Serious Games for Learning

A model and a reference architecture for efficient game development

Maira Brandao Carvalho

A catalogue record is available from the Eindhoven University of Technology Library

Serious Games for Learning: A model and a reference architecture for efficient game development/ by Maira Brandao Carvalho. Eindhoven : Technische Universiteit Eindhoven, 2016
Proefschrift. – ISBN: xxx

Typeset with LaTeX
Cover design: Maira Brandao Carvalho
Cover images: © Kuanchong Ng | Dreamstime.com
Printed by GVO, The Netherlands

Serious Games for Learning

A model and a reference architecture for efficient game development

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Eindhoven,
op gezag van de rector magnificus, prof.dr.ir. F.P.T. Baaijens,
voor een commissie aangewezen door het College voor Promoties
in het openbaar te verdedigen
op donderdag 8 december 2016 om 14.00 uur

door

Maira Brandao Carvalho

geboren te Brasilia, Brazilië

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof. dr. ir. A.C. Brombacher
1e promotor:	prof. dr. A. De Gloria (Università Degli Studi di Genova)
2e promotor:	prof. dr. G.W.M. Rauterberg
1e copromotor:	dr. J. Hu
2e copromotor:	dr. F. Bellotti (Università Degli Studi di Genova)
leden:	dr. P.H.M. Spronck (Tilburg Universiteit)
	prof. dr. B.A.M. Schouten
	prof. dr. D. Beijaard

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

UNIVERSITÀ DEGLI STUDI
DI GENOVA

TU/e Technische Universiteit Eindhoven University of Technology

The work in this dissertation has been carried out under the auspices of Erasmus Mundus Doctorate Program in Interactive and Cognitive Environments (ICE). This work was conducted towards a joint double PhD degree affiliated with the following partner universities:

UNIVERSITÀ DEGLI STUDI DI GENOVA

TECHNISCHE UNIVERSITEIT EINDHOVEN

Acknowledgments

This PhD Thesis has been developed in the framework of, and according to, the rules of the Erasmus Mundus Joint Doctorate on Interactive and Cognitive Environments EMJD ICE [FPA nº 2010-0012] with the cooperation of the following Universities:

Alpen-Adria-Universität Klagenfurt – AAU

Queen Mary, University of London – QMUL

Technische Universiteit Eindhoven – TU/e

Università degli Studi di Genova – UNIGE

Universitat Politècnica de Catalunya – UPC

According to ICE regulations, the Italian PhD title has also been awarded by the Università degli Studi di Genova.

To my family, whom I love like friends.
To my friends, whom like family I love.
To my love, my family and friend.

Summary

Digital Serious Games (SGs) are gaining increasing importance as educational and training tools. However, there is still a long way to make them widely deployed. On the one hand, balancing fun and educational elements in a SG is not trivial and requires understanding how these games can be designed to support effective and efficient learning. On the other hand, actual development can be costly and time-consuming, involving large teams of people from different backgrounds, who often do not share common vocabularies and work processes.

The objective of this research is, thus, to support the design and development of digital educational SGs, by helping reduce the costs associated with SG development, while fulfilling the game's educational and entertainment goals.

For that end, both the conceptual and the technical complexities of producing educational SGs are addressed.

A theoretical analysis is performed to determine how a game realizes its educational and entertainment objectives through its concrete mechanics. A conceptual model, called Activity Theory-based Model for Serious Games (ATMSG), is presented. The model aims to support a systematic and detailed representation of educational SGs, depicting the ways that game elements are connected to each other throughout the game, and how these elements contribute to the achievement of the game's desired pedagogical goals. The ATMSG fills a gap that previous models, frameworks and methodologies for SG analysis and design do not cover, namely the understanding of how high-level game requirements, concerning both educational and entertainment goals, can be concretely satisfied in a SG, down to the game mechanics level.

The ATMSG model is used as a starting point in the definition of a software reference architecture called Service-Oriented Reference Architecture for Serious Games (SORASG), based on the architectural pattern of Service-Oriented Architecture (SOA). The architecture provides a template solution to support SG software development, defining common elements among different SGs of different genres and topics that can be incorporated in a modular fashion into game architectures. The reference architecture is evaluated in two ways: by employing the Architecture Trade-off Analysis Method (ATAM); and through the development of an example implementation, demonstrating how game-independent services can be integrated in existing games in a relatively short time, while meeting the desired educational and entertainment goals.

The results suggest that the proposed methodology and tools (i.e., ATMSG identification of components, and their implementation and integration as services) can offer a valuable solution to reduce costs and foment quality SG development.

Contents

List of Figures

List of Tables

Acronyms

ADD Attribute Driven Design

AI artificial intelligence

ASR architecturally significant requirement

ATAM Architecture Trade-off Analysis Method

ATMSG Activity Theory-based Model for Serious Games

CAI computer-assisted instruction

CbKST Competence-based Knowledge Space Theory

COTS commercial off-the-shelf

CSV Comma-Separated Values

ECD evidence-centered assessment design

EEG electroencephalography

GBL game-based learning

GOM Game Object Model

GOM II Game Object Model II

GOP Game Ontology Project

HABS Hierarchical Activity-Based Scenario

HCI human-computer interaction

HTTP Hypertext Transfer Protocol

ITS intelligent tutoring system

KST Knowledge Space Theory

LM-GM Learning Mechanics–Game Mechanics

LMS learning management system

LRS Learning Record Store

MARL Mobile Augmented Reality Learning

MDA Model, Dynamics, Mechanics

OLM open learner model

QA quality attribute

REST Representation State Transfer

RETAIN Relevance, Embedding, Transfer, Adaption, Immersion and Naturalisation

SCORM Sharable Content Object Reference Model

SG serious game

SGM Serious Game Mechanic

SOA Service-Oriented Architecture

SOAP Simple Object Access Protocol

SORASG Service-Oriented Reference Architecture for Serious Games

TEL technology-enhanced education

TiE Travel in Europe

UML Unified Modeling Language

xAPI Experience API

A note on style

In this thesis, I use both "I" and "we" as the first person pronoun, using the following rationale.

I use "I" when I present decisions and choices I made and insights that are my own. I also use "I" when outlining the organization of the text ("In Chapter 2, **I** discuss the foundations of the work that has been developed in this thesis...").

I use "we" to indicate work that was carried out in collaboration with colleagues, particularly work that has been published as joint research papers. I indicate, as a footnote in the beginning of a chapter, the reference to published article(s), whenever relevant. Sometimes I also use "we" in explanatory discourse. In these cases, "we" stands for the reader and me ("[...] in the corresponding cell [...], **we** can read a more thorough description...").

Chapter 1

Introduction

Research indicates that video game playing can enhance performance in a wide variety of tasks and cognitive skills (Boot, Kramer, Simons, Fabiani, & Gratton, 2008), e.g. allocation of attentional resources (Green, Li, & Bavelier, 2010), task-switching (Oei & Patterson, 2014), creativity (Jackson et al., 2012), problem-solving (Shute, Ventura, & Ke, 2015), and spatial skills (Green & Bavelier, 2007; Shute et al., 2015), in addition to other non-cognitive skills such as persistence (Shute et al., 2015) and openness to experiences (Ventura, Shute, & Y. J. Kim, 2012). Such a wide variety of effects caused a justifiable interest in exploiting games for purposes other than to just entertain.

Such is the interest in the application of games for non-entertainment purposes that they have earned a special name: serious games (SGs) (Susi, Johannesson, & Backlund, 2007; Djaouti, Alvarez, Jessel, & Rampnoux, 2011). The term started to gain popularity in 2002, with the publication of a white paper by Sawyer and Rejeski (2002) in which the authors highlighted the advantages of using knowledge from the entertainment industry to improve the area of public policy, and the coincidental release of the game "America's Army", considered as the first commercially successful SG (Djaouti et al., 2011).

Of particular interest in the field of SG studies is the application of games in education, or game-based learning (GBL) (Eck, 2006; Blunt, 2009; Bellotti, Berta, & De Gloria, 2010). There is growing evidence that supports the use of games as tools to complement or enhance traditional education. Games can make learning more engaging and satisfying (Eck, 2006; Kickmeier-Rust & Albert, 2012b). They also offer the possibility to expose learners to experiences that would be impossible, unsafe or at least impractical to reproduce in the real world (Susi et al., 2007; Knight et al., 2010; Oliveira, Coelho, Guimarães, & Rebelo, 2012). Several studies comparing games to more common teaching methods point to the fact that games can provide an enhanced experience (Bellotti et al., 2010; Knight et al., 2010; Kebritchi, Hirumi, & Bai, 2010; Guillén-Nieto & Aleson-Carbonell, 2012; Kickmeier-Rust & Albert, 2012b; Erhel & Jamet, 2013). A relatively recent meta-analysis of the field confirmed that training with SGs is

more effective than training with conventional instructional methods, particularly due to improved retention (Wouters, Nimwegen, Oostendorp, & Spek, 2013).

With many evidences in its favor, but still with a long road ahead towards widespread adoption, the field of SG design and development offers a promising avenue for improved application of technology for education.

Before continuing, a clarification regarding terminology is needed. Among the many possible applications of SGs, arguably the use of games for education occupies the most prominent place. In fact, the term *serious game* is often used to refer specifically to the class of digital games intentionally created for teaching some skills, knowledge, competence, or promoting a specific attitude towards a topic, as an instructional tool in itself, or as part of a larger curriculum of activities. To conform to this understanding, in this thesis I use the term *serious game (SG)* to refer specifically to digital games created for educational purposes. To avoid confusion, when referring to other classes of SGs (e.g. persuasive games for marketing or advertising (Bogost, 2007), games for health care, games for scientific discovery), I follow the terminology proposed by R. Schmidt, Emmerich, and B. Schmidt (2015) and use the term *applied games* instead.

1.1 Issues in SG design and development

Despite the increasing interest in SGs (Cheng, Chen, Chu, & Chen, 2015; Riedel, Feng, Hauge, Hansen, & Tasuya, 2015), the deployment of SGs in real educational settings is still low. In the industry, companies appear not completely convinced of the benefits of SGs for corporate training, possibly due to a risk-averse attitude towards new technologies (Azadegan, Riedel, & Baalsrud Hauge, 2012), or due to the perception that using such tools can be costly (Batko, 2016). In schools, barriers to adoption include not only negative perceptions towards the educational value of games but also the difficulty of providing good enough games to keep students interested (Egenfeldt-Nielsen, 2006; Rice, 2007).

Indeed, developing quality games for learning can be a complex, challenging and costly process (Boyle et al., 2016), from both a conceptual and a technical standpoints.

SG design is challenging because the principles of learning and gameplay are different and frequently conflicting (Arnab et al., 2015). The term *serious game* itself evokes this conflict, since, by definition, play is a voluntary and unproductive activity that brings joy and amusement (Huizinga, 1949; Caillois & Barash, 1961); thus, trying to combine seriousness with play brings not only technical, but especially conceptual concerns. A particular challenge is linking pedagogical practices to concrete gaming aspects that can realize those practices (Arnab et al., 2015). Communication between the multidisciplinary teams that are involved in the design and development of SGs can also be problematic, due to a lack of common vocabularies that facilitate cooperation (Marne, Wisdom, Huynh-Kim-Bang, & Labat, 2012).

Technically, game development is also a daunting task, for which there is little guidance from the software architectural and development aspects (Scacchi & Cooper, 2015). Although there are many libraries and game engines for commercial game development, this environment is very heterogeneous and can present challenges in deciding how to best combine components and sub-system architectures (Scacchi & Cooper, 2015).

Furthermore, due to their specific learning requirements, SGs are typically conceived as one-of-a-kind products, fully tailored to the clients' requirements. As a consequence, these games have low reusability of the final product and of its components (Stanescu et al., 2014). Coupled with high production costs and challenging and time-consuming production processes, it is not difficult to see why adoption is still low.

1.2 Research questions

The current state of the SG development field causes a high barrier to SG adoption by schools and the industry. Development costs are high, but there is no straightforward way to enable an economy of scale in SG development because each game has unique entertainment and educational requirements – and trying to force a one-size-fits-all solution can seriously compromise the quality of the SG. Which bring us to the central issue of this thesis, formulated in the following problem statement (PS):

PS *How can we reduce costs associated with SG development, while fulfilling the game's educational and entertainment goals?*

The term "cost", in this context, refers to the effort required to develop or maintain software. It can be understood as hours of work per person and/or amount of money required to carry out the project (Heemstra, 1992).

I define three research questions (RQ) to help solve that problem.

RQ1 *How does a game realize its educational and entertainment objectives through its concrete mechanics?*

This question aims to reconcile educational theory with game development, helping us understand how concrete components of a SG can be defined, used and combined to support efficient learning. In this way, the proposed technological solutions for reducing development costs would still reflect the theoretical knowledge that has been accumulated about the subject so far.

To answer this question, I present the investigation of how a game realizes its educational and entertainment objectives through its mechanics, to uncover how the practice of game

and SG design is linked to the theories behind it. It is an effort to investigate SGs from the inside, classifying their inner components to find out how they relate to each other and to the educational and entertainment objectives of the game.

The result of this investigation is a new model that connects a game's educational and entertainment high-level objectives with low-level in-game components, and links individual gaming and pedagogical components as the game unfolds. The model I propose, named Activity Theory-based Model for Serious Games (ATMSG), is based on concepts of activity theory, a line of research in the social sciences that studies different forms of human practices and development processes (Jonassen & Rohrer-Murphy, 1999). Using activity theory allows us to consider the game not as an isolated tool, but as part of a complex system that also includes human actors (players or learners, instructors and game designers) and the motives driving their interactions with the game. The ATMSG model includes a SG components taxonomy, which is based on established taxonomies of learning, of instructional design, and of game components.

RQ2 *To which extent can SG development be simplified by reusing existing technological solutions, even the ones that were not created specifically for SGs?*

The ATMSG model provides a starting point to try to identify common elements among different SGs of different genres and topics. Could these elements be incorporated in a modular fashion into game architectures, thus allowing developers to reuse components?

The rationale for suggesting the reuse of technology is not only to speed up game development, but also to provide the SG developer with access to quality and up-to-date components, created and maintained by experts in the field. These components may apply particular theories or optimizations that are too specialized to develop from scratch. The SG developer may thus better focus on specific features and pedagogical aspects of the game itself.

RQ3 *How can SG developers incorporate reusable components into their software development projects?*

Assuming that it is possible to identify components that can be reused across different games, SG designers and developers would still need guidance on how to incorporate them into their projects. The answer to this challenge is the proposition of a software reference architecture called Service-Oriented Reference Architecture for Serious Games (SORASG), based on the Service-Oriented Architecture (SOA) pattern. A reference architecture is a template solution for a system architecture in a particular domain. There are examples of reference architectures in several domains, ranging from purely technical industry-wide architectures (OATH, 2007) to applied systems such as adaptive hypermedia (Wu, 2002) and simulations (Kuhl, Weatherly, & Dahmann, 1999); however, the examples in the field of SG software development are scarce. The SORASG fills this gap.

The SORASG has an additional goal of promoting the use of open standards and technology-independent solutions in the development of SGs. The use of open standards facilitates competition among suppliers by reducing barriers to entering the market, which in turn encourages the spread of the technology and stimulates further innovation (Maxwell, 2006). The SG industry is still relatively new; consequently, there is still room for more and better solutions to support effective learning through games.

In the literature on software engineering, business goals are seen as the expression of the reasons that an organization has for developing a system. Business goals establish the connection between the software artifact and an organization's missions and ambitions. Often such concerns are related to making profits, but they can also refer to other goals such as increasing number of customers, enhancing customer satisfaction, reducing operation costs, and so on. It is important to have clearly defined business goals at the beginning of any software-related project, as these goals establish a starting point for defining requirements and a clear reference to measure the results (Bass, Clements, & Kazman, 2012).

Thus, I summarize the three main business goals (BG) guiding the elaboration of the SORASG as follows:

> **BG1** *Reduce the costs associated with the development of SGs, while maintaining the quality of the games developed.*
>
> **BG2** *Allow the reuse of existing technological solutions, even the ones that were not created specifically for SGs.*
>
> **BG3** *Promote the use of open standards and technology-independent solutions.*

1.3 Thesis outline

In this introduction, I give a brief overview of the current state of the use of games for educational purposes. I describe one of the main issues that affects the adoption of SGs by educators and by the industry: the high complexity (and consequently high cost) of SG development. Given this scenario, I outline my research objectives in the form of three research questions that I answer throughout the thesis.

In Chapter 2, I discuss the foundations of the work that has been developed in this thesis. I review existing models, frameworks, and methodologies for the analysis and design of SGs and explain why they do not answer RQ1. Then, I discuss the importance of assessment, feedback, and adaptation in the context of SGs. Subsequently, I present the fundamental ideas of activity theory, a line of research in the social sciences that served as basis for the theoretical model proposed in this work.

In Chapter 3, I describe in detail my response to RQ1: a model for SG analysis and design called Activity Theory-based Model for Serious Games (ATMSG), and a taxonomy of

SG elements. The ATMSG model connects a game's educational and entertainment high-level objectives with low-level in-game components on the one hand, and links individual gaming and pedagogical components as the game unfolds on the other.

Chapter 4 moves to technical considerations on SG software development. I focus, in particular, on the role of a game's software architecture in the development process. I introduce concepts of software engineering that are necessary to understand the reference architecture proposed later in the thesis.

Chapter 5 delineates the Service-Oriented Reference Architecture for Serious Games (SORASG), which is a reference architecture for SGs proposed as my answer to RQ3. I also answer RQ2 in Subsection 5.3.1, showing how the ATMSG model serves as the basis for identifying common elements among different SGs of different genres and topics that can be incorporated by game developers as pieces that can be reused. I evaluate the SORASG against its objectives and discuss its quality and applicability. Furthermore, I discuss how the two main outcomes of this thesis – the ATMSG model and the SORASG – relate to each other.

In Chapter 6, I present my concluding remarks. I summarize the work and implications of the research in relation to the problem statement and research questions. Finally, I offer a brief discussion of the limitations of the work and pointers for future research on the topic.

Chapter 2

Literature review

This chapter discusses the foundations of the work that has been developed in this thesis.

First of all, I discuss models, frameworks, and methodologies that have been created so far as attempts to analyze educational serious games (SGs). I also explain why the existing works do not provide a satisfactory answer to RQ1.

Subsequently, I discuss the role of assessment, adaptation and feedback in SGs, highlighting their importance in the educational effectiveness of games for education and why they occupy an important role in answering RQ2.

Finally, I present the fundamental ideas of activity theory, a line of research in the social sciences that provided the basis for the investigation on how concrete components of a SG are defined, used and combined to support efficient learning, resulting in the model presented in Chapter 3.

2.1 Analysing serious games

In an attempt to uncover the reasons behind the success or failures of educational SGs, researchers have developed models, frameworks and methodologies to investigate and analyze games (Amory, 2007; Arnab et al., 2015; Bellotti, Berta, De Gloria, & Primavera, 2010; Freitas & Oliver, 2006; Gunter, Kenny, & Vick, 2006; Staalduinen & Freitas, 2011). The most prominent works were examined, in order to evaluate how well they could support the understanding of the deeper relationships between different components in educational SGs.

Parts of this chapter were published previously in Carvalho, Bellotti, Berta, Gloria, Sedano, et al. (2015).

The Model, Dynamics, Mechanics (MDA) framework (Hunicke, Leblanc, & Zubek, 2004) proposes three perspectives from which to understand and design games: the actual implementation of the game (Mechanics), the overarching design goals (Dynamics) and the resulting player's experience (Aesthetics). MDA is aimed at games in general; consequently, it does not explicitly support reasoning about the educational elements in a SG. Salen and Zimmerman (2004) use the concept of systems and elements (objects, attributes, relationships and environment) and define three framing levels for gaming systems (formal, experiential and cultural) to help game designers focus on specific problems without losing sight of the whole. The Hierarchical Activity-Based Scenario (HABS) framework (Marsh, 2006; Marsh, 2010) also examines games using a layered perspective, but from the viewpoint of the game's narrative and players' experiences and behaviors. HABS uses activity theory to help designers in defining levels of the user experience when modeling game scenarios and narratives. Marsh and Nardi (2014) later expanded the framework to account for user engagement and entertainment including interactions that extend beyond the game world. HABS provides valuable support for developing a high-level set of ideas and concepts for gameplay (Marsh, 2010). Nonetheless, it does not account explicitly for the interaction between gaming and learning, nor does it represent specific elements that compose the SG.

While MDA and HABS deal with the relationships between different layers of implementation of the game, other frameworks and models focus on the description of low-level components. Koster (2005a) and Bura (2006) attempted to define a compact visual language for communicating underlying principles of games. Inspired by their work, Djaouti, Alvarez, Jessel, and Methel (2007) created a diagram language to deconstruct formally video games into "game bricks". The framework Machinations (Adams & Dormans, 2012) describes a dynamic graphical notation representing games as rule-based systems (Dormans, 2009). The Game Ontology Project (GOP) (Zagal, Mateas, Fernández-Vara, Hochhalter, & Lichti, 2005) defines a hierarchical representation of the important structural elements of games in an attempt to establish a common vocabulary to the field. All these works complement each other and contribute to the effort of creating formal, precise, and scalable descriptions of games and gameplay (Sicart, 2008). However, they are limited to describing games in general, without incorporating educational elements.

Some of the models we reviewed specifically look at the educational value of SGs. The Four-Dimensional Framework (Freitas & Oliver, 2006) postulates four dimensions of learning processes that need to be considered: learner modeling and profiling, the role of pedagogic approaches for supporting learning, the representation of the game, and the context in which learning takes place. The Relevance, Embedding, Transfer, Adaption, Immersion and Naturalisation (RETAIN) model (Gunter et al., 2006) aims at determining whether a SG is appropriate for educational purposes, how well the pedagogical content is embedded in the game's narrative and how it promotes knowledge transfer. The Experiential Gaming Model proposed by Kiili (2005) assigns central importance to linking experiential learning and gameplay theory, since this connection facilitates the flow experience and has positive impact on learning. The Game-based Learning Frame-

work (Staalduinen & Freitas, 2011) also focuses on immersive learning experiences, with a structure that resembles Kolb's experiential learning cycle (Kolb, 1984). These give a general understanding of a SG, facilitate the comparison with other similar games, and possibly help us determine how well the SG fits an educator's needs. The main limitation of the frameworks above is that none of them investigates the actual elements of the game, but rather focused on high-level aspects.

However, there are works that investigate SGs in a more fine-grained manner. The Game Object Model II (GOM II) (Amory, 2007), an updated version of the original GOM model (Amory, Naicker, Vincent, & Adams, 1999), describes the relationships between game and pedagogical elements using the metaphor of interfaces in the Object Oriented Programming paradigm: abstract interfaces represent the theoretical constructs and the pedagogical goals of the game; concrete interfaces represent the design elements that realize the goals. However, GOM II does not represent how the relationship between game elements develops over time, and its diagram can become complex and difficult to understand. The Learning Mechanics–Game Mechanics (LM-GM) model (Arnab et al., 2015) provides a graphical representation of the game flow as the basis for establishing the relationships between the components that translate pedagogical practices ("learning mechanics") into concrete game mechanics. The authors call "Serious Game Mechanics (SGMs)" the identifiable abstract patterns that can be replicated across SGs. LM-GM features a clear graphical representation of the game flow and a predefined list of elements to support the analysis. A limitation of LM-GM is that it does not expose the connection between concrete mechanics and the high-level educational objectives that the game is supposed to attain.

Still with the objective of supporting SG design, some authors compiled libraries of commonly reoccurring patterns (Kiili, 2010; Games Enhanced Learning, 2010; Marne et al., 2012). However, these libraries neither offer the classification of individual components nor take into account the relationship between them.

In summary, after examining the works mentioned above, the conclusion is that they provide interpretations of the possibilities and limitations offered by SGs and explain, at a high-level, why games are motivating tools for learning. However, they do not fully answer the question of how the concrete components of the game have to be structured to support the high-level learning and entertainment goals. This question is addressed in Chapter 3.

2.2 Assessment, feedback and adaptivity in SGs

2.2.1 Learning assessment in SGs

Assessment in the education field, in a broad sense, is understood as a measurement of how much a formal educational program or institution has an impact on its students'

learning (Terenzini, 1989). However, to understand the real nature of the assessment it is necessary to consider who is being assessed (an individual or a group), what is being assessed (knowledge, skills, attitudes and values, or behaviors), and with what purpose (enhancing teaching or accountability) (Terenzini, 1989).

As any empirical measurement, assessment in learning has two important properties: reliability and validity. Reliability refers to the amount of error in a test, or the extent that the assessment yields the same results on repeated trials; validity concerns the extent that the assessment measures what it is intended to measure (Carmines & Zeller, 1979). There is no shortage of policies, practices and theories trying to help educators in achieving reliability and validity in assessment (Gardner, 2012b).

Educational researchers typically distinguish between two main types of assessment, depending on their timing and main purpose: formative and summative assessment (Gardner, 2012b). Summative assessment refers to assessments at the end of units or the end of a course with the purpose of attributing grades, issuing certifications, evaluating progress or assessing the effectiveness of a curriculum; formative assessment, conversely, refers to evaluation of a student's learning throughout the learning process as a tool in teaching (Harlen & James, 1997; Wiliam & Black, 1996). These two concepts are discussed in more detail in the context of SGs below.

Summative assessment in SGs

To support the adoption of SGs in education, there must be a manner of objectively establishing a student's progress within the context of the given objectives of the game (Loh, 2012; Bellotti, Kapralos, Lee, Moreno-Ger, & Berta, 2013). For this reason, performing summative assessment in SGs occupies an important role in research on game-based learning (GBL). It gives us tools to review games, evaluate their effectiveness and eventually improve them, particularly when aggregating results from groups of students to extract an assessment of the SG itself (and not of the learner).

