, Medford, MA, USA, July 9-11, 2007, Proceedings*. Springer.
- Selby, R. W. (2005). Enabling reuse-based software development of large-scale systems. *IEEE Transactions on Software Engineering*, 31(6), 495–510. doi:10.1109/TSE.2005.69
- Sommerville, I. (2011). *Software engineering* (9th ed.). Boston: Pearson.
- Song, X. M., & Parry, M. E. (1997). A cross-national comparative study of new product development processes: Japan and the United States. *The Journal of Marketing*, 1(18).
- Stevens, P., Myers, J., & Constantine, L. (1974). Structured design. *IBM Systems Journal*, Volume 13(Issue 2), 115–139.
- Szyperski, C., Gruntz, D., & Murer, S. (2002). *Component Software: Beyond Object-Oriented Programming* (2nd ed.). Addison-Wesley.
- Taivalsaari, A. (1993). *A critical view of inheritance and reusability in object-oriented programming*. Jyväskylä [Finland]: University of Jyväskylä.
- The evolution of PC graphics will blow your mind | TechRadar. (2015, March 27). Retrieved August 5, 2015, from <http://www.techradar.com/news/gaming/the-evolution-of-pc-graphics-will-blow-your-mind-1289593>
- Thomas, W. M., Delis, A., & Basili, V. R. (1997). An analysis of errors in a reuse-oriented development environment. *Journal of Systems and Software*, 38(3), 211–224. doi:10.1016/S0164-1212(96)00152-5
- Tiwana, A. (2008). Does technological modularity substitute for control? A study of alliance performance in software outsourcing. *Strategic Management Journal*, 29(7), 769–780. doi:10.1002/smj.673
- Tracz, W. (1994). Software reuse myths revisited (pp. 271–272). IEEE Comput. Soc. Press. doi:10.1109/ICSE.1994.296788

Ulrich, K. (1995). The role of product architecture in the manufacturing firm. *Research Policy*, 24(3), 419–440.

Unity3d Asset Store. (2015). Retrieved July 23, 2015, from <https://www.assetstore.unity3d.com/en/>

van Beek, J. B., & Valient, M. (2011). Guerrilla Games - Publications - The Creation of Killzone 3.

Retrieved July 21, 2015, from <http://www.guerrilla-games.com/publications.html>

Visceral Games Speaks Out on Battlefield Hardline Re-Using Battlefield 4 Assets. (2014, June 17).

Retrieved July 21, 2015, from <http://www.playstationlifestyle.net/2014/06/17/visceral-games-speaks-out-on-battlefield-hardline-re-using-battlefield-4-assets/>

Will the Wii be a set-top box? - CNET. (2007, November 24). Retrieved November 20, 2014, from

<http://www.cnet.com/news/will-the-wii-be-a-set-top-box/>

Worren, N., Moore, K., & Cardona, P. (2002). Modularity, strategic flexibility, and firm performance: a

study of the home appliance industry: Modularity, Strategic Flexibility and Firm Performance.

*Strategic Management Journal*, 23(12), 1123–1140. doi:10.1002/smj.276