Summative assessment of SGs involves evaluating, at the end of one or more gaming sessions, how the game supported the player in achieving the game's educational goals. One typical way of doing this is establishing the students' knowledge on a certain topic with the use of external assessment tools (tests, questionnaires, debriefing interviews, knowledge maps, causal diagrams or other types of teacher assessment) (Ifenthaler, Eseryel, & Ge, 2012; Bellotti et al., 2013). These tools can be applied as pre- and post-tests before and after playing the SG, to identify how the student progressed in his or her understanding of the topic (Bellotti et al., 2013). In some cases, the performance of students instructed with the help of a SG is compared with an alternative instructional technique (e.g. traditional expository class, instructional video, reading a text) to try to establish which method gives better results. Such setup is not without problems, particularly because of difficulties in achieving a truly randomized setting; furthermore, controlling for the influence of the pre-test itself in the post-test results is practically impossible (Loh, 2012; Bellotti et al., 2013).

Some researchers have argued for the use of in-game indicators for making inferences about what the student can do at any point in the game (Ifenthaler et al., 2012; Shute & Ke, 2012; Loh, 2012). However, particularly when using these indicators for summative purposes, there is the issue of the validity and generalizability of the assessment (Mislevy, Behrens, Dicerbo, Frezzo, & West, 2012): how can we know if the student learned anything else than to succeed in the game? In GBL, SG designers need to be aware that their decisions have a great impact on the reliability and validity of the assessment that can be extracted from gameplay (Mislevy et al., 2012).

Formative assessment and feedback in SGs

As mentioned previously, the main difference between formative and summative assessment is their main purpose. Formative assessment is a tool to *support learning*, as opposed to simply being an assessment *of learning* (Gardner, 2012a). It can provide the instructor with information that can help plan the next steps, to achieve the best results. Moreover, it has a role in shaping and improving the student's competence as a self-regulated learner, who is able to set goals, define strategies to achieve those goals, manage resources and react to feedback, instead of relying simply on the inefficient process of trial-and-error learning (Sadler, 1989; Nicol & Macfarlane-Dick, 2006).

A crucial component of formative assessment is feedback (Wiliam & Black, 1996). While there is compelling evidence on the positive effects of formative feedback in learning in general (Leahy & Wiliam, 2012), it is widely accepted that the quality of the feedback is critical in producing those effects (Sadler, 1998; Shute, 2008). Good feedback is not only technically accurate, contextualized and timely (close to the act of learning production), but it also presents explicitly what is expected of the student; it facilitates self-reflection; it is motivating; and it provides opportunities to close the gap between current and desired performance (Sadler, 1998; Yorke, 2001; Nicol & Macfarlane-Dick, 2006). Also, students should be trained in interpreting and using the feedback to improve their work (Sadler, 1998). At the department of Industrial Design in the Eindhoven University of Technology, formative assessment is actively practiced with extensive feedback and explicit self-reflection, which has been observed as being effective (Hummels, Vinke, Frens, & Hu, 2011).

Embedding formative assessment in SGs is not straightforward. Simply incorporating traditional assessment (for example questions and answers) in games is highly disturbing to players' engagement (Prensky, 2001; Bente & Breuer, 2009). To avoid interrupting immersion and flow, and to reduce test anxiety, researchers have argued in favor of stealth assessment, that is, a type of assessment that is made invisible in the learning environment (Shute, Ventura, Bauer, & Zapata-Rivera, 2009; Shute & Ke, 2012; Shute & Y. J. Kim, 2013). This is made possible by the application of what is called evidence-centered assessment design (ECD), in which a learner's performance data is continuously collected from the game, and machine-based reasoning techniques are used to make inferences about the learner's competences across a network of competences and skills (Mislevy, Steinberg, & Almond, 2003; Shute et al., 2009; Shute & Y. J. Kim, 2013), re-

sulting in a learner model. It is argued that stealth assessment and ECD are able to assess not only content-specific learning but also general abilities, dispositions and beliefs, thus being more adequate to assess the so-called 21st century competences (problem-solving skills, persistence, creativity, etc.) than traditional assessment methods (Shute & Ke, 2012).

One approach that has been employed to implement in-game assessment using ECD in SGs is the use of the Competence-based Knowledge Space Theory (CbKST). CbKST comes from the non-numerical and non-linear approach of the Knowledge Space Theory (KST) (Kickmeier-Rust & Albert, 2012a). Its main idea is to assume a finite set of roughly atomic competences – that is, well-defined, small scale descriptions of aptitude, ability, knowledge, or skill – and a prerequisite relation between those competences, which defines the competence model of the domain. The structural model of the theory focuses on unobservable competences, making hypotheses about the brain's black box. By utilizing interpretation and representation functions, these unobservable competences (or, in other words, what is "in the brain") are mapped to evidence or indicators, relevant for a given domain. Such indicators can be all sorts of performance or behavior, and not only test items. The interpretation function (p, in Figure 2.1) assigns a set of competences required to solve a task to each of the indicators. Conversely, by utilizing a representation function (q), a set of indicators is assigned to each competence state. This assignment induces a performance structure, which is the collection of all possible performance states. Due to these functions, unobservable competences and observable performance can be linked in a broad form where no one-to-one correspondence is required. This means that an entire series of indicators can be linked to underlying competence states.

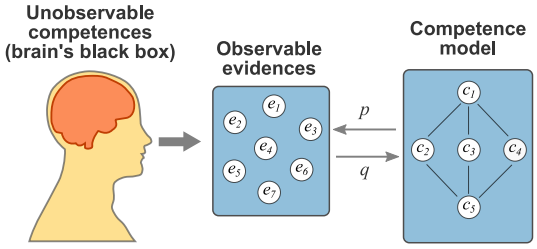


Figure 2.1: The CbKST model makes inferences about the brain's competence state using observable evidence (Kickmeier-Rust & Albert, 2012a).

As is the case with assessment, incorporating feedback in SGs also needs to be done in a way that does not disrupt gameplay or prevent flow and engagement (Dunwell & Freitas, 2011). Designing feedback in SGs implies deciding not only the content and the timing, but also the format and the type that is adequate to the learning requirements, the learner's characteristics and the learning context (Dunwell & Freitas, 2011).

Another way of providing feedback to learners in SGs is the use of open learner models (OLMs), that is, making the learner model available to the learner himself or herself, to

support and encourage metacognition and self-reflection on learning (Bull & Kay, 2007; Bull & Kay, 2013; Bull et al., 2013). In this case, choosing the format of presentation of the model to the learner is important to enable the learner to understand and use the information in his or her self-reflection on learning (Law, Grundy, Cain, & Vasa, 2015).

In addition to explicit formative feedback as described above, SGs can also implement a type of implicit feedback, in which the game world adapts to the learner's actions. Adaptivity as a response to in-game assessment is further discussed at the end of this chapter.

2.2.2 Engagement assessment in SGs

SGs exist in a balance between learning and entertainment objectives. Consequently, learning is not the only relevant aspect in SG assessment. The extent to which a game can engage and motivate the learner is also important, especially because engagement is positively associated with learning outcomes (Carini, Kuh, & S. P. Klein, 2006).

Questionnaires have been developed to assess flow, engagement, motivation and social presence in games (Sweetser & Wyeth, 2005; W. IJsselsteijn, De Kort, Poels, Jurgelionis, & Bellotti, 2007; Jennett et al., 2008; Brockmyer et al., 2009; Wiebe, Lamb, Hardy, & Sharek, 2014) and in SGs (Fu, Su, & Yu, 2009). These questionnaires are not appropriate to track variations throughout the game, as asking questions during the game would likely disrupt the player; furthermore, these questionnaires rely on the player's recollections of the experience and, as such, are limited in their ability to pinpoint particular responses (Kivikangas, Nacke, & Ravaja, 2011).

To overcome the limitations of self-reports to assess engagement in games and SGs, researchers started to measure physiological signals such as heart rate, muscle activity, electrodermal activity and measurements of electrical activity along the scalp (or electroencephalography (EEG)) to try to assess players' gaming experiences and affective state (e.g. engagement, frustration, boredom) (Mandryk, Inkpen, & Calvert, 2006; Kivikangas et al., 2011). Research on the topic is still ongoing, but some researchers have expressed their hopes that real-time monitoring of players' affective state could soon become a common feature in computing systems, including games and SGs (Berta, Bellotti, De Gloria, Pranantha, & Schatten, 2013). This type of assessment could be used to enable games and SGs to respond accordingly, in an attempt to optimize experience (Tijs, Brokken, & W. A. IJsselsteijn, 2008) and learning (Sabourin & Lester, 2014; Bosch et al., 2015; Shute et al., 2015).

2.2.3 Adaptivity

Adaptivity in general refers to the ability or tendency to change one's own characteristics to fit some purpose or situation. In computer science, it refers to interactive systems that are able to adapt their behavior based on information about the user or the environment

(Ahn et al., 2014).

It has been shown that average students tutored one-to-one, with techniques that ensure mastery of one topic before moving to the next, perform much better than average students instructed with conventional methods (Bloom, 1984). In technology-enhanced education (TEL) research, the search for methods to automatically adapt instruction to the individual needs of the student was a major driver for research on computer-assisted instruction (CAI) and intelligent tutoring systems (ITS), in an effort that continues to this date (Haynes, Underwood, Pokorny, & Spinrad, 2014). The objective is to try to make group instruction (e.g. traditional classrooms) achieve the same effects as individual tutoring.

In a similar fashion, delivering the right amount of challenge according to each user's skills is crucial to creating enjoyment in games, so that the game is neither too easy (causing boredom) nor too hard (causing frustration) (Koster, 2005b; Schell, 2008). In other words, being able to adapt to the player is a necessary condition for the success of the game. Techniques to create balanced games vary from static ones that have been used in video games for decades (e.g. increasing difficulty with each success, allowing the player to select the desired difficulty at the start or during the game or provide layers of challenges (Schell, 2008)) to new artificial intelligence (AI) techniques that allow the game to learn during a gaming session and adapt to the player's changing tactics (Spronck, Ponsen, Sprinkhuizen-Kuyper, & Postma, 2006).

Since SGs incorporate both educational and entertainment features, adaptivity in the context of SGs can refer to adaptive behavior not only to the user's game performance (as in entertainment games), but also to the user's learning progress (Steiner, Kickmeier-Rust, Mattheiss, Göbel, & Albert, 2012).

(Kickmeier-Rust et al., 2007) propose the distinction between macroadaptivity and microadaptivity in educational games. The authors compare adaptivity at a macro level with existing approaches in adaptive e-Learning and adaptive hypermedia (Brusilovsky, Peylo, et al., 2003), in which the system must select suitable learning objects to present to the learner at any given time. In educational games, this level of adaptivity would be comparable to providing branched storylines and personalized storytelling. Microadaptivity, in its turn, refers to adaptive presentation and problem-solving support within a learning object (Kickmeier-Rust et al., 2007; Kickmeier-Rust & Albert, 2007).

Adaptivity, particularly to educational aspects, seems to be an important factor in the effectiveness of SGs (Kickmeier-Rust, Marte, Linek, Lalonde, & Albert, 2008; Kickmeier-Rust & Albert, 2012b). Kickmeier-Rust and Albert (2012b) performed a (yet unpublished) meta-review of more than 300 scientific articles on the educational efficacy of computer games; the review indicated that the vast majority (90%) of the games that reported non-trivial educational results displayed some form of educational adaptation or personalization.

Typically, a game or SG already has at least one type of player assessment that can be suc-

cessfully used to perform in-game adaptations: the score (Oostendorp, Spek, & Linssen, 2014). However, research on stealth assessment can arguably provide even better models of player's performance, thus enabling more accurate and useful adaptations in SGs (Adcock & Eck, 2012; Ifenthaler et al., 2012).

2.3 Activity Theory

To help us relate the structural elements of a SG – including assessment, feedback and adaptivity – to its learning and entertainment objectives, we resorted to the theoretical framework provided by activity theory.

Activity theory is the line of research initiated in the 1920s and 1930s by a group of Russian psychologists, notably Vygotsky and Leont'ev (Engeström, 1987). It studies different forms of human practices and development processes, providing a model of human activities in their social and organizational context (Hasan, 1999). Despite the popularity of activity theory in the fields of learning and instructional design, only a few studies apply the most prominent elements of the theory to the study of games and SGs directly (Marsh, 2006; Zaphiris, Wilson, & Ang, 2010; Peachey, 2010; Islas Sedano, 2012). Related concepts, such as that of Zone of Proximal Development, by Vygotsky, are more commonly applied in SG studies, often combined with the Flow theory (Csikszentmihalyi, 1990).

In activity theory, the basic unit of analysis of all human endeavors is activity: a purposeful interaction between subject and object, in a process in which mutual transformations are accomplished. This interaction is usually mediated by physical tools (knives, hammers, computers) or mental tools (notations, maps), which shape the way humans interact with the world (Kaptelinin & Nardi, 2006).

Engeström (1987) extended the original model of activity proposed by Leont'ev (1978), describing the activity as a collective phenomenon. The model, called Activity System, is depicted as a triangle (Figure 2.2) in which the sides represent the main components of the system (subject–object–community) and the corners represent the mediation artifacts to those relationships (tools–social rules–labor division). The activity is directed at the object and results in an outcome. Years later, Engeström (2001) extended the model to represent multiple perspectives and dialogs between several interacting systems, and how the interaction between the objects of different activities can result in a shared, jointly constructed new object (Figure 2.3). This second model is called Activity System Network (Engeström, 2001; Guy, 2005).

According to activity theory, an activity happens simultaneously at three levels, in a hierarchical structure (Figure 2.4) (Leont'ev, 1978). At the topmost level, the *activity* is directed at a *motive*; in other words, the motive is the object that the subject ultimately wants or needs to attain. Typically, the activity is realized by a sequence of *actions*, each of which may not be directly related to the motive (Kaptelinin & Nardi, 2006;

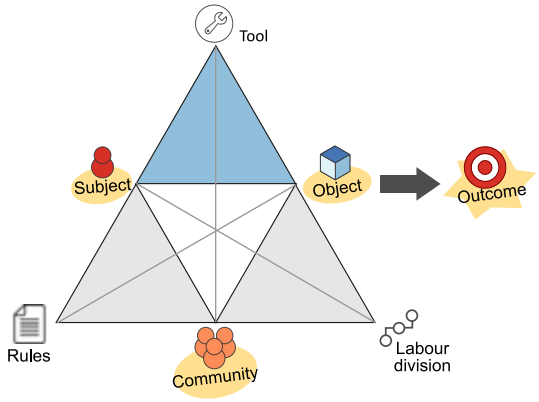


Figure 2.2: The Activity System (Engeström, 1987)

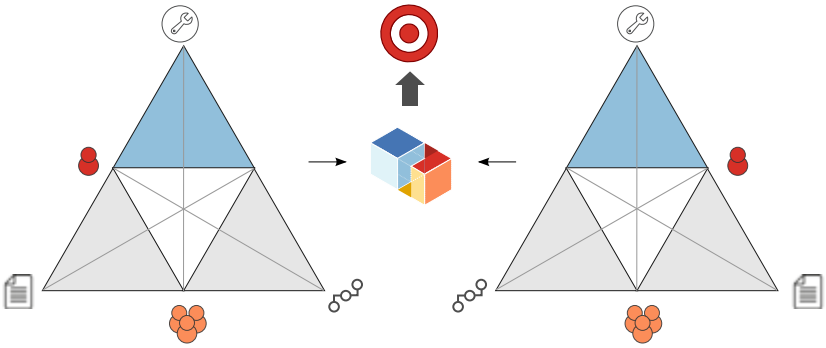


Figure 2.3: The Activity System Network (Engeström, 2001)

Devane & Squire, 2012). Each action is also directed at an object: the *goal*. Subjects are typically aware of their goals, but maybe not consciously aware of their motives. On its turn, an action is also composed of lower-level units, called *operations*, which are performed unconsciously, according to given *conditions*. Understanding the breakdown of activities in different levels can provide a valuable tool for modeling learning, training and decision-making, particularly in the context of human interaction with complex systems (Rauterberg & Felix, 1996).

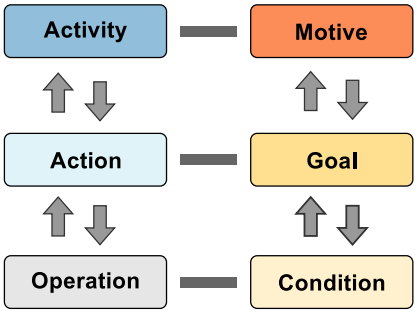


Figure 2.4: The hierarchical structure of activity, or levels of activity, and their transformations, as defined in activity theory (Kuutti, 1995)

The activity is not a static entity. Constant transformations happen between the levels, according to changes in the environment or the subject's motivations or skills (Kuutti, 1995; Peachey, 2010). Furthermore, it is possible to realize the same activity by different sets of actions and operations, and the same actions can be part of different activities simultaneously (Hasan, 1999).

2.4 Chapter summary

From the analysis of the related work, it was identified that existing models, methodologies and frameworks for SGs analysis and design are either focused only on high-level aspects of SGs, or they offer a way to investigate the inner components of the game but without providing a clear connection between the concrete mechanics and the high-level objectives of the game. There is a gap between the theoretical understanding of what constitutes an effective SG and the actual, practical way in which this is achieved at a low level. In other words, current literature on the topic does not provide a satisfactory answer to RQ1.

Formative assessment, feedback, and adaptivity are three important and closely related topics in educational research, including GBL, due to their crucial role in the educational effectiveness of SGs. As such, they deserve special consideration in this work. Formative assessment of the learner, associated with the provision of quality feedback, is seen as one of the most effective tools to support learning. Current developments point to

the use of stealth assessment and implicit feedback in the form of adaptivity and/or personalization to improve the effectiveness of SGs. These concepts occupy an important place as functional requirements in the development of the software architecture of SGs, which is presented in Chapter 5.

To be able to reconcile educational theory – including the role of assessment, feedback and adaptivity in learning – and game development, I have used activity theory as the underlying theoretical framework of the Activity Theory-based Model for Serious Games (ATMSG) (Chapter 3). Activity theory provides a hierarchical and comprehensive understanding of human practices and development processes within the social context. Concepts of activity theory have been used successfully before to support the design of educational games (Paraskeva, Mysirlaki, & Papagianni, 2010; Marsh, 2010), serving as evidence of their applicability in the SG domain.

Chapter 3

Activity Theory-based Model for Serious Games (ATMSG)

This chapter presents a model for serious game (SG) analysis and design called Activity Theory-based Model for Serious Games (ATMSG) and a taxonomy of SG elements. The ATMSG is one of the main contributions of this thesis, and my response to RQ1 (Section 1.2).

The ATMSG model connects a game's educational and entertainment high-level objectives with low-level in-game components on one hand, and links individual gaming and pedagogical components as the game unfolds on the other. It can be used mainly for SG analysis, but may also be used as a tool for the conceptual design of SGs. Being applied at early stages of prototyping, the model helps SG designers in assessing if the envisioned game structure is able to support the desired pedagogical goals.

In this work, I address this issue by proposing a modification of the Learning Mechanics–Game Mechanics (LM-GM) model, expanding it to incorporate high-level aspects of the SG into the model, using concepts of activity theory.

The ATMSG model includes a SG components taxonomy, which is based on established taxonomies of learning, of instructional design and of game components. The taxonomy, used in conjunction with the model, supports the analysis of SGs by providing an extensive list of commonly found structures. This list can be referred to when trying to identify the various components that constitute SGs.

This model is targeted at two main user groups. First and foremost, to those involved directly in the study and creation of games for learning, namely SG researchers and

Parts of this chapter were published previously in Carvalho, Bellotti, Berta, Gloria, Sedano, et al. (2015).

SG designers, who possess a high level of knowledge on the topic. The second group consists of non-expert users involved in SG design or evaluation projects, and who find themselves in need of understanding SGs in more depth, e.g. entertainment game designers, educators, or people with knowledge on the topic addressed in the project (domain experts).

3.1 Elaborating the model

To elaborate the ATMSG and the taxonomy of SG components, we performed an iterative process that alternated between literature review and practical testing of the concepts to identify points for refinement.

The survey on methods, methodologies, and frameworks for game and SG design showed the missing links and gaps that should be addressed, as discussed in Section 2.1.

We used activity theory to understand the context of use of educational SGs, by identifying the relevant network of activities (as proposed by Engeström (2001), see Section 2.3). We investigated five games of different genres and different learning domains (Darfur is Dying, DragonBox Algebra 5+, GoVenture CEO, IBM City One and Playing History: The Plague). From this step, we derived the ATMSG model, presented in Section 3.2.

A new literature search was performed to find reference frames to help users in identifying game components according to activity theory. Since we could not find a unified taxonomy with the format we needed, we combined existing taxonomies of games, of learning objectives, and of instructional design theories into a new structure. The result of this step is described in Section 3.3.

Subsequently, we produced a first set of guidelines on how to apply the ATMSG model and the taxonomy for analyzing SGs. In a user-centered design approach, these guidelines have been iteratively improved during its elaboration, through a set of user tests, described in Section 3.5. The objective was to assess both the usability and the functionality of the model, particularly its capability to support the evaluation of the educational quality and effectiveness of SGs. We identified weak and strong points of ATMSG and used the results to simplify the model, resulting in the version described in Section 3.2.

3.2 The model

In this section, our conceptual model derived from an activity-theoretical view of educational SGs, called ATMSG, is described. This model utilizes the conceptual framework of activity theory to understand the structure of educational SGs, providing a way to reason about the relationships between SGs components and the educational goals of the game.

In the ATMSG model, we do not consider the game as an isolated artifact. Instead, we propose that the SG and those using it to learn or to teach something are seen as part of a complex, dynamic system. From this perspective, educational SGs are typically used in the context of at least three activities: the *gaming activity*, the *learning activity* and the *instructional activity* (Figure 3.1).

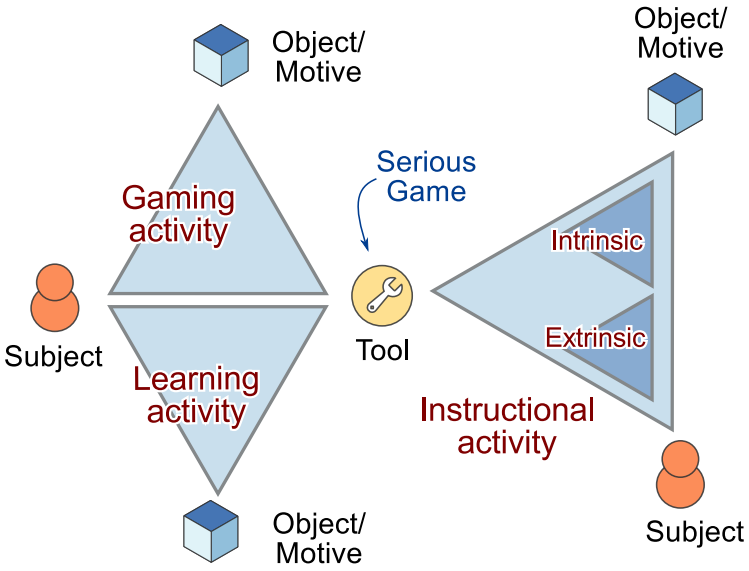


Figure 3.1: The ATMSG model, with its three main activities

There are three main activities involved in the use of SGs for education. This figure represents the highest level of the activity system involved.

Figure 3.1 depicts the three main activities and the relationships between people and artifacts in this system. It is possible to see that the gaming and the learning activities share the same subject (the player/learner) and tool (the SG), but they have different driving motives. For example, the motive driving the gaming activity might be simply to have fun, while the motive driving the learning activity might be to fulfill a course requirement. The instructional activity also shares the same tool but has a different subject (the instructor and/or the game designer) and motive. A motive for the instructor might be, for example, to use the SG to raise the learner's interest in the topic.

The difference between the learning and the instructional activities is important: while the learning activity corresponds to the point of view of the learner, the instructional activity depicts the side of the instructor(s). Acknowledging this distinction allows us to identify possible conflicts in motives driving the activities which might affect the learning outcomes of the game. It can also help in evaluating to which extent the instructional components of the SG support the stated learning outcomes.

Some educational SGs can be used by a learner on his or her own while other games may also count on complementary activities led by an instructor. The ATMSG model explicitly accounts for this distinction, to clarify the role of the teacher/instructor in the game. For this reason, the instructional activity is divided into two activities: *intrinsic instruction*, and *extrinsic instruction*. The intrinsic instructional activity takes place solely inside the game. It involves how the game itself supports learning (e.g. via tips, help messages, automatic assessments, in-game adaptive features). The extrinsic instructional activity, conversely, is performed outside by the teacher/instructor before, during or after the playing session, in the context of the overall learning setting (e.g. class, workshop, course).

In the intrinsic instructional activity, the subject is the game designer or producer, who "acts" in the SG through design decisions made when creating the game, or via in-game assessment and feedback mechanisms. An analysis of this activity can be performed without necessarily considering a specific context of use. An analysis of the extrinsic instruction of a game, conversely, is heavily dependent on how the instructor uses the game. Consequently, such analysis cannot be carried out without explicit reference to a concrete usage setting.

The hierarchical structure of the activity, as defined in activity theory, gives us the ability to change the focus of the analysis to different levels of detail. This approach has been used previously by the Hierarchical Activity-Based Scenario (HABS) framework in the study of games and SGs (Marsh, 2006; Marsh, 2010; Marsh & Nardi, 2014), providing a useful and flexible tool to analyze and design interaction and gameplay. It provides a way to reason about the player/learner's engagement, by looking at how much the motives that drive all three activities match each other or not, and how much these motives coincide with the goals of the actions (Marsh & Nardi, 2014). In ATMSG, we expand this hierarchical analysis: we also divide the activities into actions, and we divide the game itself into its smaller pieces. Specifically, each activity is broken down into a sequence of actions mediated by tools with specific goals. Similarly to the activity, actions can also be depicted as triangles, as shown in Figure 3.2.

We call the items that form these smaller triangles "SG components". SG components are the concrete pieces of a SG, e.g. characters, tokens, tips, help messages. They can be classified as gaming, learning or instructional components, according to the activity they support. The triangles can be represented in an alternative way: flattened, as nine (or twelve, if considering the extrinsic instruction activity) layers of components that interact over time during gameplay (see Figure 3.5 for an example of these layers). This representation shows the relationships between the components of the different activities over time. There can be overlaps, as one component can support actions from any of the activities simultaneously. For example, a puzzle-type challenge can play a role as a game component, as a learning tool and as an instructional tool at the same time.

Actions can also be broken down into their constituent operations. At this level, a SG is seen as a combination of its low-level components (e.g. buttons, graphics, sounds, menus), which mediate operations performed unconsciously by the subject (e.g. reading

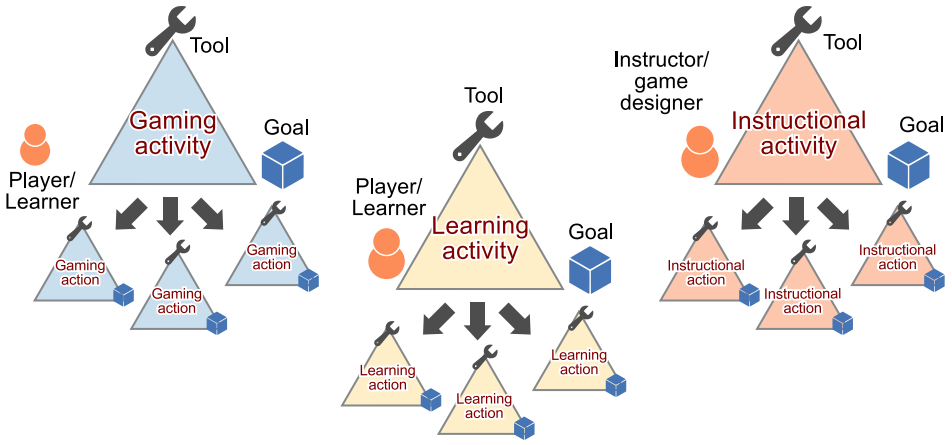


Figure 3.2: Each activity is formed by a sequence of actions. These actions mirror the triangle of activity: they are also mediated by tools, with specific goals

a text, clicking a button). The ATMSG model does not explicitly consider this level of analysis since there are no significant differences on how digital SGs and other software are constituted at this level. Hence, existing frameworks for applying activity theory in human-computer interaction (HCI) research (Kuutti, 1995) and usability studies (Kaptelinin, 1996) can also be used to analyze SGs at this level of detail.

3.3 Taxonomy

We used the ATMSG model to reorganize existing taxonomies of learning, instruction, games and SGs into a unified vocabulary that can be easily consulted when needed. It aims to aid in the identification and classification of components according to their characteristics and roles in the game.

The taxonomy is organized in a tree structure in which items are classified according to the activity to which they belong. Within the activity, they are classified as actions, tools or goals. For easier referencing, the elements are further divided into categories. Details of this classification are given below.

3.3.1 Gaming components

The list of gaming components incorporates terms described by previous works on game mechanics (Adams & Dormans, 2012; Djaouti et al., 2007; Koster, 2011; Schell, 2008; Zagal et al., 2005), in addition to the game mechanics identified in the LM–GM model (Arnab et al., 2015) and the game components of the Game Object Model II (GOM II)

model (Amory, 2007).

There are a number of different definitions of "game mechanics" (Sicart, 2008). To avoid confusion with existing terminology and inconsistencies in the definitions, we do not use this term. Instead, we classify gaming components according to the three layers of the gaming activity, i.e. actions, tools, and goals. These are roughly equivalent to what has typically been defined as game mechanics by game researchers and designers.

The components classified as *gaming actions* (Table 3.1) describe, from the player's point of view, the actions that can be performed in the game at any given point. They have been grouped in categories that express similar types of player's interaction with the game.

Gaming actions	
Category	**Elements**
Entity Manipulation	Capture, Collect, Create, Customize, Design, Destroy, Edit, Eliminate, Exchange, Generate, Manage resources, Manipulate gravity (physics), Match, Own, Plan / Strategy, Remove, Select, Tactical maneuver, Trade virtual items
Movement	Avoid, Collide, Move, Evade, Rotate, Shoot, Target, Teleport, Traverse, Visit
Time-related	Manipulate time, Start/ Stop time, Advance game period
Information	Ask questions, Answer questions / trivia, Obtain help, See performance evaluation, Watch / Listen to / Read information, Watch / Listen to / Read story

Table 3.1: Gaming actions

While gaming actions describe what a player does, *gaming tools* (Table 3.2) are the components that make actions possible, i.e. the components that the player manipulates or with which he or she interacts. They can also be the rules or characteristics of gameplay that define how actions can be taken in the game.

Gaming goals (Table 3.3) describe, in general terms, the types of goals and sub-goals typically found in games. Gaming goals complete the activity system triangle (see Section 2.3) at this level of analysis: every gaming action is performed using one or more gaming tools, to achieve at least one gaming goal.

3.3.2 Learning components

The list of learning components is mostly based on Bloom's Updated Taxonomy (Anderson, Krathwohl, & Bloom, 2001), which is arguably the most commonly used framework to describe learning goals. Two other taxonomies are used to complement Bloom's Updated Taxonomy: Kolb's Experiential Learning Cycle (Kolb, 1984) and Fink's Tax-

Gaming tools	
Category	**Elements**
Objects	2D/3D space, Avatars, Cards, Gifts, Goods, Grids, Information, Modifiers, Non-playing characters (NPCs), Tiles, Tokens, Virtual money
Attributes	Lives, Position in space, Roles, Secrets, Virtual skills
Time	Chronometer, Time pressure
Feedback	Achievements, Leaderboards, Penalties, Performance meters, Performance record, Points, Progress bars, Rewards, Status levels
Help	Advice and assistance, Guide character, Checklists/ Task lists, Tips, Tutorial, Warning messages
Chance / Randomness	Dice, Lottery, Random appearances, Randomizers
Narrative (aesthetics)	Cut scenes, Role play, Story (text)
Rules	Complete information, Incomplete information, Competition, Game modes, Game master / referee, Multiplayer, Zero-sum / Non-zero-sum
Segmentation of gameplay	Alternating turns, Challenges, Checkpoints, Game Period, Infinite gameplay, Levels, Meta-game, Puzzles, Quest / Problem, Time
Goal metrics	Achievement, Performance record, Score, Success level, Time
Score	Video Game Score, Cash Score, Social Network Score, Composite Metrics, Experience Points, Redeemable Points, Karma Points

Table 3.2: Gaming tools

Gaming goals	
Category	**Elements**
Score	Maximize performance, Maximize score
Tasks	Collect resources, Collect information, Solve puzzle
Narrative	Complete quest, Complete side quests, Form/discover goal, Get acquainted with story, Reach narrative end
Competition	Be the first to reach the end, Be the last player standing
Other goals	Configure game, Learn to use interface, Perform task within allotted time, Reach resources end

Table 3.3: Gaming tools

onomy of Significant Learning (Fink, 2003). Kolb's Cycle incorporates a constructivist perspective while Fink's Taxonomy includes learning goals that are less curricular and more focused on transferable skills such as critical thinking, creativity, problem-solving, among others.

Learning actions (Table 3.4) are the actions that the player/learner performs in the game while *learning tools* (Table 3.5) are the in-game artifacts that support one or more actions. To generate the list of actions and tools, we combined the original learning mechanics of the LM-GM with a list of illustrative action verbs based on Bloom's Updated Taxonomy (Almerico & Baker, 2004; Illinois Central College, 2011).

The *learning goals* list (Table 3.6) is a direct reproduction of Bloom's Updated Taxonomy (cognitive, affective and psychomotor domains) (Anderson et al., 2001), Kolb's Experiential Learning Cycle (Kolb, 1984), and Fink's Taxonomy (Fink, 2003).

3.3.3 Instructional components

The instructional activity has a different subject than the other two activities: the person(s) teaching something using the SG. There is a conceptual overlap between the instructional activity and the learning activity, as they are complementary ways of analyzing the same process. The instructional activity depicts how instructors and/or game designers act to facilitate the learning process, particularly by providing adequate conditions for it to occur.

The taxonomy does not distinguish between intrinsic and extrinsic instructional components since the distinction between the two depends solely on where the components are used: if inside the game, they correspond to intrinsic instruction; if outside of it, they are related to extrinsic instruction.

Just as in the case of learning actions and tools, *instructional actions* (Table 3.7) are the actions that the game and/or the instructor perform during the game with the objective

Learning actions	
Category	**Elements**
Remembering	Define, Describe, Draw, Find, Identify, Imitate, Label, List, Locate, Match, Memorize, Name, Observe, Read, Recall, Recite, Recognize, Relate, Reproduce, Select, State, Write, Tell
Understanding	Compare, Convert, Demonstrate, Describe, Discuss, Distinguish, Explain, Explore, Find more information about, Generalize, Interpret, Objectify, Outline, Paraphrase, Predict, Put into own words, Relate, Restate, Summarize, Translate, Visualize
Applying	Apply, Calculate, Change, Choose, Classify, Complete goal, Complete, Construct, Examine, Experiment, Illustrate, Interpret, Make, Manipulate, Modify, Perform action/task, Produce, Put into practice, Put together, Show, Solve, Translate, Use
Analyzing	Advertise, Analyze, Categorize, Compare, Contrast, Deduce, Differentiate, Discover, Distinguish, Examine, Explain, Identify, Investigate, Separate, Subdivide, Take apart
Evaluating	Argue, Assess, Choose, Critique, Debate, Decide, Defend, Determine, Discuss, Estimate, Evaluate, Judge, Justify, Prioritize, Rate, Recommend, Review, Select, Value, Verify, Weigh
Creating	Add to, Build model, Combine, Compose, Construct, Create, Design, Devise, Forecast, Form goal, Formulate, Hypothesize, Imagine, Invent, Originate, Plan, Predict, Propose

Table 3.4: Learning actions, based on Almerico and Baker (2004) and Illinois Central College (2011)

Learning tools	
Category	**Elements**
Dramatizing	Dramas, Dramatizations
Graphical information	Art, Cartoons, Diagrams, Displays, Graphed information, Graphics, Graphs, Illustrations
Interaction	Court trials, Debates, Demonstrations, Experiments, Group discussions, Questionnaires, Simulator, Speculations, Surveys, Tests
Multimedia	Animation, Films, Media presentations, Recordings, Songs, Speech, Television programs, Videos
Problem-solving	Problems, Puzzles
Textual information	Analogies, Arguments, Bulletin boards, Classifications, Conclusions, Definitions, Editorials, Forecasts, Information, Magazine articles, Models, Newspapers, Organizations, Outlines, Poems, Posters, Recommendations, Reports, Routines, Rules, Standards, Story, Student diary, Summaries, Task list/checklist, Tasks, Textbooks, Texts, Tips
Other	Challenge, Creations, Events, Inventions, Sculptures, Self-evaluations, Systems, Values

Table 3.5: Learning tools, based on Almerico and Baker (2004) and Illinois Central College (2011)

Learning goals	
Category	**Elements**
Bloom's Taxonomy – Cognitive domain	Remembering, Understanding, Analyzing, Applying, Evaluating, Creating
Bloom's Taxonomy – Affective domain	Receiving phenomena, Responding to phenomena, Valuing, Organization, Internalizing values
Bloom's Taxonomy – Psychomotor domain	Perception (awareness), Set, Guided response, Mechanism (basic proficiency), Complex overt response, Adaptation, Origination
Kolb's experiential learning cycle	Concrete experience, Active experimentation, Reflective observation, Abstract conceptualization
Fink's Taxonomy	Foundational knowledge, Application, Integration, Human dimension, Caring, Learning how to learn

Table 3.6: Learning goals

of stimulating learning actions and facilitating learning goals.

Instructional actions	
Category	**Elements**
Assessment	Qualitatively assess performance, Quantitatively assess performance
Feedback	Reward good performance, Sanction bad performance, Suggest improvements, Support recovery from errors
Information presentation	Demonstrate, Present material, Present problem, Present quiz, Repetition, Review lesson, Scaffold, Show similar problems, Stress importance, Tell story

Table 3.7: Instructional actions

Instructional tools (Table 3.8) are components present in the game that support instructional actions, providing help and feedback to learners and assessing their performance. There may be overlaps between learning tools and instructional tools.

Instructional tools	
Category	**Elements**
Behavior elicitation	Challenge, Deadlines, Limited set of choices, Multiple chances
Feedback	Penalties, Performance measures, Rewards
Information	Checklists, Help text, Story, Tips / assistance, Warning messages
Interaction	Discussion, Questions & answers
Practice	Practice tests, Simulators

Table 3.8: Instructional tools

Each instructional action has one or more *instructional goals* (Table 3.9). Two theories widely employed in instructional design were used as a reference to identify the goals of instructional actions: Gagné's Nine Events of Instruction (Gagné, 1985) and Keller's ARCS Model of Motivational Design (Keller, 1987). The events of instruction are external events that the instructor can elicit, in sequence, to provide an adequate environment for effective learning. The ARCS model, on the other hand, lists four steps that can promote and sustain motivation during the learning process.

3.4 Application of the model

In this section, we propose a four-step approach that progressively guides the user in applying the ATMSG model for game analysis (Subsection 3.4.1). The same approach can also be used as a tool for SG design, with small modifications that are explained in

Instructional goals	
Category	**Elements**
Gagné's Nine Events of Instruction	Gain attention, Inform learner of objective, Stimulate recall of prior learning, Present the stimulus, Provide learning guidance, Elicit performance, Provide feedback, Assess performance, Enhance retention and transfer
ARCS Model of Motivational Design	Attention, Relevance, Confidence, Satisfaction

Table 3.9: Instructional goals

Subsection 3.4.2. To illustrate the application of the model, we present an example of game analysis in Subsection 3.4.3.

3.4.1 ATMSG for serious game analysis

The ATMSG four-step approach for SGs analysis aims to help the user (i.e. the person analyzing a game) gain a better understanding of how learning takes place in the game. These steps take the user from a high-level understanding of the activities to the concrete components that implement those activities. The user identifies game components with the help of the taxonomy of SG components (described in Section 3.3).

Figure 3.3 outlines the four steps of the approach. Each step is described below.

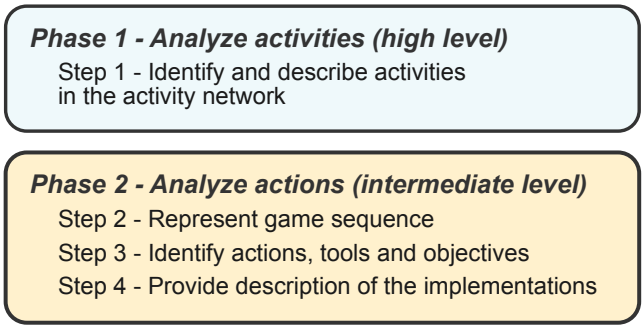


Figure 3.3: The four-step approach for applying the ATMSG to the analysis of educational serious games

Step 1: Describe the activities

In the first step, the user describes the main activities involved in the activity system and identifies their subjects and corresponding motives (Table 3.10). Each description shifts the user's understanding of the game and highlights the main aspects of each activity, encouraging the user to observe the game from different but complementary aspects.

Activity	Subject	Description
Gaming activity	Who is the player?	Why is the subject playing? What are the general objectives of the game?
Learning activity	Who is the learner?	Why is the subject engaging with the game? What are the learning objectives of the game?
Intrinsic instructional activity	Who designed/ produced the game?	Why was the game produced? How is the game trying to convey its learning contents?
Extrinsic instructional activity	Who is using the game to teach something?	Why is the subject using the game? How is the game used to teach something? Are there any other tools used in conjunction with the game to achieve the learning objectives?

Table 3.10: Guiding questions to describe activities

Filling in the fields for the extrinsic instructional activity is not necessary when the analysis is not related to a specific usage context.

It is not always possible to give a precise description of what are the motives driving the activities, as motives are highly personal and variable. In these cases, the motives have to be presumed by the person performing the analysis. Nevertheless, even mere presumptions are valuable, as they can help detect inconsistencies and contradictions between the high-level motives driving the activities and the concrete actions chosen to implement that activity, indicating possible problematic points for the player's engagement with the game.

Step 2: Represent the game sequence

To help in the identification of the components of the SG, the user produces a diagram that represents the game sequence in a rough timeline. The purpose of this diagram is to establish a reference point that helps uncover how the components of the activity system, which will be identified in Step 3, are connected throughout the game. It also facilitates a visual comparison between multiple games, even if they belong to completely different genres. The game sequence visually describes the overall structure of the game, marking points in which choices or evaluations of the game state are made and loops that indicate the repetition of similar arrangements in the game.

The game sequence representation follows the Unified Modeling Language (UML) activity diagrams notation, which uses shapes connected by arrows to represent the flow of the activities (see Figure 3.5). UML was chosen for its status as de facto standard in the software engineering field (C.-H. Kim, Weston, Hodgson, & Lee, 2003).

	Gaming activity	Learning activity	Intrinsic instruction activity	Extrinsic instruction activity
Actions	How does the game unfold? Which actions does the subject perform in the game?	What tasks does the subject do in the game that are directed towards the learning goal?	What happens in the game that supports the learner to achieve the learning goals (assessment, feedback)?	What happens, during the game but outside of it, that supports the learner to achieve the learning goals?
Tools	Which elements are involved/ used in the gaming actions?	Which elements are involved/ used in the learning actions?	Which elements are involved/ used in the game to support the instructional actions?	Which elements are involved/used, outside the game, to support the instructional actions?
Goals	What does the subject have to achieve in the game at this point?	Which knowledge or skills is the learner expected to acquire with the learning actions?	What are the instructional goals of the game at this point?	What are the instructional goals driving the actions described above?

Table 3.11: Guiding questions to identify actions, tools and goals

Step 3: Identify actions, tools and goals

In this step, the user proceeds to identify components related to each node of the game sequence. At this level of the analysis, each event in the game is decomposed into its actions, tools and goals. Together, the components answer, for each step of the game, the question: "what is the subject doing, how, and why".

The user chooses the relevant component directly from the taxonomy of SG components. The graphical representation of these relationships consists of a layered table in which the components are placed, matching vertically the node of the game sequence to which they are related (see Figure 3.5 for an example). For each activity involved (gaming, learning, intrinsic instruction and the optional extrinsic instruction), there are three layers to be filled (actions, tools, and goals), totaling nine (or twelve, if considering the extrinsic instruction) layers.

Table 3.11 presents guiding questions that can help the user when mapping SG components.

Not all nodes of the game sequence will have actions of all the activities happening at

the same time. Similarly, more than one action can happen at the same time within the same activity. Some components of the taxonomy may be relevant to the game as a whole (e.g. certain rules of play, whether the game is a 3D space), and should be indicated at the beginning of the table, before the start of the game sequence representation.

The result of this step is a blueprint of the game structure that reflects the essential form of the SG components (Figure 3.5).

Step 4: Description of the implementation

In this step, the user groups each set of actions, tools and goals that are from the same type of activity and that are related to the same node of the game sequence. For each of those blocks, the user provides a more thorough description of their implementation, explaining what is being done at that point in the game, using which tools, and with which purpose. In this description, the user can complement the description of the component with more specific details of its implementation (e.g. how a score is calculated, or the characteristics of a non-player character) and explain how the usage of such components and characteristics support the achievement of the entertainment and/or pedagogical goals of the game. This step is done separately for each type of activity (Table 3.12).

When combined, the four steps described above provide a comprehensive view of the structure of the game, from its high-level purposes and general characteristics to its concrete implementation.

3.4.2 ATMSG for serious game design

The ATMSG model can be used as a tool to support the SG design process. The model can be applied during the design phase in the iterative analysis of game prototypes – preferably, but not limited to, low-fidelity ones (e.g. sketches, storyboards, game diagrams). In addition, it can serve as inspiration to designers by offering a comprehensive list of SG elements to choose from, according to their educational goals. Furthermore, SG designers can use analyses of other SGs to help them understand characteristics and patterns of other successful (or unsuccessful) games.

When applying the ATMSG for SG design, Step 1 (in Phase 1) of the application guide (Subsection 3.4.1) remains the same, as it focuses on high-level characteristics of the SG that should be defined in the very beginning of the project. The difference lies in Steps 2–4 (in Phase 2), which should be applied in conjunction with prototyping techniques.

Starting from the description of the activities, the designer produces a first version of the game prototype, using his or her preferred method. This prototype is analyzed according to Steps 2–4 described in Subsection 3.4.1. The resulting evaluation provides insights on the level of integration of the gaming and learning components, shedding light on possible weak points in the design. The designer adjusts the prototype accordingly, and

subsequently repeats the steps until a satisfactory structure has been achieved.

3.4.3 Example analysis

This section presents an ATMSG analysis of DragonBox Algebra 5+ (WeWantToKnow, 2012), a critically acclaimed and commercially successful video game for teaching algebra concepts to young children (J. H. Liu, 2012). In the game, the child manipulates colorful cards according to certain rules that are demonstrated with interface tips. The goal of each puzzle is to isolate the DragonBox on one side of the game board. As the game progresses, the pictures in the cards are replaced with numbers and variables: the puzzles are in fact algebraic equations that must be solved for an unknown variable, represented by the DragonBox (Figure 3.4).

Figure 3.4: An early puzzle in the game DragonBox Algebra 5+. Image credit: WeWantToKnow

The analysis considers a child playing on his or her own, outside of any classroom activities and without the help of a parent; thus, it includes only the intrinsic instructional activity.

Table 3.12 shows the descriptions of the activities of the game (Step 1). Note that these descriptions are related to the same event (the child playing the DragonBox game), but from three different standpoints (e.g. the fun aspect for the child; the learning aspect, also for the child; and the game's point of view).

Figure 3.5 represents the game sequence (Step 2) with the related game components depicted in layers (Step 3). The game sequence visually describes the overall structure of the game, which in this case is a repetition of the sequence "Interface tip", "Puzzle" and "Rewards", and constant evaluations about the state of the game ("is this a new skill to the player?", "are there more puzzles in this chapter?"). The game is split into chapters, but the chapters do not differ in their structure – they only mark the progression through the topics. Vertically aligned to the nodes in the game sequence are the layers

Activity	Subject	Description
Gaming	Children aged 5–12	The objective of the game is feed the dragon and watch it grow. To pass each level, the player must solve a series of puzzles, manipulating tiles until the DragonBox is alone in one side of the game board. Graphics, music and rewards follow the same general style of apps and games typically targeted at the same age group, keeping it familiar and fun.
Learning	Children aged 5–12	The puzzles are in fact algebraic equations that must be solved for an unknown variable, represented by the DragonBox. Graphical icons are progressively replaced by numbers and variables. Typically, there is no conscious motivation for the learning activity.
Intrinsic instruction	WeWantToKnow	The game aims to introduce basic concepts of algebra in a fun way. It tries to remove the negativity surrounding the topic by making it as simple as possible to understand.

Table 3.12: Description of activities in DragonBox Algebra 5+

of SG components. Only nodes 3, 4 and 5 contain components in the layers related to the learning and instructional activities while nodes 1 and 2 are related to customization, learning the interface and getting the player involved in the game. This analysis allows us to identify where the core of the learning experience is and which components characterize it (e.g. the tips, the challenges, the rewards, the scaffolding of challenges, the ability to recover from errors).

There is a clear overlap between the motivations driving the gaming and the learning activities, to the point that the learner typically will not be consciously aware of the learning goals (see Table 3.12). For the target audience, this is likely to be a positive characteristic to promote engagement. Also, the designers of Dragonbox tried to make the gaming motives compelling enough to its audience, by using appealing and familiar graphics, music and rewards. Furthermore, the main gaming goals ("solve puzzles", "maximize performance") are directly related to the gaming motive ("feeding the dragon to watch it grow"), indicating that the player will be engaged in the concrete actions performed in the game so that the driving motive is fulfilled.

Table 3.13 provides more details on the implementation of the gaming components (Step 4). For example, in Figure 3.5 it is shown graphically that node 4 ("Puzzle") has the following learning components: the actions "Repetition", "Imitating", "Experimenting", the tool "Challenge" and the goal "Remembering". Conversely, in Table 3.13, in the corresponding cell (i.e. in the "Puzzle" row, "Learning" column), we can read a more thorough description of how these components are connected.

	Gaming						Learning			Intrinsic Instruction		
Actions	Customization	Watch	Obtain help	Matching	See performance evaluation	See performance evaluation						
Tools	Avatar	Information	Tutorial	Tiles; Performance Meter; Warning messages	Rewards; Performance meter	Rewards; Levels						
Goals	Configure game	Learn UI	Learn UI	Solve puzzle; Maximize performance	Maximize performance	Maximize performance						
Actions			Observe	Repetition; Imitating; Experimenting	Verifying/ reviewing							
Tools				Challenge	Report			Tips				
Goals				Remembering	Understanding			Remembering				
Actions									Demonstrating	Scaffolding; Repetition; Recover from errors	Quantitatively assessing performance; Rewarding good performance	
Tools									Tips	Challenge; Multiple Chances; Limited set of choices	Performance measures	
Goals									Providing learning guidance	Eliciting performance, confidence	Providing feedback; Assessing performance	

Figure 3.5: Game sequence and SG components in DragonBox Algebra 5+

Game sequence node	Gaming	Learning	Intrinsic Instruction
1. Choose avatar	The player chooses a character as his or her avatar. The avatar is not used anywhere else in the game.	-	-
2. Introduction	A short animation explains the basic objectives and rules of the game.	-	-
3. Interface tip	If a new skill ("power") is needed to solve the puzzle, the game shows an animation explaining the allowed movements.	The game conveys the rules of algebra by demonstrating the allowed movements.	No verbose explanations are given. Simple tips provide the guidance the player needs to solve the puzzles.
4. Puzzle	The player has to move and combine tiles to isolate the DragonBox in one side, using as few movements as possible. The interface forces the player to follow the rules. The player can play the same puzzle as many times as she wants.	Puzzle after puzzle, the player has to repeat the same patterns until they become automatic. Experimenting with the rules is encouraged, as the interface forces the user to balance the equations correctly.	Puzzle complexity increases very gradually. Skills are accumulated over several levels. The interface prevents the player from making mistakes, which avoids frustration and increases the player's confidence.
5. Rewards	After completing the puzzle, the player earns stars for each possible achievement. Extra points are given when the puzzle is solved in fewer movements and when no extra elements are left in the board.	It is not possible to give wrong answers, but the player can earn extra points for eliminating extra pieces and for using fewer movements. The player can repeat the level to achieve a better score with no penalties.	Assessment of player's performance gives feedback on which rules were not completely followed and elicits the player to try again.
6. End of chapter	A screen showing the full-grown dragon marks the end of the level. A player can share his or her achievements in different social networks.	-	-
7. End screen	When all levels have been completed, the player is invited to play the bonus stages, which feature algebraic equations in proper mathematical notation.	-	-

Table 3.13: Description of the implementation of DragonBox Algebra 5+

3.5 Evaluation

We performed three preliminary evaluation studies of ATMSG in which we investigated the users' perception of the usability and usefulness of the model. The goal was to obtain early user feedback to address issues, particularly on usability, before proceeding with more extensive user testing. The first study evaluated ATMSG on its own while the subsequent ones compared it with the LM-GM model (Arnab et al., 2015). For a short description of the LM-GM model, see Section 2.1.

The data set supporting the results of this evaluation is available in the DANS repository (Carvalho, 2015a). A complete report of the studies and replication files are also available (Carvalho, 2015b).

3.5.1 Participants

We recruited, in total, 32 participants aged 19–44 ($M = 23.34, SD = 4.78$). Participants of Study 1 ($N = 13$) were students of a Masters-level course on Entrepreneurship using SGs in the University of Genoa, Italy. Participants of Study 2 ($N = 15$) were industrial engineering students of an undergraduate course at the University of Bremen, Germany. For Study 3, we recruited, via specialized mailing lists and social media, a group of self-identified SG experts ($N = 4$), who were offered a small monetary compensation for their time.

Table 3.14 lists the participants' self-reported level of familiarity with digital games and with SGs on a 1–5 scale.

Familiarity	With games	With SGs
None	.	.
Played once or twice	.	17
Played a few times	14	10
Plays now and then	3	1
Plays frequently	15	4
Sum	**32**	**32**

Table 3.14: Participants by familiarity with games and with SGs

3.5.2 Setup

The general structure of the three studies was the same, with the difference that, while in Study 1 the participants evaluated only the ATMSG model, in the subsequent studies they evaluated both ATMSG and LM-GM.

Study 1 Participants used the ATMSG model to analyze the game Marketplace Live, a business simulation SG, which they had been playing for 8 weeks as part of the normal activities of the course.

Study 2 Participants evaluated both ATMSG and LM-GM over the course of two weeks. The games used in the evaluations were Playing History: Vikings, an adventure game to teach history for children, and Senior PM Game, a simple simulation game to teach project management to university students. Participants were split into two groups to alternate which model was used first. Three participants did not participate in the second day of the evaluation. Their responses were discarded in the comparisons between the two models (since they did not have a matching sample), but were kept to compute average usability scores.

Study 3 Participants were asked to evaluate one single game (Senior PM Game) using both ATMSG and LM-GM. The order in which the models were presented to each participant was assigned at random. The study was conducted using an online survey tool.

In all cases, participants first received an explanation of the model. Subsequently, they were asked to apply the model to analyze a SG, using either paper or digital templates. The templates included the complete tables of the taxonomy of serious games elements Section 3.3. After completing the analysis, participants were asked if the model had contributed to any change in their perception of the game and if they had any suggestions to improve the model. In the cases where participants evaluated both ATMSG and LM-GM, we also asked them to compare the models.

We asked the participants about their experience with the model using an adapted version of the System Usability Scale (SUS) questionnaire (Brooke, 1996). SUS is a simple, ten-item attitude Likert scale giving a global view of subjective assessments of usability, which yields a single usability score on a scale of 0–100.

3.5.3 Qualitative data processing

In addition to the usability scores, we also collected qualitative data on the participants' experiences with the models: the open-ended questions on how the model affected their perception of the game, the comparisons between ATMSG and LM-GM, the game diagrams and tables (Figure 3.6 and Figure 3.7[1]) and written observations made on the days of the studies.

To process the participants' comments, we first discarded empty answers, and answers in which the participant misunderstood the question (e.g. they provided feedback about

[1]Figure 3.6 and Figure 3.7 show only the Gaming actions, tools and goals layers (with a slightly different notation, reflecting an earlier version of the table template). For simplicity, the six layers representing Learning and Instructional actions, tools, and goals have been omitted.

the game itself and not about the model). We were left with feedback from 25 participants. These answers were coded to identify general statements about both models. Each answer could have one or more general statements. These general statements were grouped, and the results are presented subsequently.

3.5.4 Results

We could identify usability issues with the ATMSG model, both in the qualitative data and in the scores obtained with the SUS questionnaire. Results from the three studies indicate that the ATMSG model has a steep learning curve: six participants (19%) mentioned that the application of the ATMSG model could be simplified, and, in four cases (12%), the participants stated that they needed detailed instruction and examples to be able to perform the analysis. This feedback was consistent with the average usability score of the ATMSG model obtained from the SUS questionnaires, which was 58.83 ($N = 30$, $SD = 17.5$), in a scale of 0–100.

Nevertheless, in general, participants considered that using either model helped them better understand the characteristics of the game. LM-GM was positively evaluated by a large number of participants: among the 18 participants who used LM-GM, 13 (72%) mentioned that LM-GM was helpful for them. Conversely, 14 participants (47% out of 30) said the same about the ATMSG model.

To make direct comparisons between ATMSG and LM-GM, we only considered data from participants who used both models, discarding three responses that did not have a matching sample. We were left with a sample of 32 questionnaires from 16 participants. Thirteen participants (81%) stated that ATMSG is more complete and detailed than LM-GM, and two participants (12%) considered that the ATMSG game diagram is easier to draw. LM-GM, on the other hand, was considered simpler or easier to grasp by nine participants (56%), seven of them non-gamers. One participant commented that maybe they would have been more comfortable with ATMSG if they had been exposed to LM-GM first.

The difference in perception between the two models noted in the qualitative data was also apparent in the SUS usability scores, although the sample size is too small to draw definitive conclusions. For the non-gamers group ($N = 8$), the average usability score for ATMSG was 42.8, while for LM-GM it was 57.5. For the gamers group ($N = 8$), the average usability score for ATMSG was 74.4 and for LM-GM it was 65.6. A mixed between-within subjects ANOVA was conducted to compare the usability scores given by participants for each of the two models and to identify if the scores varied with familiarity with games. There was a significant effect of the different levels of familiarity in the usability scores, $F(1, 14) = 14.87, p = .002, \eta_G^2 = 0.32$, but no effect due to the model used, $F(1, 14) = 0.37, p = 0.55, \eta_G^2 = 0.007$.

We also considered in the qualitative analysis our observations of the participants' behavior and the documents they delivered at the end of the studies. We noticed that

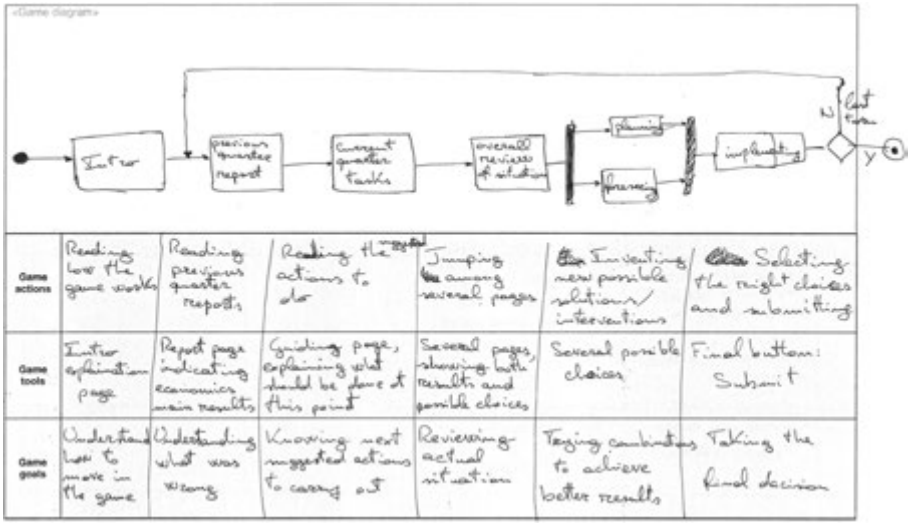


Figure 3.6: ATMSG diagram layers filled as expected, with proper vertical alignments

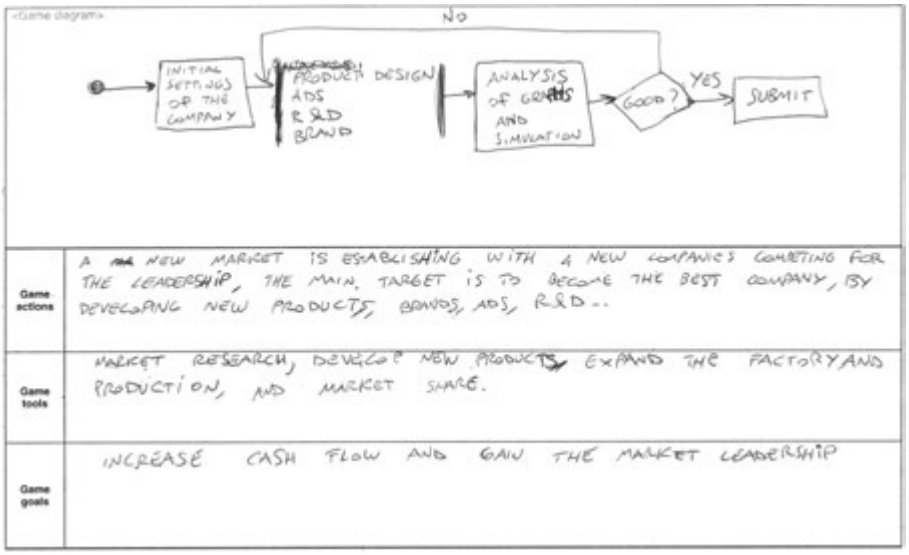
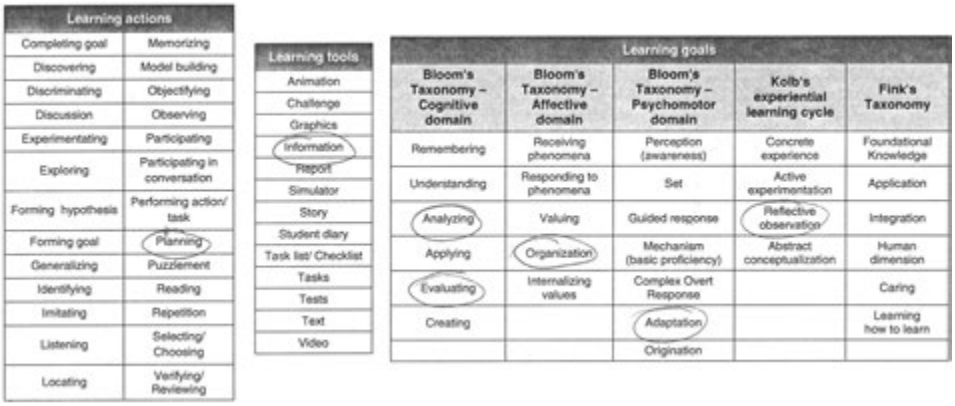


Figure 3.7: ATMSG diagram layers filled in an unexpected way

some participants using ATMSG relied heavily on the taxonomy to perform their analyses, particularly in making the distinction between items from gaming and learning activities. Furthermore, some participants did not produce the game diagram with the components in a vertically matching position (Figure 3.7), simply preferring to circle the relevant items in the taxonomy reference tables (Figure 3.8). Finally, in many instances during the studies, participants asked for clarifications on the entries of the taxonomy of SG components, being unsure of the meaning of certain terms.



Learning actions	
Completing goal	Memorizing
Discovering	Model building
Discriminating	Objectifying
Discussion	Observing
Experimentating	Participating
Exploring	Participating in conversation
Forming hypothesis	Performing action/ task
Forming goal	Planning
Generalizing	Puzzlement
Identifying	Reading
Imitating	Repetition
Listening	Selecting/ Choosing
Locating	Verifying/ Reviewing

Learning tools
Animation
Challenge
Graphics
Information
Report
Simulator
Story
Student diary
Task list/ Checklist
Tasks
Tests
Text
Video

Learning goals				
Bloom's Taxonomy – Cognitive domain	Bloom's Taxonomy – Affective domain	Bloom's Taxonomy – Psychomotor domain	Kolb's experiential learning cycle	Fink's Taxonomy
Remembering	Receiving phenomena	Perception (awareness)	Concrete experience	Foundational Knowledge
Understanding	Responding to phenomena	Set	Active experimentation	Application
Analyzing	Valuing	Guided response	Reflective observation	Integration
Applying	Organization	Mechanism (basic proficiency)	Abstract conceptualization	Human dimension
Evaluating	Internalizing values	Complex Overt Response		Caring
Creating		Adaptation		Learning how to learn
		Origination		

Figure 3.8: Some participants preferred to circle items in the taxonomy reference tables

3.6 Discussion

The ATMSG model has the objective of supporting the analysis and design of SGs for two user groups: experts in SGs, and non-experts who are involved in SGs related projects, such as teachers and application domain experts (e.g. trainers, advertisers, managers). Our preliminary evaluation indicates that the structured analysis supported by ATMSG is helpful to users in understanding in depth the roles of each piece in each action that happens inside the game. The decomposition of components is more detailed than that provided by the LM-GM model, which only specifies two main sets of components, namely "game mechanics" and "learning mechanics", with no other distinctions on the nature of these components. For example, an LM-GM analysis of the game DragonBox Algebra 5+ is able to identify that the component *tutorial* is present as a "learning mechanism". An ATMSG analysis of the same game, conversely, allows the user to be more precise and describe that the player's action of *observing* the *tips* exposes him to the mathematical concepts that have to be *remembered*. Those same *tips* are used by the game when *demonstrating* allowed moves, thus *providing learning guidance* to the player. Furthermore, the ATMSG model also provides a more extensive list of components (almost 400 items classified in 36 categories, versus LM-GM's list of 38 game mechanics and 31 learning mechanics), which also contributes to the precision of the identification of components in the game.

The increased level of detail provided by ATMSG, nevertheless, results in a steeper learning curve, particularly to non-expert users or to those who are less acquainted with digital games. To this group of users, LM-GM's simpler analysis was already enough to provide useful insights on the game structure and educational purposes. This fact suggests that LM-GM provides a good understanding of the game when only a general idea of the game's learning mechanisms is needed, such as when several different games need to be quickly evaluated by non-SGs experts, e.g. teachers selecting a game for a class. ATMSG, conversely, is more suitable for situations in which a more profound understanding of the components is necessary, for example when adapting games for use in specific learning settings, when detailing the analysis to identify and catalog learning patterns or when evaluating game prototypes during the design process – in other words, when a thorough understanding of the characteristics of the game is needed.

The evaluation study highlighted a few usability problems of the application of the ATMSG model. Some of these issues, such as the need to simplify the ATMSG model and to clarify the tables, have already been addressed. Nevertheless, a few other points still need improvement, such as the fact that the taxonomy entries should be expanded to include descriptions and examples as well. A complementary textual description may provide details on typical characteristics of the item while examples would be useful to illustrate the actual usage of the item in a game. Additionally, the application of the taxonomy to a wider variety of SG genres could favor the expansion or refinement of the entries by incorporating eventually missing components and removing those that are not relevant. Finally, we identified the need to provide a more appropriate medium for the application of ATMSG, such as a computer application, a set of cards or similar tools that can be easily moved around while performing the analysis.

In summary, the ATMSG model provides a comprehensive way to investigate, in detail, how a serious game is structured, using activity theory as the theoretical background. It can be useful not only for SG analysis, but also as a tool for SG design. Compared to other models, methodologies and frameworks currently available, ATMSG offers a more precise model for the analysis of the educational and gaming aspects of a game, allowing the user to perform a more exhaustive decomposition of components as the game unfolds, and to link these components to the overall learning objectives. It offers a detailed graphical and textual representation of the SG that facilitates comparing different games. Users with familiarity with digital games were more comfortable with the model. For non-gamers, ATMSG seems to have a somewhat steep learning curve, although this user group still recognizes benefits from applying the model in the analysis of serious games.

ATMSG provides a more detailed analysis of serious games but is also more complex. Consequently, the application of the method requires more time and a better understanding of game components. Thus, if only a general idea of pedagogical aspects of a game is needed, other tools (e.g. Four-Dimensional Framework, Relevance, Embedding, Transfer, Adaption, Immersion and Naturalisation (RETAIN) model, LM-GM) are most likely more appropriate. LM-GM, which was the other model evaluated in this

work, also provides users with valuable insights into the structure of a serious game, but the description of the inner components of the serious games are not as detailed as ATMSG.

❧

Up to this point, I focused on theoretical aspects of the design of SGs. The next two chapters of this thesis move to technical considerations about SG software development. In Chapter 4, I focus on the role of a game's software architecture in the development process. Subsequently, in Chapter 5, I present the second main contribution of this thesis, the Service-Oriented Reference Architecture for Serious Games (SORASG).

Chapter 4

Software development and serious games

In this chapter, I first briefly introduce software architectures and reference architectures, also discussing their design and evaluation. Next, I present a review of architectural strategies that have been employed so far in games and serious games (SGs) for reducing development costs. Finally, one architectural approach – Service-Oriented Architecture (SOA) – is further analyzed as an underused but potentially beneficial approach to SG development.

4.1 Software architectures and reference architectures

A software architecture is an abstract representation of a system, which purposely hides internal information that has no ramification outside of an element with the objective of reducing the amount of complexity that has to be dealt with at a given time (Bass et al., 2012)[2].

The main benefit of taking the time to design an appropriate software architecture is that it allows important decisions about the system's characteristics to be defined early in the project. Early decisions allow for better communication with stakeholders and more precise estimates of a system's development schedule, costs, and qualities. The alternative – letting the architecture "emerge" during the coding phase – does not give the

Parts of this chapter were published previously in Carvalho, Bellotti, Berta, Gloria, Gazzarata, et al. (2015).

[2]Much of the literature in software engineering refers to a related, but broader concept: enterprise architectures. Enterprise architectures include not only software architectures as described in this chapter, but also how the software is used by humans to perform business processes, and the whole computational environment in which the software is going to be used (The Open Group, 2011; Bass et al., 2012).

team any way to deal with potential problems (e.g. conflicting requirements, unforeseen costs) before they arise, possibly bringing unexpected difficulties later on that can significantly increase development costs. As mentioned in Chapter 1, software development costs indicate the effort required to develop or maintain software, expressed in hours of work per person and/or amount of money required to carry out the project (Heemstra, 1992).

Well-defined software architectures also facilitate the practice of evolutionary prototyping, in which the software is quickly implemented as a skeletal system: parts of the infrastructure and components are realized by third-party components that are plugged into the architecture, conforming to the defined communication flows. In this way, the system is executable early in its lifecycle, facilitating prompt detection of potential problems in the overall structure and the user interface. When the architecture has been consolidated, the prototyped or surrogate parts can be substituted by complete, improved versions as the development progresses (Bass et al., 2012). Evolutionary prototyping can be used not only to speed up development, but also to increase overall system quality. Furthermore, early detection of problems contributes to lower development costs, since generally it takes less effort to fix problems in earlier phases of the project.

Nevertheless, creating adequate software architectures can be difficult and time-consuming, requiring many iterations to get a satisfactory solution. For this reason, reference architectures are used as template solutions for a determined domain, providing major guidelines for the specification of the architecture of one class of systems (Angelov, J. J. M. Trienekens, & Grefen, 2008). Reference architectures can also function as generalizations of a set of solutions, often based on existing projects, with the objective of establishing standards for future projects (Reed, 2002). They are similar to design patterns, but have a broader scope, targeting whole systems instead of just specific software development problems. Reference architectures can be used by software architects as a starting point to support the definition of architectures for particular projects according to best practices, thus facilitating decision-making in the early phases of a project (Reed, 2002). They can also help promote reusability and interoperability of the software produced (Angelov, J. Trienekens, & Kusters, 2013), which, once again, can contribute to reducing development costs through component reuse.

4.1.1 Types of reference architectures

Usually, a reference software architecture is defined as a list of components, functions, interfaces, interactions between internal and external components, common vocabularies, and so on. Examples of reference architectures include the Reference Architecture for Adaptive Hypermedia (AHA) (Wu, 2002), the High Level Architecture for simulations (HLA) (Kuhl et al., 1999), and the OATH Reference Architecture (OATH, 2007).

Reference architectures can have different characteristics, depending on their goals, context of application, and design. A taxonomy of reference architectures presented by Angelov, Grefen, and Greefhorst (2012) defines a total of five main types of reference

architectures and several variants. It classifies the architectures according to a series of criteria, such as where they will be used, who creates them, if they describe existing or future systems, their goals (standardization of existing systems or facilitation of development of future systems), which elements are described and at which level of detail, abstraction, and formality. The taxonomy aims to help the architect in designing a reference architecture with characteristics that match the context and goal, or, in other words, that are "congruent". According to the authors, architectures that are congruent have a higher chance of acceptance by the target user group.

4.1.2 Design and evaluation

To design a software architecture, be it a "concrete" architecture or a generic reference architecture, the crucial first step is collecting the architecturally significant requirements (ASRs), that is, the requirements that have a direct impact on the architecture. ASRs emerge not only from the functional requirements of a system but also from its quality attributes (QAs) and constraints (Bass et al., 2012).

Functional requirements define what the system is supposed to achieve. QAs, or non-functional requirements, conversely, indicate how well the system achieves it, in relation to some dimension of interest to the stakeholder (Bass et al., 2012). Common QAs for software are reliability, availability, usability, testability, modifiability, among others. QAs can be further characterized by using *quality attribute scenarios*, which have the benefit of making a QA more specific and consequently unambiguous and testable (Bass et al., 2012). Scenarios are an important asset for system architects and occupy a central role in many methods for architecture development and evaluation.

Once the requirements have been defined, the architect can proceed to propose the architecture design. One methodology for the design of reference architectures is the Attribute Driven Design (ADD) methodology (Bass et al., 2012). This methodology features a set of specific steps that an architect must follow to develop a software architecture. It recommends that the architect focus on smaller parts of the system at a time, selecting only the requirements relevant at that stage, proposing a design, evaluating the design against the requirements and making adjustments or proposing alternative designs. It is a highly iterative process, because each interaction must include a re-evaluation of how the parts affect the overall requirements of the system.

Software architectures should be evaluated in the context of their explicitly stated goals. The extent to which a software architecture achieves those goals (or requirements) is used to judge the quality of the architecture. Consequently, the process of evaluating a software architecture is typically carried out by systematically matching the characteristics of the architecture to its previously defined requirements (Bass et al., 2012). Among the many methods dedicated to evaluating the quality of software architectures, Architecture Trade-off Analysis Method (ATAM) has established itself as one of the preferred methods, notably because of its easy integration with the design process (Ionita, Hammer, & Obbink, 2002). ATAM is a scenario-based method for assessing the quality of sys-

tem architectures, which provides a structured way to identify risks, sensitivity points, and trade-off points in the system. ATAM is especially suited for evaluating systems in relatively early design stages and prototypes, serving as a valuable way for discovering weak spots that can be addressed in subsequent iterations of the development (Kazman, M. Klein, & Clements, 2000). ATAM relies strongly on the concept of *scenarios*, which are explicit descriptions of the possibly ambiguous requirement statements into more concrete, measurable scenarios of use.

Although many characteristics of the design and evaluation of reference architectures are similar to those of "concrete" architectures, there are some important differences. Since reference architectures have a more generic nature, methods aimed at concrete software architectures are not necessarily able to deal with these specific characteristics (Angelov et al., 2008). In particular, an ATAM evaluation of a reference architecture cannot be performed without some adaptations, due to the lack of a clear group of stakeholders to define the desired requirements, and the very generic nature of reference architecture, which makes it difficult to generate concrete scenarios. To address the shortcomings of ATAM for evaluating reference architectures, Angelov et al. (2008) propose adaptations to the ATAM method. In short, the authors propose that, instead of asking stakeholders about generic scenarios for the reference architecture, the architect tries to elicit from stakeholders a variety of concrete scenarios (i.e. scenarios that are specific to a concrete system, and not to a class of systems), which are easier to generate and to attribute measurable qualities to. With a representative list of concrete scenarios, the architect can then combine them into generic attributes and scenarios that are relevant to the reference architecture.

4.2 Reusability in SG design and development

In the entertainment game industry, because of the increasing costs of ever more realistic games, developers started looking for ways to optimize development. One of the main strategies is reusing game components, specifically through the use of game engines. Game engines are collections of modules that handle input, output and generic physics or dynamics of the game world, general enough to be reused in different games (Lewis & Jacobson, 2002). Incorporating such components is expected to reduce development time and cost (as buying components is usually cheaper than developing them from scratch) and ensure quality (since the components are tested and used in different environments); furthermore, it allows game developers to focus their efforts in one single area, thus allowing advances at faster rates (Folmer, 2007). Nevertheless, integrating components and managing the complexity of the resulting architecture are still challenges that need to be overcome (Folmer, 2007).

BinSubaih and Maddock (2007) present an architecture that aims to enable game portability across different game engines, with the objective of removing the strict dependency of a game to the engine underneath it. The work highlights well the benefits of

separation of concepts and reusability in game design.

Specifically for SGs, similar strategies have been proposed to reduce development costs.

Bellotti, Berta, De Gloria, and Primavera (2009) suggest that decoupling the content of the SG from the underlying gaming software is a way of facilitating the extensibility of SGs and to support domain experts in the creation of content, which can then happen independently of the development of the game itself. The project Travel in Europe (TiE) is an example, in which an architecture style that supports both code reuse and consistent interaction modalities across games is proposed (Bellotti, Berta, De Gloria, & Zappi, 2008; Bellotti et al., 2010; Bellotti, Berta, De Gloria, D'Ursi, & Fiore, 2012). The MetaVals Serious Game, a game for practicing basic finance concepts, also implements decoupling between content and game: it consists of a modular database and an independent graphic interface, with a management interface that facilitates configuring the game to different contexts (Popescu, Romero, & Usart, 2012).

The use of authoring platforms is another strategy aiming at reducing the complexity of game development. The eAdventure game platform serves as an authoring platform for educational point-and-click adventure games, executing games defined in a specialized markup language (Moreno-Ger, Sierra, Martínez-Ortiz, & Fernández-Manjón, 2007; Torrente, Blanco, Moreno-Ger, Martínez-Ortiz, & Fernández-Manjón, 2009). The authoring tool Puzzle-it divides the process of developing games into content authoring and core engine development, making it possible for instructors to create content for the games via the authoring tool without needing to be concerned about engine behind the games (Pranantha, Bellotti, Berta, & De Gloria, 2012).

The Horizon-2020 Programme RAGE project, initiated in early 2015, has the objective of creating software modules for SGs and making them available as software assets to facilitate SG development (Vegt, Westera, Nyamsuren, Georgiev, & Martínez-Ortiz, 2016). The architecture defines assets as the software modules that can be linked or integrated with the game. These assets can be incorporated locally in the game engine (client-side assets) or remotely via network calls (server-side assets). The project is still in its early phases, and the architecture defined so far defines in general terms the general structure of the client-side components that are going to be developed throughout the project.

4.3 Service-oriented architectures

SOA is a software architectural pattern (Bass et al., 2012) that "implements business processes or services by using a set of loosely coupled, black-box components orchestrated to deliver a well-defined level of service" (Hurwitz, Bloor, Baroudi, & Kaufman, 2006). It is a set of ideas, recommendations, policies and practices for architectural design. One of its goals is to employ modularization and compositionality to achieve flexibility and to enable the reuse of software parts, in an attempt to manage the complexity of large

systems (Sprott & Wilkes, 2004; Aalst, Beisiegel, Hee, Konig, & Stahl, 2007).

4.3.1 SOA and SG development

SOA architectures are already widely and successfully employed in several areas of software engineering, including game development. One reason is that core principles of SOA, such as modularization and compositionality, can help achieve flexibility in the development and enable reuse of software parts. Unlike the case of traditional library reuse, which requires replication of code, SOA supports reuse of the services themselves, which provides a significant benefit regarding having up-to-date components without concerns about maintenance of the code. Also, it supports such a level of abstraction that multiple services can offer the same functionalities, potentially giving the developer a wider choice of providers from which to obtain the service needed. Furthermore, SOA establishes standardized contracts between endpoints, placing formal obligations between the consumer and the provider of the service and largely increasing reusability and interoperability. An implementation that complies with known web service standards (e.g. Representation State Transfer (REST) or Simple Object Access Protocol (SOAP)) has additional benefits, such as standardization, technology/platform neutrality and automatic discovery and use (Sprott & Wilkes, 2004). The automatic binding of services removes compile-time dependencies; the interface definition happens in runtime, removing the need to alter the code whenever there is a change in the service provider. This provides flexibility in the development and improves maintainability (Stevens, 2005; Erl, 2005).

There are, nevertheless, challenges in adopting SOA. Quality assurance and testing module integration tend to be more difficult when developing SOA applications (Hurwitz et al., 2006). Also, a service can be practically unusable if its interfaces lack clarity or are badly documented. Finally, extra attention has to be given to service descriptions, as they are the way to advertise the capabilities, interfaces, behavior and quality of a service, providing the required information for discovery, selection, binding and composition with other components (Papazoglou, Traverso, Dustdar, & Leymann, 2007).

The use of SOA in entertainment games is not uncommon. Houten and Jacobs (2004) present an SOA architecture for distributed multiplayer simulation games that can be used for training and learning purposes. Shaikh, Sahu, Rosu, Shea, and Saha (2006) describe the design of an on-demand service platform to enable sharing resources across online games, particularly targeted to solving problems of scaling the infrastructure in response to players' demand in massively multiplayer online games.

In addition, there is an increasing availability of service-based tools for game development that make the choice of employing SOA in game development a particularly appealing alternative. Such services include cloud-based infrastructure for building, deployment and distribution (Amazon Web Services, 2014), platforms providing social connectivity to games (Facebook, 2014) and services that provide generic gaming features such as achievements, leaderboards and cloud saving (Lumos, 2014; Hartrell, 2013).

Tools created specifically for SGs are less common in comparison to tools for entertainment games. For example, the Serious Games Web Services Catalog (Serious Games Society, 2013a) is a repository of web services that acts as a showcase for services for SG development; nevertheless, as of the writing of this thesis, it contains only seven services available, with two of them in prototype stage.

The SOA approach can enhance SG quality, particularly by enabling the incorporation of features that are still rare in SGs, such as adaptation techniques, learning analytics, and social media integration. It can also bring the potential benefit of decreased interdependencies and usage-dependent payment models (Arslan, 2012). Furthermore, it facilitates dealing with scalability issues, which is particularly relevant to online games in which several thousand players interact among themselves in a common platform, as the increased load on the servers may bring performance concerns (Arslan, 2012). SOA also makes it possible to access games from simple devices, eliminating the dependency on the quality of gaming hardware. Also, providing pervasive gaming experiences becomes easier, as support for different platforms is highly simplified if the core of the gaming experience is provided via a service on a centralized server (Hassan et al., 2012).

Despite the suggested benefits of the application of SOA to SG development, there are very few examples of SOA-based SGs.

While the game itself has not been developed, a Service-Oriented Architecture was the approach of choice for an envisioned gaming platform based on mobile augmented reality, called Mobile Augmented Reality Learning (MARL). In this system, on-demand location-based instruction would be delivered through a head-mount display by a virtual instructor. The complete MARL game service would be composed of subsystems that would provide visual, human computer interface, and training services, allowing for the lower level objects to be encapsulated by the higher level interfaces, making it easier for improvements in the algorithms to be incorporated into the service (Doswell & Harmeyer, 2007).

The Rashi Intelligent Tutoring System teaches human anatomy through a problem-based environment. Rashi is built with a web service architecture that supports on-demand requests for small chunks of specific knowledge, instead of requests for an entire case specification at once, giving developers the flexibility to develop lightweight inquiry tutors that run efficiently over the web (Floryan & Woolf, 2011). On top of the same existing service structure for the original (2D) inquiry system, the researchers built a 3D game in which the student is a doctor who must diagnose a patient in a virtual hospital.

The Journey is a SG that teaches basic concepts of probability theory to high school and entry-level university students. We developed the game as a prototype implementation of a service-based adaptive SG, employing the Competence-based Knowledge Space Theory (CbKST) service to implement basic adaptation features for learning (Carvalho, Bellotti, Berta, Gloria, Gazzarata, et al., 2015). Using our experience in the development of The Journey and the subsequent evaluation of the architecture, we argue that the application of SOA in the development of SGs can result in shorter development

times, and more focus and flexibility in the development. The most immediate benefit is the possibility to reuse services, as it directly impacts development times and the ability of development teams to focus their efforts on other aspects of game development (e.g. graphics, questions, game flow). Nevertheless, points of attention were also identified, particularly related to the performance of the game in case of unexpected network/service unavailability, failure or slowdowns.

4.3.2 Open Group SOA Reference Architecture

While there are many benefits in adopting SOA, there can also be many challenges, particularly at the enterprise level. Achieving consistent architectures that meet the desired objectives is important, but the learning curve to obtain those results is high, particularly due to the significant shift in thinking that switching to a services paradigm requires.

For that reason, a framework has been created to provide guidelines and options for making architectural, design, and implementation decisions in SOA solutions. The Open Group SOA Reference Architecture (The Open Group, 2011) is intended to be a blueprint for creating or evaluating service-based enterprise architectures. It has a two-dimensional view with nine layers representing key clusters of considerations and responsibilities that typically emerge in the process of designing an SOA solution or defining an enterprise architecture standard (see Figure 4.1).

The horizontal layers (i.e. Consumer interfaces, Business processes, Services, Service components and Operational systems layers) are functional layers. The lower layers (Services, Service component, and Operational systems) are concerns for the provider, and the upper ones (Services, Business processes, and Consumer) are concerns for the consumer. The vertical layers (i.e. Integration, Quality of service, Information, and Governance) represent cross-cutting concerns of a more supporting (non-functional or supplemental) nature.

The services in the services layer can be sourced by an individual component, or by a composition of services (Figure 4.2). In other words, the relationship does not have to be one to one. One service can be implemented by more than one component (Service 1 in the figure exposes functionalities from both Component 1 and Component 2), and multiple services can expose functionalities offered by one single component (Services 2 and 3 expose functionalities provided by Component 3). It is also possible that the service itself provides functionalities, without any component behind it (Service 4 in the figure).

There are two main ways of establishing compositions of services, that is, defining the sequence of activities that allows services to achieve a common goal: *choreography* and *orchestration*.

A choreography model captures the interactions in which the participating services en-

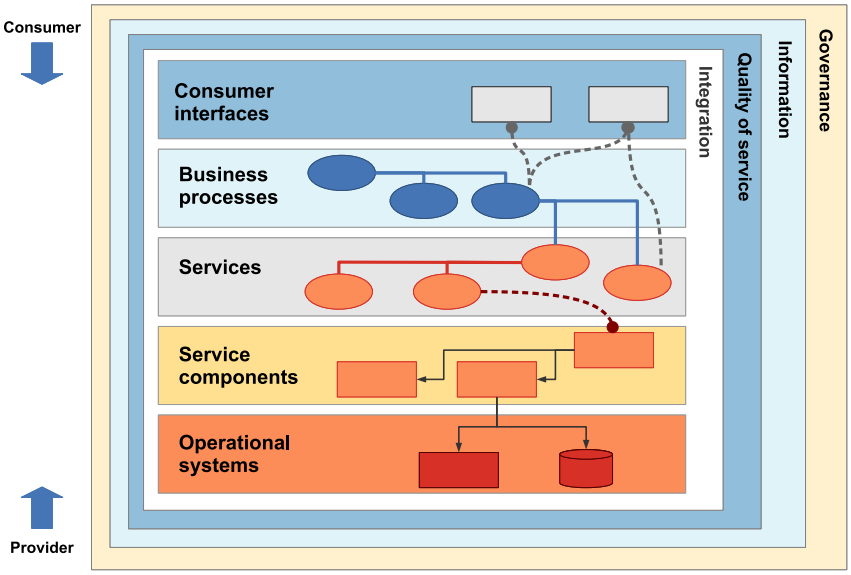


Figure 4.1: Layers of the Open Group SOA Reference Architecture (The Open Group, 2011)

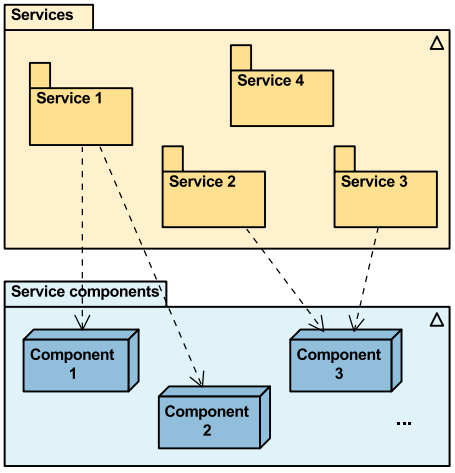


Figure 4.2: Service components

gage and the dependencies between these interactions, and it includes causal and/or control-flow dependencies, exclusion dependencies, data-flow dependencies, interaction correlation, time constraints, transactional dependencies, among others (Barros, Dumas, & Oaks, 2006). It is decentralized, so there is no central entity controlling the flow of messages and agreed rules of interaction between the services (left side of Figure 4.3).

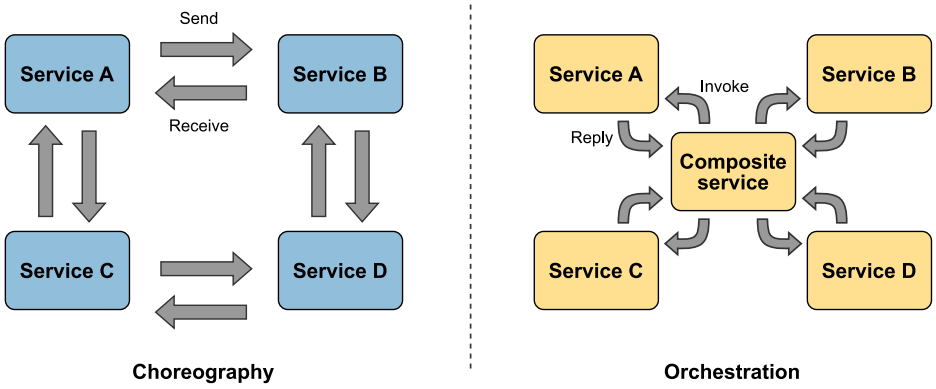


Figure 4.3: Choreography versus orchestration

An orchestration model, conversely, represents a process that is controlled by one party, in an executable business process (or orchestration engine) that coordinates the services, invoking and combining them (Peltz, 2003). The right side of Figure 4.3 illustrates the orchestration model.

In the Open Group SOA Reference Architecture, the Business Process layer represents the parts of a system that are responsible for managing the flow of the activities of the services. In other words, it is responsible for orchestrating services in the Services Layer.

❧

This chapter discussed concepts of software engineering that are important to understand and evaluate the reference architecture proposed in Chapter 5 as the response to RQ3. It also showed how the SOA architectural strategy, on which the Service-Oriented Reference Architecture for Serious Games (SORASG) (Chapter 5) is based, can bring benefits for SG development, particularly in reducing development cost while maintaining software quality.

Chapter 5

Service-Oriented Reference Architecture for SGs (SORASG)

In Chapter 3, I described the Activity Theory-based Model for Serious Games (ATMSG), which is a tool that allows us to analyze serious games (SGs) of different genres and topics, identifying their underlying structure and most important components. In this chapter, I propose a software reference architecture called Service-Oriented Reference Architecture for Serious Games (SORASG), based on the Service-Oriented Architecture (SOA) architectural pattern. The SORASG used the ATMSG model as a starting point to identify requirements for the architecture.

In the next sections, I first outline the relevance of the SORASG, highlighting how it relates to the overall objectives of this thesis (Section 1.2). Subsequently, I describe the process used in defining the architecture (Section 5.2). In Section 5.3, I list the requirements that guided the elaboration of the SORASG, including how the ATMSG model guided the beginning of the process and how the quality attributes of the architecture address the business goals, also presented in Section 1.2. Then, in Section 5.4, I describe the reference architecture in detail. An example implementation of the SORASG is briefly described in Section 5.5, demonstrating the buildability of the reference architecture. Finally, in Section 5.6, I report the results of an evaluation of the SORASG.

5.1 Relevance

As mentioned previously, one of the research questions of this thesis is stated as: *RQ3: how can SG developers incorporate reusable components into their software development*

Parts of this chapter appeared previously in Carvalho, Bellotti, Hu, et al. (2015).

projects? This question comes as a consequence of the following business goals: the necessity of reducing costs associated with the development of SGs, while maintaining the quality of the games developed (BG1); allowing the reuse of existing technological solutions (BG2), thus providing the SG developer with access to quality and up-to-date components; and promoting the use of open standards and technology-independent solutions (BG3). These business goals were presented and discussed in Section 1.2.

The SORASG was built to address directly the three business goals listed above. The in-depth discussion of how these goals are achieved is presented in Subsection 5.3.3. Here is a summary:

Firstly, the SORASG can be used as a starting point for development by offering established solutions to typical problems encountered in the field. In this way, it can reduce the complexity of the process at the beginning of the project, when decisions tend to be more critical and have greater effects in its future phases.

Secondly, the SORASG offers a way for developers to reason about the system early in the process, particularly for uncovering how the structure of the software is related to the desired requirements and quality attributes (QAs). Making critical decisions early reduces the probability of rework, by ensuring from the beginning that the system is built according to the requirements. This way, late-stage changes, which are usually much more costly than earlier in the process, are avoided.

Finally, the SORASG can facilitate consistency based on best practices, and guide developers on which aspects, building blocks, and layers of an SOA-based SGs should be considered when designing solutions or evaluating a game architecture. At the same time, this consistency provides a common understanding of SGs elements, requirements and typical solutions, which in turn can improve communication within teams and within the industry/research groups through common vocabulary and concepts.

The points listed above are common to all reference architectures, and software architectures in general. Nevertheless, currently there are not any other reference architectures aimed specifically at educational SGs, nor any significant effort in employing SOA in the development of SGs, as discussed in Subsection 4.3.1.

There is one more benefit of the SORASG, which is particular to the work that I present in this thesis. The SORASG aims specifically at promoting the reuse of pieces of software for SGs that are common among SGs of different genres and different learning topics. Reuse of software is associated with reduced development time. In this case, I promote specifically the implementation of games in an SOA architecture, to take advantage of its high emphasis on the use of decoupled interfaces, and to facilitate the production of more readily integrated systems (see Subsection 4.3.1).

The SORASG can benefit SG developers, SG designers, and researchers in the field directly, as is argued in Section 5.6. It is also expected that the SORASG will aid vendors in providing software products that are more likely to support the needs of SG developers.

5.2 Architecture definition process

The SORASG was the result of an iterative process in which design and consultation with stakeholders (and consequently evaluation) were constantly intermingled. In this chapter, I give an overview of the process and methods that were followed. I also describe the involvement of stakeholders in the process.

Figure 5.1 illustrates the different phases of this work and how they influenced each other. The arrows indicate when the outcome of a certain phase was used as input for another phase. Chronologically, the process happened as illustrated in Figure 5.2. Each relevant aspect of the process is detailed in the following sections.

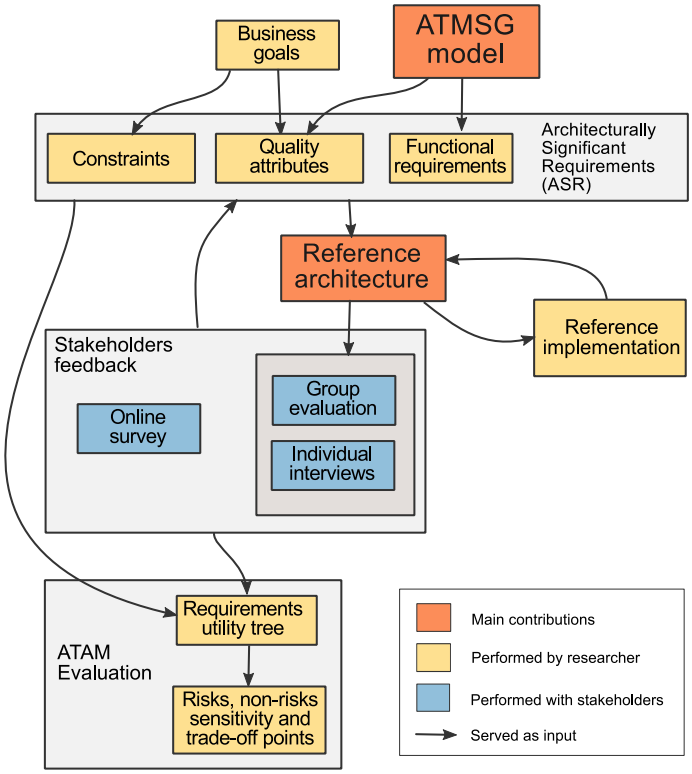


Figure 5.1: Relationships of each step in elaborating the SORASG

5.2.1 Requirements definition

The first step in the process of designing the SORASG was to define a series of functional requirements that would be relevant for the whole field of the development of educa-

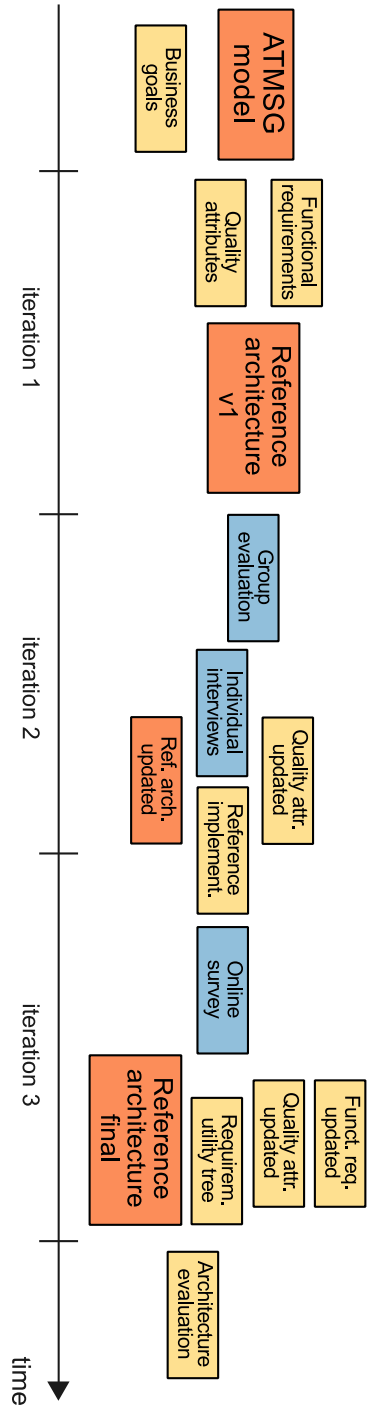


Figure 5.2: Timeline for elaborating the SORASG

tional SGs. For that end, we made a qualitative analysis of the field to identify functional areas, subsystems, and desired goals. The ATMSG model was the starting point for this analysis. Subsection 5.3.1 expands this investigation and details how the ATMSG model was used to identify the functional requirements for the reference architecture.

We also created a list of QAs, collected from the business goals of the architecture and the analysis of the domain. At this point, we compiled an initial list of possible constraints for the architecture.

The functional requirements and QAs were validated and complemented with the feedback obtained from stakeholders, particularly in the individual interviews and the online questionnaire.

The updated QAs, functional requirements and constraints served as input to generate the list of architecturally significant requirements (ASRs) and its related scenarios in a utility tree. The ASRs are important input artifacts in the design of software architectures, for example when applying the Attribute Driven Design (ADD) methodology, and in the formal evaluation of software architectures and reference architectures. The ASRs and scenarios are described in Section 5.3.

5.2.2 Design iterations

Once the ASRs were defined, the process of designing the reference architecture started.

The first version of the reference architecture contemplated mostly the required modules and connections between them, without details on the interfaces. The focus of this iteration was to establish the division of modules and functionalities and the general pattern of interaction between them.

The first phase of consultations with stakeholders (group evaluation and individual interviews, see Subsection 5.2.3) informed smaller updates in the reference architecture. The feedback helped in identifying weak and unclear points in the description that needed to be addressed. At the same time, the development of the reference implementation also helped identify other necessary modules and refine details of the interaction between modules. This step marked the second version of the reference architecture.

After the feedback was collected and consolidated into ASRs, these requirements and their scenarios were used to update the reference architecture once again. The SORASG was also further refined to include the level of detail and necessary views that would be consistent with its purpose and audience (Angelov et al., 2012; Bass et al., 2012).

In total, the architecture was defined in three iterations (see Figure 5.2). The consolidated version of the SORASG is described in Section 5.4.

5.2.3 Stakeholders involvement

The design and evaluation of reference architectures is a slightly different process than that of software architectures (see Subsection 4.1.2). While involving stakeholders remains one of the main strategies, some adaptations in the process are needed to be able to collect knowledge and experience from the stakeholders and to use that knowledge in defining a relevant set of QAs and functional requirements to inform the design and evaluation.

We gathered a group of people that we considered representative of the stakeholders for the SORASG. A total of 18 people, divided into three groups, were consulted in three different stages: one group discussion, a set of individual interviews, and participation in online questionnaires. Table 5.1 summarizes the background of the participants interviewed and lists the stage in which they participated. Each stage is discussed in more detail in the following subsections.

Stage	Game/SG developers	Game/SG researchers	Others	Total
Group discussion	2	.	1	3
Individual interview	.	2	.	2
Online survey	6	2	5	13
Total	**8**	**4**	**6**	**18**

Table 5.1: Stakeholders consulted, by stage of participation and background

Group discussion

The group discussion happened during a tutorial on SOA development for SGs, offered as part of the 14th IFIP International Conference on Entertainment Computing (ICEC), held in Trondheim, Norway, in 2015 (Carvalho, Hu, Bellotti, De Gloria, & Rauterberg, 2015). The target audience of the conference consisted of practitioners, academics, artists and researchers involved or interested in the creation, development and use of digital entertainment in general. The conference offered a track on entertainment for purpose and persuasion, including games for learning and SGs. Participation in the tutorial was open to any registered participants in the conference. The tutorial was advertised as an introduction to the benefits of applying an SOA approach to SG development, including a practical session in which a preliminary version of the reference architecture would be used to rethink existing SGs or game ideas into services. Participants could then expect to learn about current service technologies that could be useful in their daily practice, while we, as the organizers of the tutorial, would obtain their feedback on our work in the reference architecture.

Three participants remained in the tutorial for the discussion and evaluation of the reference architecture. In the discussion, the explicit objective was to define QAs which

would be relevant to the definition of a reference architecture for SGs. Two of the participants were game developers while one participant was a secondary school teacher.

A more detailed discussion of the data collected during the group discussions is available in Appendix A.

Individual interviews

The second stage of the consultation with stakeholders happened as individual interviews. We consulted two academic researchers on game design: one full professor and one PhD student. The interviews followed the procedure: first, we explained to the participants, in general terms, the objective of the research. They were shown a preliminary version of the reference architecture, and the goals of the architecture were presented. They were asked to give their free impressions on the reference architecture, particularly focusing on if they believed the reference architecture would be useful in their daily practice. Finally, they were asked to mention how they thought the reference architecture could be improved. The whole procedure lasted 60–90 minutes.

A more detailed discussion of the data collected during the individual interviews is also available in Appendix A.

Online questionnaire

The last stage of consultation with stakeholders took the form of an online questionnaire. A total of 13 participants – including game and SG developers and researchers (see Table 5.1) – filled in the questionnaire. The participants were recruited online, via direct e-mail messages and social media posts. Participants were requested to forward the invitation to their contacts. The questionnaire was anonymous. Participants would only be asked to inform their email addresses if they agreed to participate in an eventual follow up on the questions. The questionnaire was expected to take 25–30 minutes. No compensation was offered. The questionnaire is available in Appendix B.

While the techniques for collecting QAs and scenarios from stakeholders, particularly in the context of architectural evaluation using the Architecture Trade-off Analysis Method (ATAM) method, are typically performed in synchronous co-located meetings, this is not a requirement of the method (Bass et al., 2012). Distributed and asynchronous methods can be more cost-effective because they are easier to organize, particularly when it is difficult to get all participants together for a full day event.

The main objective of the questionnaire was to collect desired QAs, scenarios of use and, most importantly, to prioritize the attributes. We also wanted to validate our list of functional requirements, which so far had been obtained solely from a theoretical analysis of the domain.

We followed the suggestion proposed by Angelov et al. (2008) of trying to obtain concrete QAs from the stakeholders, instead of abstract attributes from imagined situations

(see Subsection 4.1.2). For this reason, we asked the participants to select, from a predefined list of QAs, those attributes that were relevant to one project of SG design or development in which they were directly or indirectly involved in the past, or at least with which they were familiar[3]. Subsequently, we asked the participants to rank those QAs and write scenarios illustrating them.

To have an estimation of the variety of projects that were used to select the QAs and generate scenarios, we asked the participants to list the three most important educational and entertainment goals of their project. We labeled the answers, combining similar responses. Table 5.2 lists the labels that were mentioned by the participants, in order of frequency. From this table, we can see that the responses illustrate a wide variety of gaming and learning goals, ranging from games that aim to provide information about a topic or teach a particular skill, to simulations and games used to extract user profiles from interactions with the game.

	Item	Occurrences
1	Provide information about certain content	8
2	Teach a particular skill	7
3	Engagement and motivation	5
4	User profiling and stealth assessment	4
5	Promote creative thinking	3
6	Simulate a certain real-world situation	3
7	Allow for practicing a particular skill	2
8	Persuade about certain topic	2
9	Data logging	1
10	Provide real-life examples	1

Table 5.2: Game goals mentioned by participants in free text questions

We also collected desired functional requirements. We asked participants to list three main functionalities that they think are relevant for educational SGs in general, independently of their topic or learning objectives. Participants were asked to inform this as free text answers. However, some responses were not functional requirements, but rather quality attributes (QAs). We coded all answers, grouping similar responses, and subsequently excluded the answers that were not functional requirements.

To obtain an ordered list of the most important QAs, according to the participants, we asked them to select and rank a list of QAs that they considered relevant for the SG project in which they were involved. We attributed weights to occurrences of items in the ranking, in which a weight of 7 was attributed to the items ranked in the 1st position, and a weight of 1 was attributed to items in the 7th position. Items ranked lower than

[3]The full list and descriptions of quality attributes presented to the participants are available in Appendix D.

the 7th position, if any, were not considered. We calculated a normalized weight ($\bar{x}$) for each attribute according to (1):

(1)
$$\bar{x}_{\text{attribute}} = 10 \times \frac{\sum_{i=1}^{7} w_i n_i}{7P}$$
$$(w_j)_{j=1}^{k} = 8 - j$$

where i is the rank position, w_i is the weight for position i, n_i is the number of occurrences of the attribute in position i, and P is the total number of participants who provided ratings. The maximum position in the rank we consider, and also maximum weight that can be given to an attribute, is 7. The weights were multiplied by 10 to fit a 1–10 scale. The attributes were sorted in descending order of their normalized weights.

The original rankings are listed in Table 5.3. The ordered list of attributes is shown in Table 5.4 and illustrated in Figure 5.3. In the free-text questions, participants also mentioned the following QAs: functional correctness, data security, responsiveness, reusability of components and use of industry standards.

		Rank position							
	Attribute	1	2	3	4	5	6	7	**Count**
1	Availability	.	.	1	.	.	1	.	**2**
2	Deployability	.	3	1	1	1	.	1	**7**
3	Development distributability	.	.	.	1	.	1	.	**2**
4	Interoperability	.	1	.	.	.	.	.	**1**
5	Mobility	2	1	1	.	.	.	.	**4**
6	Modifiability	2	.	1	1	.	.	.	**4**
7	Monitorability	.	.	.	.	.	.	.	**.**
8	Performance	1	1	1	.	.	1	.	**4**
9	Portability	.	.	2	1	.	.	.	**3**
10	Safety	1	.	1	.	1	.	.	**3**
11	Scalability	.	.	.	.	.	.	.	**.**
12	Security	.	.	.	.	.	.	.	**.**
13	Testability	2	.	.	.	1	.	.	**3**
14	Usability	2	4	.	1	.	.	.	**7**
15	Variability	1	.	.	1	.	.	1	**3**
	Count	**11**	**10**	**8**	**6**	**3**	**3**	**2**	**43**

Table 5.3: Ranks attributed to each quality attribute

	Attribute	N	Total weight	Norm. weight (0-10)
1	Usability	7	42	5.45
2	Deployability	7	31	4.03
3	Mobility	4	25	3.25
4	Modifiability	4	23	2.99
5	Performance	4	20	2.60
6	Testability	3	17	2.21
7	Safety	3	15	1.95
8	Portability	3	14	1.82
9	Variability	3	12	1.56
10	Availability	2	7	0.91
11	Interoperability	1	6	0.78
12	Development distributability	2	6	0.78
13	Security	0	0	0.00
14	Scalability	0	0	0.00
15	Monitorability	0	0	0.00

Table 5.4: Normalized weights of each quality attribute

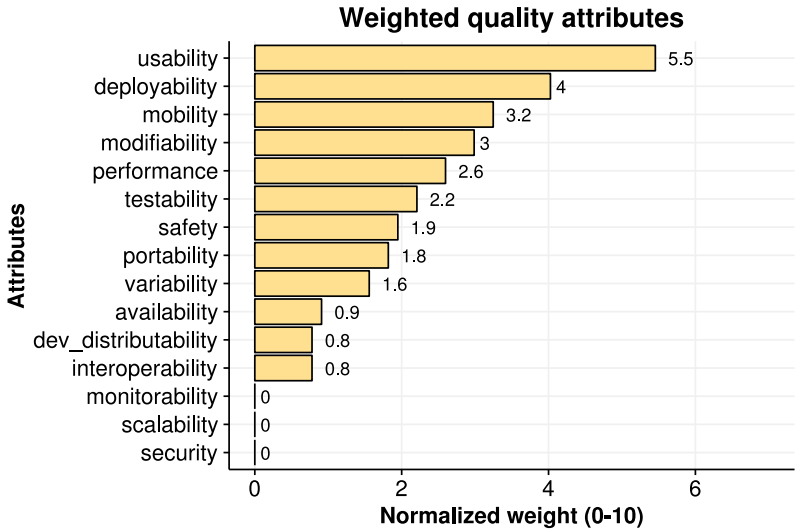


Figure 5.3: Ranked order of importance of QAs

Finally, we asked participants to write scenarios illustrating the top three QAs that they selected and ranked for their project. These scenarios served as input for us to write our consolidated scenarios for the QAs included in the ASRs. The original scenarios described by the participants for each attribute are listed in Appendix E.

5.2.4 Implementation

Parallel to the updating of QAs and refinement of the second version of the reference architecture, we started developing a prototype that has been implemented following the reference architecture. The objective of this effort was mainly to use the development process itself as input to refine the definition of the reference architecture, as it allowed for more concrete definitions of interfaces and practical issues in connecting the modules proposed in the architecture.

We chose to update an existing open source puzzle game called Lix (Naarmann, 2011) and integrate some of the identified SG functionalities to it, namely assessment and learning analytics capabilities, and simple adaptation. By altering an existing game, we could focus on aspects of the implementation of the architecture, without the overhead of developing a whole game from scratch. Conclusions and lessons learned from the development process were taken into consideration in the second version of the reference architecture.

In addition to helping us elaborate the details of the reference architecture, the reference implementation also played a role in complementing the evaluation of the architecture. It served as a partial evaluation of the buildability of the reference architecture, using it to survey existing technologies that support the functionalities defined in the reference architecture and how much work would be needed to include external pieces in the architecture. We could also evaluate how useful the architecture is when trying to incorporate new educationally-relevant features to existing games.

The conclusions and lessons learned from the development of the example implementation are discussed in Section 5.5.

5.2.5 Evaluation

Since architectural design choices have a direct impact on the system to be designed, the evaluation of a software architecture (or reference architecture) is an important step to be performed before the beginning of any system development, to allow for timely discovery and resolution of potential problems (Angelov et al., 2008).

To evaluate the architecture, we followed a two-step approach. First, we evaluated the functional completeness of the SORASG, comparing the desired functions to the functions that the architecture supports. Then, we evaluated the quality of the architecture by employing an adapted ATAM evaluation (Angelov et al., 2008), which checks if the

reference architecture supports the desired quality attributes of the system. Additionally, we complemented the analytical evaluation of the architecture by examining the experience of building the example implementation of the SORASG.

The ATAM evaluation gives us a straightforward way to enlist the expected benefits and drawbacks of the reference architecture, comparing its characteristics to the requirements that the design should fulfill. This type of analysis is important because it can be done early in the development process, before any line of code is written. Since software development is expensive, the possibility of avoiding future errors is preferable and cost saving (Bass et al., 2012; Angelov et al., 2008). The procedure of the ATAM method is described in detail by Kazman et al. (2000). The adaptations suggested by Angelov et al. (2008) are described in Subsection 4.1.2.

A complete ATAM report includes a series of information that can be used to give us a complete picture of the architecture, its goals and the result of the evaluation (Kazman et al., 2000). In this thesis, the typical parts of an ATAM evaluation are presented in different parts of the document, namely:

- The business drivers of the architecture are described in Section 1.2;
- The description of the architecture is given in full in Section 5.4, and the summary of the architectural approaches is available in Subsection 5.4.2;
- The prioritized quality attribute tree is given in Subsection 5.3.3;
- The tables with the full analysis of the architectural approaches are available in Subsection 5.6.2;
- The summary of the results of the analysis is presented in Subsection 5.6.3.

The results of the ATAM evaluation and the lessons learned from the development of a reference implementation are described in Section 5.6. The characteristics of the game and the conclusions that were drawn from the experience of developing a reference implementation are discussed in Section 5.5.

Nevertheless, an analytical evaluation of a reference architecture can only go so far. Once the reference architecture is completed and published, more concrete evaluation metrics can be applied, particularly acceptance by the development community, quality of the resulting systems and the architecture's effects in the development process.

5.3 Requirements

As mentioned in Subsection 4.1.2, the design of software architectures (including reference architectures, which are a special type of software architectures) is highly dependent on well-defined architecturally significant requirement (ASR). These requirements serve as starting point for the design and as a reference for constant evaluations, as recommended by the ADD methodology.

In this chapter, I first provide a decomposition of the business domain that is based on the theoretical model proposed in this work (the ATMSG, see Section 3.2).

This decomposition is followed by an analysis that identifies the functional requirements that the reference architecture must support. In particular, I provide an analysis of the common characteristics of SGs, that is, characteristics that are shared between games of different domains and different learning topics.

Subsequently, I describe the list of QAs that are relevant for the reference architecture. The QAs have been extracted primarily from the business goals (see Section 1.2), but complemented with characteristics extracted from the domain analysis, and later validated and expanded from the consultations with the stakeholders.

5.3.1 Analysis of business domain

A crucial initial step to determine the functional requirements of a system is decomposing the business domain into its functional areas, subsystems, and desired goals, to identify processes and high-level business use cases. Then, these processes are selected as possible candidates for system functionalities or modules. This step is common among several methodologies for system development (Ning, 1996; Ricordel & Demazeau, 2000; Papazoglou & Heuvel, 2006; Erradi, Anand, & Kulkarni, 2006; Arsanjani et al., 2008). In this section, we, therefore, provide such an initial analysis of the business domain.

The motivation behind this analysis was to identify explicitly which learning elements exist independently of game genre and topic, and can be incorporated in a modular fashion into game architectures. In other words, we wanted to identify which elements would be useful in achieving the two main business goals identified previously, namely to reduce costs of SG development and to promote reusability. However, there is a legitimate concern that reusing generic software pieces in game development could result in generic solutions that do not meet the desired educational and entertainment goals. For this reason, it is also important that there is an explicit link between theory and practice. The ATMSG model and its associated taxonomy of game elements (see Chapter 3) served as this link, helping us reflect upon which SG components are relevant in a wide variety of situations.

The ATMSG taxonomy provides an overview of a large number of commonly found elements of SGs. Among these, we collected a number of relevant items for our purpose. The criteria for the selection were (a) relevance for the effectiveness of educational SGs, and (b) possibility of reuse across different games and learning domains, at least within the same game genre. These items are discussed below, grouped by the activities according to the ATMSG model.

Game activity

Game engines (such as Unity) have been successfully used to abstract gaming elements, for example to speed up the creation of 3D game worlds and characters. Nevertheless, even when game engines are used, games are typically still built as black boxes, that is, in a way that conceals the game's inner workings. On the one hand, this significantly reduces the complexity of game development; on the other hand, it hinders the possibility of implementing functionalities that are not directly available in the game engine. To be able to enhance games with external functionalities, it is thus necessary to create a way to expose what happens inside the game (in-game events such as players' actions, scores, achievements, level progression, and so on) to external modules in a reusable manner, translating those events into information that will be useful to these modules.

Because the majority of gaming elements are highly related to game genre and topic, we did not specify gaming elements as candidate functionalities, but instead we treated most of the gaming actions as events that can be listened by external modules. One exception: actions that are related to obtaining information are also game- and genre-independent, and thus they represent excellent candidates for abstraction.

Among gaming tools, elements that are independent of genre are those related to goal metrics and feedback on the goals, such as achievements, performance scores, and leaderboards. Moreover, in a related fashion, among gaming goals, the elements related to competition based on performance can be abstracted, particularly if existing social networks are used to connect players to their peers.

Learning activity

Learning activity elements are highly dependent on the context of the SG, particularly on game genre.

Dependency on the context is especially true in the case of learning actions. For example, actions such as memorizing, locating, classifying or assessing have particular implementations depending on both the topic and the game itself. Consequently, no reusable elements were selected from this list for inclusion as candidate services.

Among learning tools, however, some elements were considered general enough to be useful for a subset of game genres, even if they are not relevant to all kind of games at all times. In particular, surveys and questionnaires – similarly to knowledge bases – are tools that can be easily abstracted and implemented separately from the game. We also identified that stand-alone modules to store and display media (e.g. audio, video, pictures) can be developed. Finally, student diaries can also be implemented in a way that can be game- and topic-independent.

Lastly, among learning goals, the most genre- and topic- independent element is "learning how to learn", or self-reflection on learning. A module to provide students with an overview of their progress is another good candidate for a reusable service for SGs.

Instructional activity

Intrinsic instructional activity The following intrinsic instructional actions were selected as candidate services: scaffolding, repetition, show similar problems and supporting recovery from errors. These elements are related to adapting the level of challenge to the player's current capabilities, which is an important factor for the efficacy of educational mediums (Kickmeier-Rust & Albert, 2012b).

Among intrinsic instructional tools, we highlighted quantitative assessment of performance, either using simple metrics (e.g. goal achievement, scores) or more complex methods such as the Competence-based Knowledge Space Theory (CbKST) (Kickmeier-Rust & Albert, 2012b).

From the list of intrinsic instructional goals, several items were selected: presenting the stimulus, providing feedback, assessing the performance, fostering confidence and providing satisfaction to the player. The items above relate to in-game assessment, feedback, and automatic adjustment of instruction (adaptivity).

Extrinsic instructional activity Some elements in the extrinsic instruction list are similar to the ones already described in the intrinsic instructional activity. Here, however, the focus is on elements that a service-based reference architecture could still incorporate to support the instructor in assessing and giving feedback to the student via the game.

Qualitative assessment of performance is an action that is both important and potentially relevant across SG genres. It is the only item selected among the list of extrinsic instructional actions, since the other actions happen, by definition, outside of the game.

Among tools and goals, the selected items are also featured in the intrinsic instruction activity: performance measures, performance assessment, and feedback. These elements were selected as candidates because they can be used to inform instructors about learners' performance, then provide a way to incorporate the instructor's input back into the game.

In this section, I described our analysis of the educational SG domain, in which elements that were considered both relevant and with high potential for reuse were selected. Table 5.5 summarizes the elements according to the activity to which they belong.

5.3.2 Functional requirements

Once the relevant educational SG components described in the ATMSG taxonomy were identified and collected, we regrouped them according to their functional domains. In this way, we could identify the clusters of components particularly suited for reusability, which would form the base for the functional requirements of the reference architecture. The result of this grouping is shown in the column "Functional domains" of Table 5.5.

Activity	Element type	Items	Functional domains	FR
Gaming	Actions	Events listener	Game connectors	FR4
		Watch/Listen to/Read information	Information storage and retrieval	-
		Ask questions	Information storage and retrieval	-
	Tools	Social network score	Between-players interaction	-
		Leaderboards	Between-players interaction	-
	Goals	Competition	Between-players interaction	-
Learning	Actions	-	-	-
	Tools	Surveys, questionnaires	Student-instructor interaction	-
		Student diary	Student-instructor interaction	-
		Media assets (audios, films, graphics, etc.)	Information storage and retrieval	-
	Goals	Reflective observation	Feedback	FR1
		Learning how to learn	Feedback	FR1
Intrinsic instruction	Actions	Quantitative assessment	Assessment	FR2
		Scaffolding	Personalization and adaptivity	FR3
		Show similar problems	Personalization and adaptivity	FR3
		Support recovery from errors	Personalization and adaptivity	FR3
	Tools	Performance measurements	Assessment	FR2
	Goals	Assess performance	Assessment	FR2
		Provide feedback	Feedback	FR1
		Confidence	Assessment, Personalization and adaptivity	FR2, FR3
		Satisfaction	Assessment, Personalization and adaptivity	FR2, FR3
Extrinsic instruction	Actions	Qualitative assessment	Assessment	FR2
	Tools	Performance measures	Assessment	FR2
	Goals	Assess performance	Assessment	FR2
		Provide feedback	Feedback	FR1

Table 5.5: ATMSG elements grouped by functional domains, with their associated functional requirements (FR)

From the list of functional domains, we selected the functional requirements for the reference architecture, based on their potential applicability in different games of different domains and learning topics. The functional requirements are numbered FR1–FR6 for later reference, and further described below.

[FR1] Feedback Feedback functionalities provide a way to send assessment results to the player, to support his or her self-reflection on learning. The term feedback, as used here, refers to information explicitly disclosed to the player, directly inside the game or indirectly via other means. It can also be provided during gameplay, immediately after, or at a later stage (e.g. debriefing sessions, test results).

[FR2] Assessment Assessment functionalities can include modules for quantitative (automatic) and qualitative (instructor-provided) assessment, in addition to usage data that can help in identifying patterns of usage (learning analytics). It can also include modules for assessment of player/learner's engagement, confidence, and satisfaction.

[FR3] Personalization and adaptivity Adaptivity functionalities are responsible for enabling the game to respond differently to different players, according to their profiles or preferences (personalization), or to their in-game performance (adaptivity), as illustrated in Figure 5.4. Personalization occurs once, at the beginning of the game, and the user profile is updated at the end of a gaming session. Adaptivity occurs throughout the game, in a constant exchange of messages between the game and the module providing adaptivity functionalities.

Adaptivity is distinguished from Feedback in which the former is implicit, and often the player will not be consciously aware of the game's reactions to his or her performance while the latter is explicitly displayed to the player.

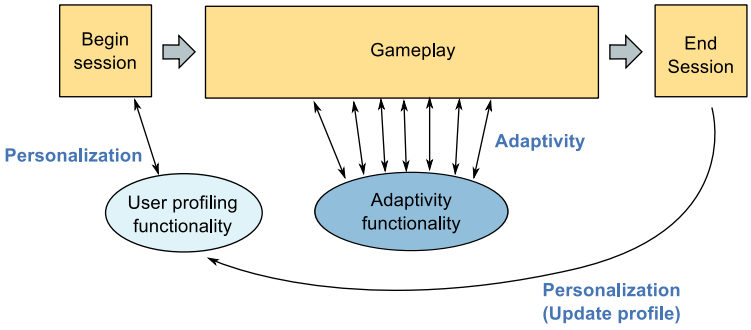


Figure 5.4: Distinction between game personalization and adaptivity

[FR4] Game connectors Game connectors provide adapter modules and data models that link external services to the game. These connectors are possibly a game engine plug-in responsible for implementing trigger managers that detect important

in-game events and forward messages to the other modules or services connected to the game. These connectors will most likely be game- or at least genre-specific.

[FR5] User profiling Although not directly derived from the taxonomy, a common user profile functionality is required to enable interaction, synchronization and persistent features across different games and learning settings. User profiling is also necessary to make it possible to personalize the game according to user's profile or preferences.

[FR6] Data logging A data logging module is responsible for collecting relevant game data from the game and exposing it to the other services, and, whenever needed, storing the data for further processing. This functional requirement was collected from stakeholders via an online questionnaire (see Appendix C).

As can be seen in Table 5.10, three other functional domains were identified as suited for reusability, but were not mapped as functional requirements. These domains, albeit generic, are not universally relevant for all games, nor homogeneous enough for inclusion in the scope of an overarching reference architecture for SGs. For this reason, we decided to not include them as functional requirements for the SORASG. These domains are described below.

Between-players interaction These are functionalities to collect, display and compare scores, such as social leaderboards. They do not include multiplayer capabilities inside the game.

Student-instructor interaction These are functionalities to allow instructors to query players/students, prompting for answers to questions or surveys, or for reflections in their learning process (student diaries).

Information storage and retrieval These are functionalities to allow the management and use of information about topics in the game (i.e. non-playing characters' knowledge about the game world) or about the learning domain itself. These functionalities can be especially useful when dealing with topics potentially relevant for different games or learning applications. If these functionalities include descriptive metadata, it can be much easier to implement resource discovery, and also to reuse those resources in different games. Functions that can connect to knowledge databases and convert their data to other formats are also included in this category (e.g. adapters for natural language interaction, automatic generators of question and answers, and so on).

5.3.3 Quality requirements

Previously, I listed the main functionalities that the reference architecture should contemplate. In this section, I describe the QAs, or non-functional requirements, that the reference architecture must meet.

In the software architecture life cycle, gathering QAs is an important step, and sometimes not a straightforward one. In many cases, the QAs will not be found in official, written requirement documents, but instead they must be uncovered through interviews with stakeholders and analysis of the organization's business goals (Bass et al., 2012).

In the case of the SORASG, the QAs have been identified from two sources: the architecture's business goals and from the feedback from stakeholders. From the three business goals (BG1–BG3) (described in Section 1.2), we extracted the related QAs. From the feedback from stakeholders (Subsection 5.2.3), we collected the highest ranked attributes and analyzed them according to their relevance to the SORASG. In the prioritization of the QAs, the business goals were given the highest importance, given that they reflect the overall objectives of the reference architecture and of this research. The attributes mentioned and ranked by the stakeholders were considered according to their relevance to different types of SGs. The results of this analysis are presented below. For each QA, a scenario that characterizes and further defines the attribute is provided.

[QA1] Development distributability Development distributability represents the ability of the software of supporting distributed software development. This attribute is elicited mainly by the goal "reduce development costs while maintaining quality" (BG1), because if parts of the game development can be reused from existing components, the cost associated with including those functionalities is reduced. At the same time, the quality of the implementation depends on how well the different pieces that have been developed by different teams, and sometimes with different purposes than those needed by the game, can connect to each other while still realizing the desired business requirements of the software. In our case, the business requirements are the learning and entertainment goals of the SG.

The goal "allow reuse of technological solutions" (BG2) also elicits the QA development distributability. As mentioned in Section 1.2, the access of up-to-date components, created and maintained by experts in the field, requires that the software be fit for development by independent groups. The architecture must support this structure to allow for reuse of parts that have been developed by completely diverse teams, often from different companies or project groups, and sometimes from completely different fields.

The scenarios that illustrate this attribute are:

- **[S1]** Different coding teams can work in different parts of the system at the same time, with minimal coordination.
- **[S2]** It is possible to reuse existing components in the game, even if developed by third parties.

[QA2] Modifiability Modifiability is the ability to change the software easily. The business goal of reducing development costs also elicits the QA modifiability, since it opens the possibility of recycling existing games into other games with similar game mechanics and pedagogical approaches but in different learning domains. It

also gives instructors some degree of flexibility to customize the game according to their needs. Furthermore, modifiability is also desirable when developing software in an iterative way, as being able to modify the software between the multiple iterations reduces the cost of development while keeping the quality.

Usability was the highest ranked attribute by the stakeholders we consulted. Nevertheless, we do not include usability directly as a QA for the reference architecture, since this requirement depends on the characteristics of each system. However, given that one of the techniques to design usable interfaces is testing several development iterations with users, we consider modifiability as a proxy requirement for usability, as modifiability facilitates evolutionary prototyping.

The scenarios that illustrate this attribute are:

- **[S3]** The system can be easily reconfigured for use with alternative components providing similar functionalities.
- **[S4]** Related games, with similar characteristics or content, can be created with minimal effort.
- **[S5]** The game can be changed to incorporate new content or tasks.

[QA3] Interoperability Interoperability accounts for the software's ability to exchange meaningful information via interfaces. The goal "allow reuse of technological solutions" (BG2) implies the QA interoperability, as it is obviously an attribute that is desirable when trying to reuse software developed by third parties.

The goal "promote the use of open standards and technology-independent solutions" (BG3) is also directly related to the QA interoperability.

Interoperability can be considered closely related to the QA mobility, which was ranked as a highly relevant attribute by the stakeholders. Interoperability is needed to support a number of distributed components running on different platforms in an SOA architecture. We do not include mobility directly as a QA in our list of ASR, as the SORASG is not meant to target specifically mobile platforms. However, as discussed in Subsection 4.3.1, the architectural pattern of SOA is considered one possible solution for accessing games from devices with limited computing power, and interoperability is a crucial attribute for SOA.

The scenarios that illustrate this attribute are:

- **[S6]** The game is capable of exchanging information with connected components, and the data is automatically interpreted.
- **[S7]** Game components can run on different platforms.
- **[S8]** Player's data is stored in an open standard format. The data can eventually be imported and interpreted by third-party tools (e.g. learning management systems (LMS)).

[QA4] Privacy and security Privacy and security issues were mentioned by participants as important points. Of particular notice were the implications of storing

user data in servers in different countries, which can also bring difficulties for developers to work in accordance with privacy laws of different countries.

While privacy and security are two different issues, we consider them together as, from the viewpoint of the user, the desired effect is the same: that user data is protected from unauthorized access.

The scenarios that illustrate this attribute are:

- **[S9]** Players' personal data is not disclosed to third parties. Storage and transfer of players' personal data are minimized.
- **[S10]** No personal data is disclosed without users' explicit consent. Users can revoke rights to their data at any time.

[QA5] Performance Performance was mentioned by group interview participants and questionnaire respondents as a relevant QA. Particularly for games, the system must be responsive enough as not to interfere with the gaming experience. Moreover, in the case of systems that make extensive use of network calls, it is important to make sure that the gaming experience is not negatively affected by eventual delays.

The scenario that illustrates this attribute is:

- **[S11]** The player's gaming experience is not affected by the software response times.

While the QA deployability was considered important by stakeholders (because installing games often needs to be done by users with little or no familiarity with computers, e.g. children, teachers, doctors), it is an attribute that is highly dependent on each software's requirements. As such, it was not considered general enough for inclusion as an attribute in the reference architecture.

Table 5.6 relates each QA to its source (business goal or feedback from stakeholders).

Utility tree and scenario prioritization

A utility tree is a visual representation of the desired attributes of a software architecture, as the expression of the "goodness" of a system (Kazman et al., 2000; Bass et al., 2012). It gives us a compact representation of the desired QAs, in a format that is easy to reference during the design and evaluation of the architecture.

As its name implies, the utility tree defines a tree structure, with levels that depict further refinements of the collected attributes of the system. The first level of the utility tree lists the relevant QAs. These QAs are quite generic, however, and must be further refined. The second level of the tree presents a decomposition of the attributes, refining the attributes into a more detailed description. Each refinement is then represented in the form of scenarios, which help concretize vague qualities into specific examples of

Quality attributes	Source of attribute
[QA1] Development distributability	- Business goal "Reduce costs while maintaining quality" - Business goal "Reuse of technological solutions"
[QA2] Modifiability	- Business goal "Reduce costs while maintaining quality" - Individual interviews - Questionnaires, to support evolutionary prototyping, which in turns supports usability
[QA3] Interoperability	- Business goal "Reuse of technological solutions" - Business goal "Open standards and technology-indepence" - Questionnaires, particularly to eventually support mobility
[QA4] Privacy and security	- Group interviews
[QA5] Performance	- Group interviews - Questionnaires

Table 5.6: Sources of quality attributes (QAs)

current and future use cases of a system (Kazman et al., 2000). Finally, each scenario is evaluated according to its business value and the potential impact on the architecture. Each scenario is assigned a priority level (H–High, M–Medium, L–Low), according to how important the attribute is to the success of the system, i.e. business value (P1), and the related risk for achieving the attribute, i.e. impact on the architecture (P2).

In the process of designing regular software architectures, scenario prioritization happens over working sessions involving stakeholders. However, as it was explained in Subsection 4.1.2, in the case of reference architectures, both the lack of a clearly defined group of stakeholders and the generic nature of the architecture make it very difficult to involve stakeholders directly in this phase. For this reason, the prioritization of scenarios and definition of the utility tree was performed by the researcher, based on the business goals and the feedback given by the stakeholders (Subsection 5.2.3).

Table 5.7 shows the utility tree of the SORASG, including the prioritization of the scenarios (columns P1 and P2).

As can be seen in Table 5.7, the highest priority scenarios from the business goals viewpoint (i.e. where column P1 is marked as "H") relate to development distributability, interoperability, security of user data and performance. Of these high priority qualities, three of them are also marked as high-risk decisions (i.e. column P2 is marked as "H"):

- The reuse of components developed by third parties is a high priority scenario since the main business goals of the SORASG refer to the possibility of reusing existing technological solutions. It has high impact on the development, as developers would depend on the quality of published interfaces from third parties. It is possible that extra work is needed to incorporate components if they do not

Quality attribute	Attribute refinement	Scenario	P1	P2
[QA1] Development distributability	Coordination among teams	[S1] Different coding teams can work at different parts of the system at the same time, with minimal coordination.	H	L
	Reuse of components	[S2] It is possible to reuse existing components in the game, even if developed by third parties.	H	H
[QA2] Modifiability	Service reconfiguration	[S3] The system can be easily reconfigured for use with alternative components providing similar functionalities.	L	H
	Variability	[S4] Related games, with similar characteristics or content, can be created with minimal effort.	M	M
	New educational requirements	[S5] The game can be changed in order to incorporate new content or tasks.	M	H
[QA3] Interoperability	Data exchange	[S6] The game is capable of exchanging information with connected components, and the data is automatically interpreted.	H	L
	Distributed components	[S7] Game components can run in different platforms.	H	L
	Use of standards	[S8] Player's data is stored in an open standard format. The data can eventually be imported and interpreted by third-party tools (e.g. LMS).	H	L
[QA4] Security	Personal data security	[S9] Players' personal data is not disclosed to third parties. Storage and transfer of players' personal data is minimized.	H	H
	User controls data	[S10] No personal data is disclosed without users' explicit consent. Users can revoke rights to their data at any time.	H	M
[QA5] Performance	Response time	[S11] The player's gaming experience is not affected by the software response time.	H	H

Table 5.7: Utility tree of the Service-Oriented Reference Architecture for Serious Games (SORASG)

conform to the existing code.
- Security of personal data is a high priority because this is a legal requirement in many countries. It is also high risk because the use of third-party components brings the possibility of using services that cannot be trusted.
- Game performance is important because one cannot disregard the main objective of the game: to provide a pleasant user experience. Nevertheless, this is a high-risk decision, since the use of third-party components, particularly when accessed through the network, brings the possibility of bad network connections and poor quality of service.

The three high-importance–high-risk scenarios discussed above are considered, then, the ASRs that deserve most consideration in the development and evaluation of the SORASG.

5.3.4 Constraints

As described in Section 1.2, one of the driving business goals of the reference architecture is to promote the use of open standards and technology-independent solutions. Consequently, one strong recommendation of the architecture is the use of existing open standards to represent data, whenever such standards exist. This recommendation is considered here as a constraint because it restricts our freedom of choice in defining the ways in which the modules can interact.

The Experience API (xAPI), formerly Tin Can API, is establishing itself as a standard for the learning community, used for collecting both formal and informal learning activities from different sources (Kevan & Ryan, 2016). It is considered the successor of Sharable Content Object Reference Model (SCORM), which has been the dominant standard in the e-Learning industry for the past decade (Foreman, 2013). The first version of the xAPI was released in 2013 (Advanced Distributed Learning, 2015), and it continues to be updated with the involvement of a community of developers and businesses. Given its growing adoption both by the industry (Rustici Software, 2016) and by the research community (Glahn, 2013; Megliola, De Vito, Sanguini, Wild, & Lefrere, 2014; Hruska, Long, Amburn, Kilcullen, & Poeppelman, 2014; Streicher & Roller, 2015; Qazdar, Cherkaoui, Er-Raha, & Mammass, 2015; Kevan & Ryan, 2016; Serrano-Laguna et al., 2017), the reference architecture should support data exchange using the xAPI standard.

5.4 Reference architecture

So far, I described the overall goals and requirements that drove the design of the SORASG. In this section, I present the reference architecture in its current version.

The reference architecture consists of a set of documents that, together, communicate

the design decisions that will serve as instructions for future implementation. This documentation is reproduced in the following pages, under the following headings:

Classification The characteristics of the SORASG and its classification according to a taxonomy of reference architectures (Subsection 4.1.1).

Architectural approaches The list of architectural approaches and patterns used in the SORASG.

Roles and use cases The actors that are expected to interact with a system (the SG) that was built according to the reference architecture and its use cases.

Architectural views The SORASG is a complex architecture that cannot be depicted by one single dimension. Instead of trying to describe it in one large diagram, we chose to select a number of different *views* that depict different aspects of the architecture, one at a time. In this way, we can abstract some aspects while focusing on others, making the representation simpler and easier to understand. The set of all the views, together, represent the whole in a more manageable way.

- **Layered modules view** This view documents the sets of responsibilities in the system, showing which modules implement the core functionalities of the SORASG (described in Subsection 5.3.2), and their combination into layers of responsibilities, according to the principles explained in Subsection 5.4.2.
- **Entities and relationships view** This view lists the main entities of the SORASG and their relationships. Since the reference architecture is abstract and models the relationships between the components at a high level, only interface classes have been modeled and included in this view.
- **Components-and-connectors view** This view depicts runtime entities, the components, and how they interact with each other via the interfaces described in the Entities and relationships view.
- **Behaviors view** Shows the sequences of interaction between the components over gameplay.

5.4.1 Classification

According to the characteristics and goals of the SORASG, it can be classified as a Type 5.1 reference architecture in the taxonomy defined by Angelov et al. (2012). Type 5.1 indicates an architecture that is a "preliminary, facilitation architecture designed to be implemented in multiple organizations". The dimensions and values used for the classification are summarized in Table 5.8.

The SORASG is a Type 5.1 architecture, and as such it has been defined with a high level of abstraction, in which it only specifies the functionalities of its elements very generally, leaving many choices for specific implementations open to the designers. The

Dimension	Value	Description
G1: Goal	Facilitation	The SORASG is an architecture that aims to facilitate the design of future SGs, by providing guidelines and inspiration for the design of systems. It does not aim to establish a standard for the design of SGs.
C1: Where	Multiple organizations	It is aimed at multiple organizations (and not restricted to a single organization).
C2: Who	Research center	It was created in a research environment, and not by any organization in particular.
C3: When	Preliminary	It is a preliminary architecture, because it suggests how future SGs could be structured to achieve the benefits proposed by the architecture. It does not aim to represent a class of existing games – in fact, there are very few examples of games using Service-Oriented Architectures.

Table 5.8: Dimensions and values used for the classification of the SORASG

SORASG includes definitions of its *components and connectors*, and of the *aggregated protocols and interfaces* that represent the general lines of communication between the components. Furthermore, it is described using *semi-formal notation* using Unified Modeling Language (UML) 2.5 diagrams (Object Management Group, 2015), which allows for an unambiguous but still abstract representation of the components and their relationships.

5.4.2 Architectural approaches

The following architectural approaches have been employed in the SORASG, with the objective of achieving its desired QAs (described in Section 5.3). The rationale for the choice for each approach is explained below. The approaches are numbered A1–A7 for later reference.

[A1] Modularization and layers Any complex software benefits from modularization, in which separation of concerns is employed to enable different parts of a system to be developed and evolve independently. Modularization facilitates not only development but also maintenance and portability. In addition to the separation in modules, the SORASG follows a layered pattern, in which modules are grouped into units called layers. Layers establish a cohesion between the modules by creating constraints in the relationships between the layers, particularly defining unidirectional allowed-to-use relationships and enforcing strict ordering relations (Bass et al., 2012). Typically, modules from a layer are only allowed to use modules from a lower adjacent layer, and always through exposed (public) interfaces. This pattern tries to support extra portability since changes in one layer will only affect the next layer and not the rest of the system.

The SORASG implements a layered structure based on a simplified version of

the Open Group SOA Reference Architecture (see Subsection 4.3.2). The rationale for the choice of layers in the architecture is presented in more detail in Subsection 5.4.4.

[A2] Service orientation The choice for an SOA approach aims to give the SORASG the benefit of well-defined interfaces as a way to manage software complexity. It lets us treat multiple different and highly specialized, "standalone" functionalities as black boxes. A more thorough discussion of the benefits and drawbacks of a SOA approach to SG development is available in Subsection 4.3.1.

[A3] Choreography The SORASG implements service choreography instead of orchestration (see Subsection 4.3.2 and Figure 4.3), resulting in a software architecture in which the communications between modules are pre-configured (at design time or configuration time, but not at runtime). Note that the choice for choreography differs from the Open Group SOA Reference Architecture, which explicitly depicts the business layer as an orchestration layer (Subsection 4.3.2). We chose to use choreography instead because a decentralized approach has a lower impact on performance; is considered a more decoupled and flexible solution than an orchestrated one; and is more suited to the organization of services that employ asynchronous communications (Newman, 2015).

[A4] Asynchronous messaging The communications between the components, and particularly between the game and the other components, are performed mostly via asynchronous messaging (i.e. "fire-and-forget" information exchange). This characteristic reduces the impact of slow networks in game performance. Furthermore, the game might need to use multiple threads, to make sure that sending game events to other components does not block the game interface. Time-sensitive messages (game events and adaptation responses) include timestamps to allow the game to synchronize its responses, making sure that the suggested adaptations are still valid (e.g. establishing a maximum acceptable delay for a hint or suggestion to be considered by the game).

[A5] Limitation of exposure of user data The SORASG uses session IDs to establish communication and data exchange between components, instead of identifiers connected to the user's personal identity. The only service that contains information that links a session ID to a user ID is the User profile service. The objective is to protect users' personal data.

[A6] Explicit authorization Through a user interface, the player/learner authorizes which other components have access to his or her data, to ensure that explicit permission is given to all parties accessing user data. To reduce impact on the gaming experience, authorizations should happen in the beginning of the game and be valid for a certain duration (defined by the developer). Authorizations are always time-limited and can be revoked at any time.

[A7] Use of xAPI specification xAPI is a standard for packaging and transmitting the learner's actions in the form of "Activity Statements" that consist of a minimum of

three properties: “Actor”, “Verb” and “Object” (Advanced Distributed Learning, 2015; Kevan & Ryan, 2016). The specification is designed to support both formal and informal distributed learning (i.e. learning can happen at different places, not only in one single formal setting) (Foreman, 2013). As such, the standard can be used to represent activities carried out inside a SG in a way that is syntactically and semantically compatible across games and other learning tools. Furthermore, the specification is extensible, so that it can account for unforeseen data collection needs (Kevan & Ryan, 2016).

[A8] Use of configuration files Configuration files offer a compact way for the developer to supply a large amount of required instructions to the software at once, minimizing downtime in case of changes.

5.4.3 Roles and use cases

The SORASG defines three roles of actors who interact with the system, and the use cases that characterize this interaction (Figure 5.5).

The *user* (the player/learner) is the person who plays the serious game. One or more users can interact with the system at the same time (but not with each other, as this version of the reference architecture does apply to multiplayer games).

The *instructor* oversees the learning session, both in real-time and after the game session, through the dashboard and consolidated reports.

The game *developer* provides the files that configure relevant aspects of the interactions between the components of the game (e.g. defining the mappings between game and learning elements, defining a knowledge map for the game, defining an intervention model).

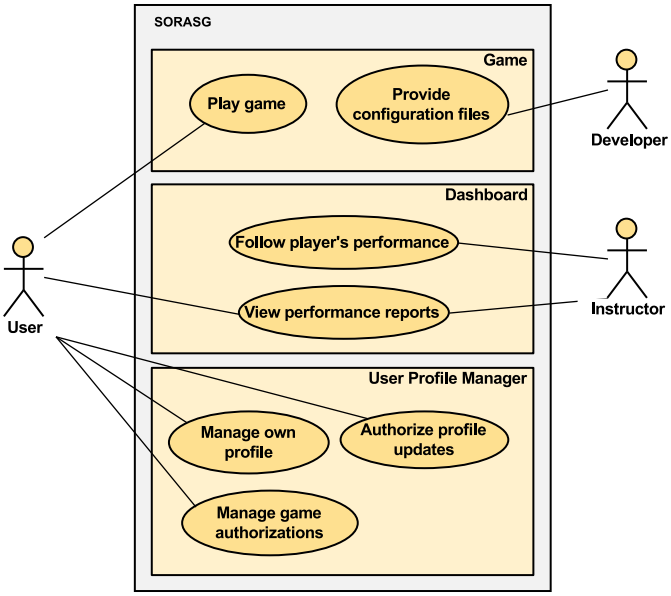


Figure 5.5: Roles and use cases of the SORASG

5.4.4 Architectural views

Layered modules views

A module is the static representation of an implementation unit in the system (Bass et al., 2012). The modules view presented in this section represent the abstract units that compose the system and how they are organized. It also depicts the general characteristics of their interaction. Note that the physical architecture of the implementation of these modules (i.e. the actual machines where the modules would run) is not necessarily reflected in this view.

Figure 5.6 shows the SORASG modules and their dependencies (dashed arrows), and how the modules are organized into layers. These modules largely reflect the functionalities identified previously (in Section 5.3 and Table 5.10). As mentioned earlier, this structure is based on the Open Group SOA Reference Architecture (Subsection 4.3.2). We implement the layers *Consumer interface*, *Services* and *Service components* (Figure 5.6). The Consumer interface layer depends on the Services layer, and each service in the Services layer depends on specific components from the Service components layer. The layer *Operational System* is the infrastructure layer supporting the Service components layer, but it is not referred explicitly in this documentation. For performance reasons, the *Business process* layer is not implemented, being replaced by service choreography (see the Behaviours view), as explained in the architectural approach A3 (Subsection 5.4.2).

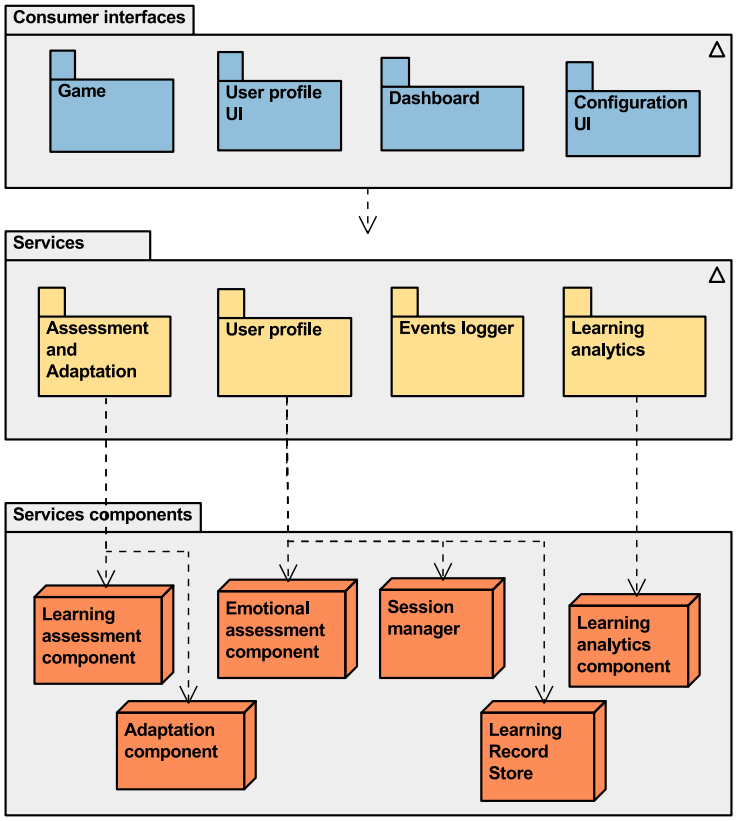


Figure 5.6: Layered representation of the modules and their dependencies

Furthermore, the cross-cutting layers (*Integration*, *Quality of Service*, *Information*, and *Governance*) are not included in the SORASG. This is because while the extra functionalities added by these layers are important for enterprise-level systems, they are not as relevant at the level of one single game, and could affect negatively game performance.

The Consumer interfaces layer holds the modules that have direct interaction with the users (green modules in Figure 5.6). The *Game* and the *User Profile UI* are the main points of interaction for the user. The *Dashboard* is the module that gives the instructor access to real-time monitoring of a gaming session; it also gives access to consolidated reports, which can be made available to players to enable self-reflection on his or her learning process. The *Configuration UI* gives the developer an access point to provide important configuration files for the system.

The SORASG defines four modules in the Services layer (see Figure 5.6). These are described below.

The *Assessment and adaptation* module is responsible for interpreting data coming from the game – and potentially from other sources, such as sensors attached to the player – with the objective of assessing player's performance and providing recommendations for implementing adaptivity in the game. The Assessment and adaptation module is realized by the *Learning assessment* and *Adaptation* components.

The *User profile* module holds persistent information about the user, including his or her emotional state (when relevant). This information is not game-dependent; on the contrary, it can potentially be used across different games and even other types of learning technologies (i.e. non-games). It is connected to the *User profile UI* as the point of interaction for the users, particularly for them to manage access to their personal data by other modules. Also, it provides profile data for the game at the start of a gaming session, so that the game can be personalized (i.e. settings changed to the player's preferences before the game starts). Finally, in the cases when adaptation includes not only gaming data but also emotional or engagement assessment, the User Profile can hold data about the User's emotional state.

The *Events logger* module collects raw game events from the game and processes them to generate xAPI statements, which record experiential data in the form of "player did this", or more generally "actor verb object" (see Section 2.2). These statements are forwarded to the other modules for further processing.

The *Learning analytics* module is responsible for collecting, processing and producing visualizations of user's data collected during gameplay. While the Assessment and adaptation module provides real-time assessment so that the game can perform adaptations, the Learning analytics module is responsible for the analysis of data after the gaming session, possibly aggregating the performance in a session with data from other players. The learning analytics module produces report data that is displayed on the dashboard to the instructor. The system may also give the user access to the dashboard to enable richer learning feedback and formative assessment.

The modules in the Service components layer (blue components in Figure 5.6) are the backend of the services modules, supplying the actual functionalities that these services provide. For example, the User profile module relies on two different Service components to provide its functionalities to the game: a *Session manager* to generate and manage access to gaming sessions, and a *Learning Record Store (LRS)*, which holds the xAPI statements associated with a user profile. For the game developer, it is not important which implementation of a LRS is used, since the game never interacts with the LRS directly. The service provider can switch to a different LRS and the game developer does not need to change the game – as long as the service's published interfaces are not altered.

Entities and relationships view

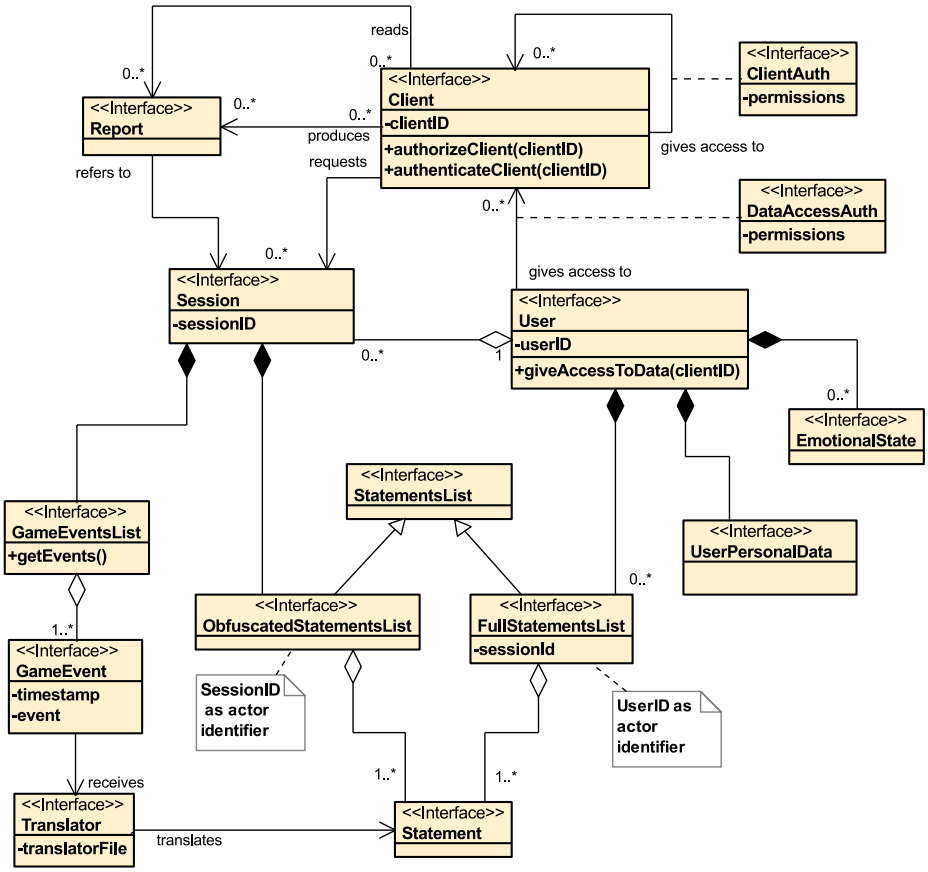


Figure 5.7: Structural model of the entities and relationships of the architecture

The structural model of the SORASG describes the entities of the architecture and their

relationships. They are the data entities created, read, updated and deleted by the components in their collaboration during configuration and gameplay. These entities are illustrated in Figure 5.7.

A *Client* refers to a running system component that establishes communication with other components or with the *User*. Each Client has a unique ID, and it can authenticate (verify the identity) and authorize (verify access permissions) other Clients. These authorizations are established using the interface *ClientAuth*, which is the associative class between Clients that defines the read-write permissions established between them. Authorization to read-write user data is given directly by the User to each Client that requests it, via the *DataAccessAuth* interface.

Some Clients produce *Reports*, which can be read by other Clients who are authorized to do so.

At the beginning of a gaming session, the game (Client) must request a *Session*, which has a unique *Session ID*. This Session ID is related to one User. A single User might have multiple Sessions associated with his or her profile although only the *User Profile* component knows who is the User associated with a Session ID.

A Session has a *GameEventsList*, which aggregates multiple *GameEvents*. Each GameEvent includes a timestamp, in addition to the representation of the event itself.

Each GameEvent is translated into a *Statement* (an xAPI statement, see Section 2.2) by the *Translator* file, which is provided by the Developer at the time that the system is set up.

The Statements are stored and exchanged between Clients in a *StatementsList*. There are two types of StatementsLists: *ObfuscatedStatementsList* and *FullStatementsList*. The User Profile includes the FullStatementsList. This version of the list uses the *User ID* as the actor identifier in each statement, as defined in the xAPI specification. Each User can have multiple StatementsLists, one per each Session. The full list has an attribute that indicates to which Session ID the statement belongs. The ObfuscatedStatementsList, conversely, is aggregated by a Session. It does not include the User ID – instead, it uses the Session ID as the main identifier for the actor. The ObfuscatedStatementsList is the version of the StatementsList that is exchanged between the Clients during gameplay.

A User has the associated *UserPersonalData* entity, which includes profile data such as sex, age, personality traits, learning preferences, and so on. The User entity can also include records of transitional *EmotionalStates*, when relevant in the system.

Component-and-connectors view

While the layered modules view documents the static distribution of the system in logical modules, the components-and-connectors view documents components. Components are autonomous, modular runtime units with well-defined provided and/or required

interfaces. They are replaceable within their environment, and can be reused. Their internal workings are hidden and inaccessible to other components, except to those functionalities exposed by its interfaces (Object Management Group, 2015). The modules of a system are mapped to their runtime components; sometimes the mapping is one-to-one, but often one module can correspond to many components, or a single component can be mapped to many modules.

In the SORASG, the modules listed in Subsection 5.4.4 are mapped in a straightforward way, with most of the modules corresponding to a component with the same responsibilities. Nevertheless, to facilitate visualizing the relationship between the module and the components, the figures in this section use the same color coding as depicted in Figure 5.6. Components that are related to modules in the Consumer interfaces layer are pictured in green; components that are related to modules in the Services layer are yellow, and the components that correspond to the Service components layer are shown in blue.

To facilitate the understanding of how the components interact with each other, these interactions are decomposed in many parts below. First, I present the authentication and authorization interactions, which characterize how the components establish their initial communications and how they perform their requests to the user to gain access to his or her personal data, and how the use of a session manager can protect user data. Subsequently, I focus on one system component at a time, presenting its ports and interfaces and depicting the interactions with other components. In the figures that follow, the components of main interest are presented on the left side, in a color that matches its representation in the layered modules view (depicted Figure 5.6); the components on the right are components that establish direct communication with the component of main interest, and are depicted in gray. In any figure, only the ports that are of current interest are shown; the other ports are hidden to increase the readability of the diagrams.

Authentication, authorization and session management In the SORASG, sometimes components must authenticate and authorize each other, and sometimes a user must authorize a component. For the purpose of authentication and authorization, we use the term *Client* to denote a running system component that establishes communication with other components or with the User. A more formal description of the entities depicted in the SORASG and their relationship can be found in Subsection 5.4.4.

There are two levels of authentications and authorizations in the SORASG. One is the Client authentication and authorization, in which a Client must be able to verify other Clients' identities and access rights. This process will most often happen at design time, possibly hard-coded or via configuration files. Nevertheless, it is also possible to implement runtime authorization, binding the services using service brokers and contracts that allow a Client to choose another service and authenticate without any human intervention. All components in the interface must implement the authentication interface via the *clientAuth* port as shown in Figure 5.8. The SORASG leaves open to each service developers the choice as to which authentication protocol to implement in each

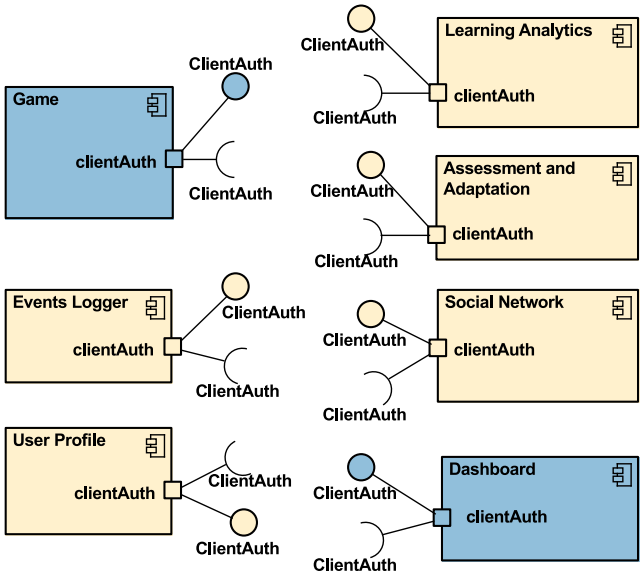


Figure 5.8: Components and ports involved in Client authentication and authorization

case. For example, one possible mechanism of authentication and authorization is the simple authentication method specified by the Hypertext Transfer Protocol (HTTP) protocol; another option is the OAuth2 protocol, which establishes the exchange of an authentication token that is submitted with every request to a service.

The second level of authorization is given by a User to a Client, using the interface *DataAccessAuth* (Figure 5.9). It refers to the access to the User's personal data. Components that need to access user profile data (username, age, sex, personality, preferences, or game interaction records) need to ask for users' permission. Permission is given through the *provAccess* port, which requires explicit user intervention through the User Profile UI component. The DataAccessAuth interface establishes an explicit expiration date (i.e. no indefinite time authorizations should be possible), and it can be revoked at any time by the user.

At the beginning of a gaming session, the Game sends a "Start game" event to all the connected components. These components then request a Session from the User Profile (Figure 5.10). All subsequent communications between components use a Session as identifier, instead of a User ID or any information identifying the User. The only component that stores the mapping between a User and its Sessions is the User Profile component. The User Profile component holds the information about which Clients are authorized to access user data, and which clients are authorized to know the User who created a given Session. This way, clients hold data associated with a Session only, and that data is not directly associated back with the user.

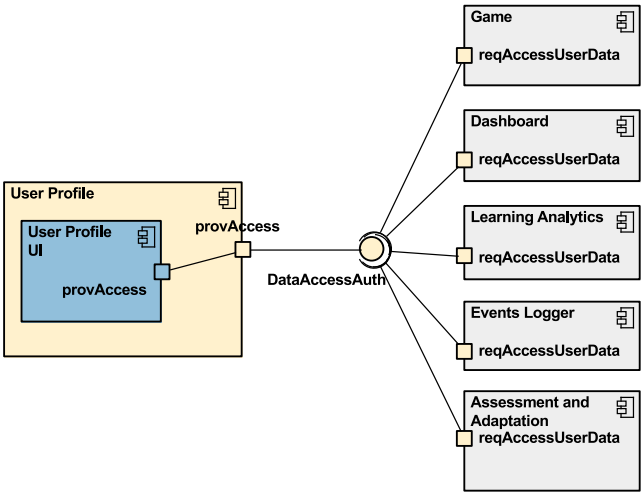


Figure 5.9: Components and ports that request and provide access to User data

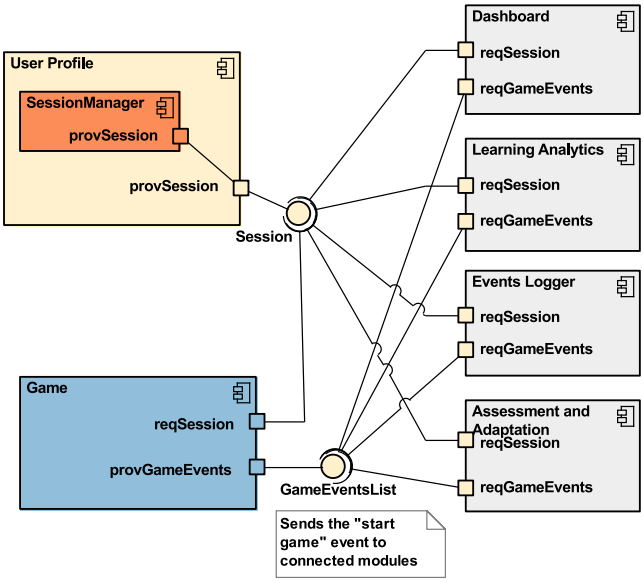


Figure 5.10: Components and ports that request and provide Sessions

For simplicity, the ports discussed above (clientAuth, provAccess, reqAccessUserData, provSession and reqSession) will be omitted in the subsequent diagrams.

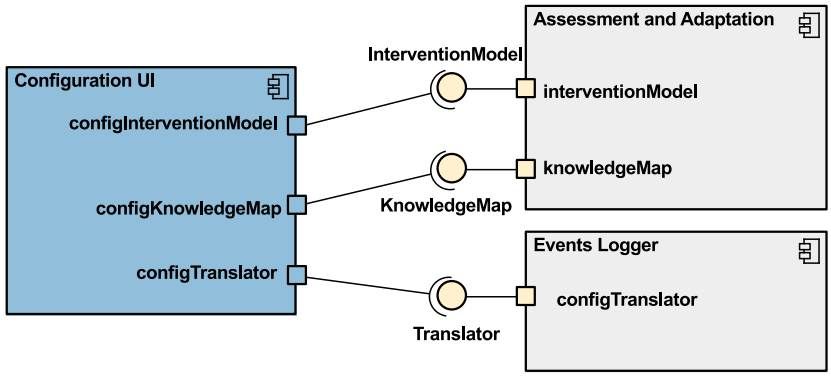


Figure 5.11: Components and ports involved in system configuration

Configuration During the setup of the whole system, it is necessary that the developer provide three important configuration files to the system. The Configuration UI component provides a graphical user interface for the developer where the configuration files can be uploaded and edited. The Configuration UI component then serves these files to the components via the ports described below.

Via the *configTraslator* port, the Configuration UI serves the Translator file to the Events Logger component. The Translator is used by the Events Logger component to convert raw game events into the xAPI statements that can be used by the other components.

The Configuration UI sends two configuration files to the Assessment and Adaptation component, the *knowledgeMap* and the *interventionModel*. Via the *configKnowledgeMap* port, the Configuration UI sends the structure of the knowledgeMap to the Assessment and Adaptation module. The knowledgeMap consists of a tree structure of xAPI statements, establishing a pre-requisite relationship between them. This structure allows the Assessment and Adaptation module to compare the current performance of the player to the structure of the domain. The interventionModel, sent via the *configInterventionModel* port, conversely, provides the rules for intervening during gameplay with hints and/or suggestions of activities based on the completion of certain xAPI statements.

Game Games are treated largely as black boxes with ports that expose game data or request information from the other components (Figure 5.12).

Through the *provGameEvents* port, the game exposes raw events to the rest of the architecture. The events to be exposed can vary according to the game. The recommended events are the universal traces proposed by Serrano-Laguna, Torrente, Moreno-Ger, and Fernández-Manjón (2014): game traces (start, quit, finish), phase traces (phase start,

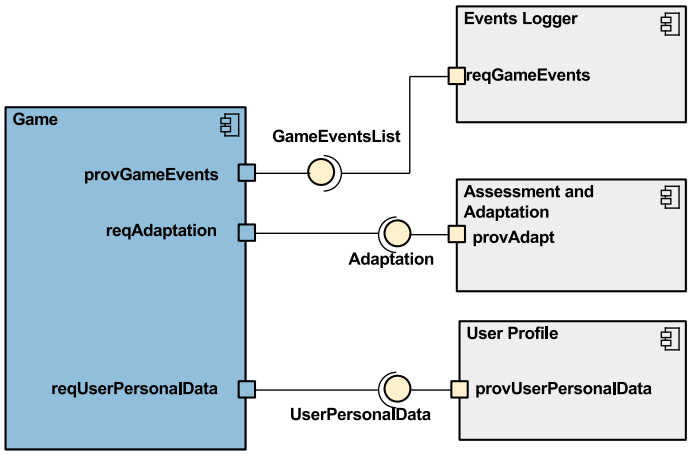


Figure 5.12: Components and ports for interaction with the game

phase end, completion status), meaningful variables (on performance and games state, such as scores, number of attempts, etc) and input traces (input action, input device, and associated input data). In addition to universal traces, other variables related to player performance and interaction with the game may be recorded and exposed. The raw game events always include a timestamp, which can be used to ensure that the components do not mistakenly react to outdated information.

The *reqAdaptation* port allows the game to receive processed data from the assessment and adaptation service, so that it can adjust its characteristics during a gaming session. As is the case with game events, the adaptation messages also include timestamps to allow the game to synchronize its responses.

Finally, the *reqUserPersonalData* port is used by the game to receive personal data from the User Profile, allowing it to personalize the game according to the player's characteristics (if relevant).

Events logger The Events Logger component is responsible for collecting raw game events from the game and translating them into xAPI statements. The standardized representation of game events in the xAPI format can be used by the other components when dealing with game data, without the need for game-specific configurations.

As explained earlier, the developer needs to provide a Translator mapping file to the Events Logger component. This configuration file is received by the Events Logger component via the *configTranslator* port, which is connected to the Configuration UI. The Events Logger component receives raw game events from the Game via the *reqGameEvents* port. It then provides statements list via the *provStatements* port to the User Profile component, which stores the statements in an LRS. The statements lists

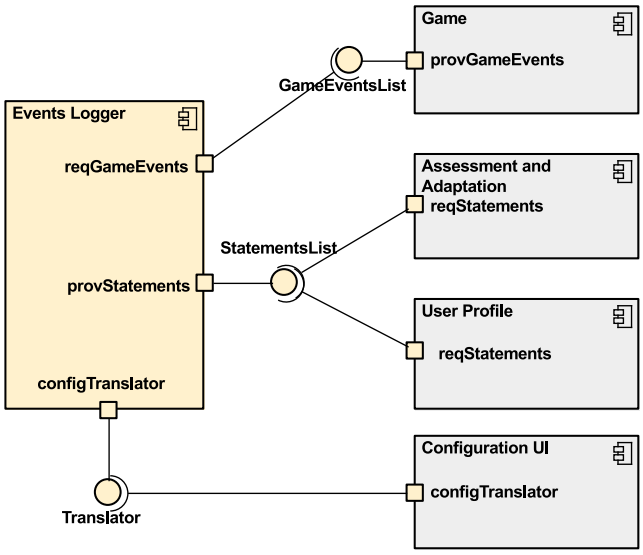


Figure 5.13: Components that interact with the Events Logger component

are also provided via the *provStatements* port directly to the Assessment and Adaptation module for real-time processing. These ports and connections are illustrated in Figure 5.13.

User profile The User Profile component is of central importance to the SORASG. In addition to managing gaming sessions and permissions (explained earlier in this Section), the User Profile component holds a centralized store of user profile data and a collection of past interactions with games in an LRS. If emotional or engagement assessment is included in the system, the User Profile component also holds the User's emotional data.

Figure 5.14 illustrates the ports and connections of the User Profile component. Via the *provUserPersonalData* port, it serves information about the User Profile (e.g. age, sex, playing preferences) to the Game and the components Assessment and Adaptation, Dashboard, and Learning Analytics, after confirming that they have the authorization to read the data. Via the *reqStatements* port, the User Profile component receives xAPI statements, converted from raw game traces by the Events Logger component. These statements are stored in an LRS, and provided to the components Dashboard and Learning Analytics via the *provStatements* port.

If the developer wishes to implement emotional monitoring in the system, the *reqEmotionalState* port can be connected to an external component that provides this data, for example measures of physiological signals for detecting players' affective states (see Subsection 2.2.2). The User Profile component receive store and subsequently pro-

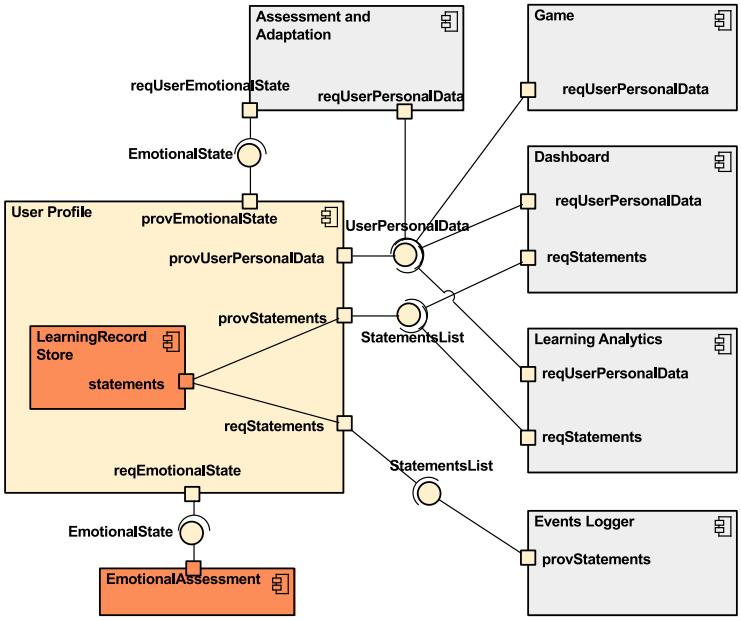


Figure 5.14: Components that interact with the User Profile component

vide emotional data to the Assessment and Adaptation component via the port *provEmotionalState*.

Assessment and Adaptation The Assessment and Adaptation component is responsible for processing game data in real time and returning recommendations back to the game so that it can perform adaptations to suit better the User's current performance.

During system configuration, the Assessment and Adaptation component needs to be configured with the Knowledge Map and the Intervention Model, which provide the service with game-specific knowledge about how to interpret and react to game events. This configuration is provided via the *knowledgeMap* and *interventionModel* ports, and these configuration files are stored in the component for each game that it serves. The Assessment and Adaptation component needs to collect the User's personal data at the beginning of a gaming session, received via the *reqUserPersonalData* port, so that the game's adaptation responses can also incorporate general characteristics of the user (e.g. gaming preferences or learning styles). The ports and connections are illustrated in Figure 5.15.

During gameplay, a constant stream of xAPI statements (processed game events) is received by the *reqStatements* port and constantly processed according to the knowledgeMap and the interventionModel to produce adaptation suggestions that are sent back to the game via the *provAdapt* port. We call this the adaptation cycle (highlighted in

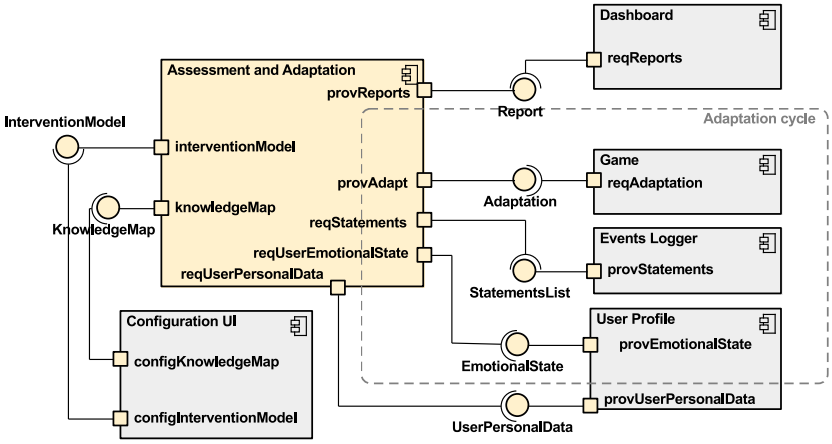


Figure 5.15: Components that interact with the Assessment and Adaptation component

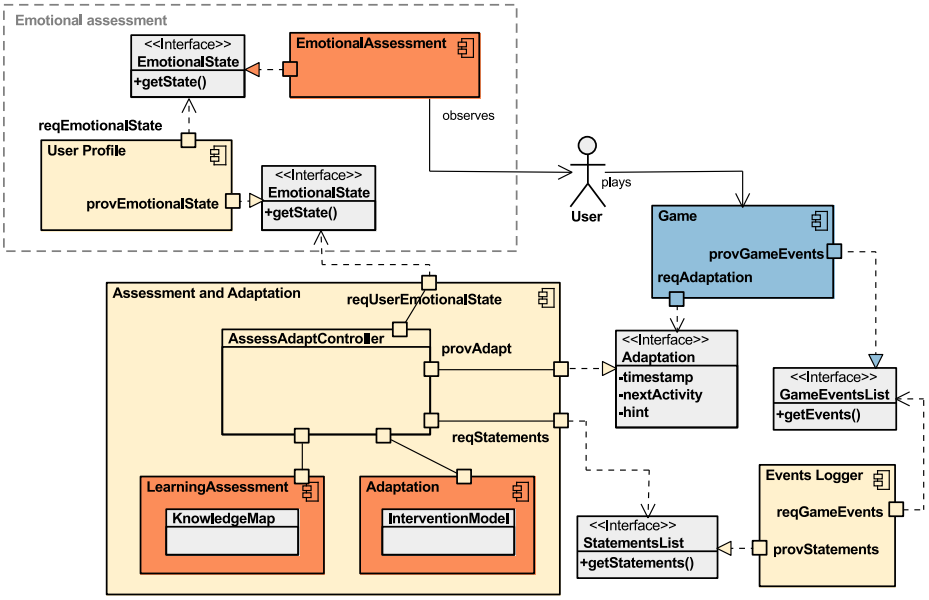


Figure 5.16: The assessment and adaptation cycle

Figure 5.15 and illustrated in more detail in Figure 5.16). The adaptations suggested by the Assessment and Adaptation component can have the format of hints or indications of which activities should be presented next to the User, and they include a timestamp so that the Game can evaluate the timeliness of the suggestions. The User plays the game, which is connected to the Events Logger via the *GameEventsList* interface. The Events Logger sends the processed statements via the *StatementsList* interface to the Assessment and Adaptation component. The Assessment and Adaptation component sends the statements to the *LearningAssessment* component, so that it makes inferences about the current knowledge of the user. It also sends the statements to the *Adaptation* component, for it to generate suggestions or hints to send back to the game via the adaptation interface. If the system implements the emotional assessment (e.g. by means of collecting physiological signals as described in Subsection 2.2.2), the *EmotionalAssessment* component observes the user while he or she plays the game, and sends the observations to the User Profile component via the *EmotionalState* interface. The User Profile sends the data to the Assessment and Adaptation component, which directs the emotional data to the Adaptation component so that the current emotional state of the User is also taken into consideration when producing the interventions.

Finally, the Assessment and Adaptation component is also responsible for producing reports on the progress of the User, by comparing the User's performance to the *KnowledgeMap*. These reports are sent to the Dashboard via the *provReports* port.

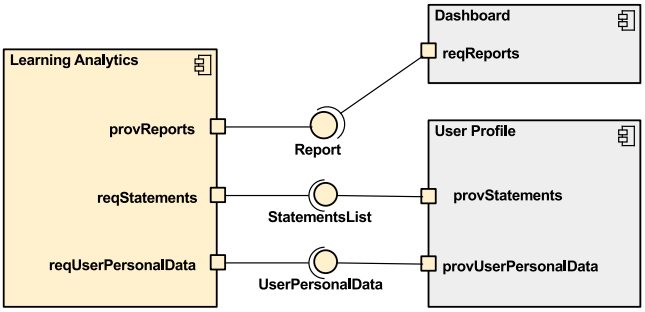


Figure 5.17: Components that interact with the Learning Analytics component

Learning analytics The Learning Analytics module uses general user profile data and the logged data from the user's activity during a gaming session to analyze game data. It consolidates data from one or more gaming sessions, from a single user or from multiple users, depending on what is provided by the User Profile module. Since it does not have access to User IDs, it can provide individual or consolidated reports without having to have access to identifying user data. When requested by the Dashboard, it sends reports that can be consulted by the Instructor or by the User.

The Learning Analytics module receives personal data from the *reqUserPersonalData* port and xAPI statements from the User Profile module via the *reqStatements* port. In

this manner, it does not need to deal with raw game data – which would require extra configuration for processing different types of raw game events. It serves the report data to the Dashboard via the *provReports* port. These ports and connections are illustrated in Figure 5.17.

The Learning Analytics module receives the statements from the User Profile module, and not from the Events Logger module. Since the Learning Analytics module does not need to feed data back to the game, it has less strict requirements regarding eventual network delays. A connection made directly to the User Profile module has the benefit of allowing for easier control of the data that is provided to the Learning Analytics module. It also makes it possible that data from previous gaming sessions is also analyzed, giving a more complete picture of the User's interactions with the game.

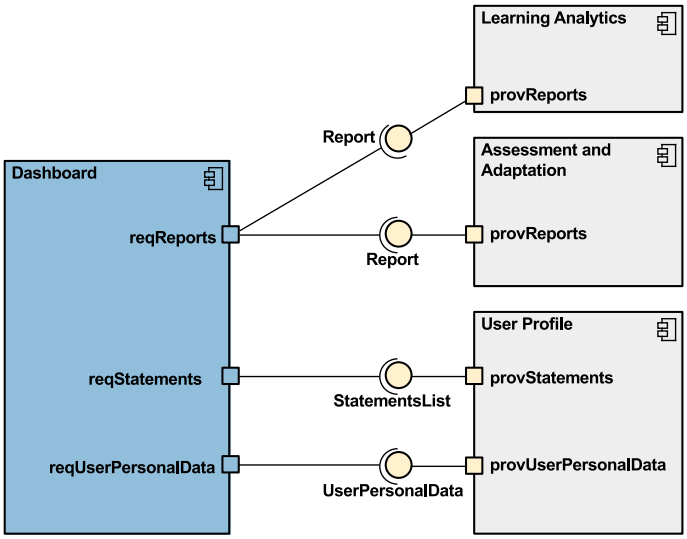


Figure 5.18: Components that interact with the Dashboard component

Dashboard The Dashboard component offers a graphical interface through which the Instructor can follow the User's performance in the game. It tracks the activities performed in the game (as xAPI statements) in a stream so that the Instructor can see in real time what the player is doing, and it gives access to consolidated reports generated by the Assessment and Adaptation and the Learning Analytics components. The consolidated reports can be accessed after a gaming session has been finalized, and potentially they can be made available also to the User, to give him or her the possibility to reflect on his or her performance.

When the Instructor consults the information displayed in the Dashboard, he or she will likely need to have access to the User identity, i.e. the Dashboard component needs to ask the User Profile for the User ID associated with a given Session ID. If the User

allows it, the Dashboard is able to make the connection between a Session ID and a User ID, disclosing the User's identity. This connection should not be stored permanently in the Dashboard, but only kept for the duration that the Instructor needs to have access to that information. This measure is to reduce the risk of unauthorized access to user's data.

The Dashboard makes the request to the User Profile component for the User ID associated with a given Session ID via the *reqUserPersonalData* port. It also receives the stream of xAPI statements from the User Profile component via the port *reqStatements*. The consolidated reports are received from the Assessment and Adaptation and the Learning Analytics modules through the *reqReports* port. These ports and connections are illustrated in Figure 5.18.

Behaviours view

While previously I depicted how the components are connected to each other in the system, in this section I show the process of the interaction during gameplay using UML 2.5 sequence diagrams. To make the model easier to read, I separate the process of collaboration in three parts: the process of initializing game, the process during gameplay, and the process of finalizing the game.

When a User starts the Game, the Game requests a SessionID to the User Profile component. The User Profile responds with a SessionID, which is used throughout the whole interaction between the components. When relevant, a MacroAdaptation can happen at this point. The Game can request User data from the User Profile (e.g. sex, age, learning preferences, personality traits). The User Profile requests the User to authorize sharing their data. If the user gives authorization, the User Profile sends the data to the game, which can make adjustments before the actual playing starts. The Game sends a notification indicating that the game started to the relevant components (Events Logger, Assessment and Adaptation, Learning Analytics and Dashboard). Each component requests a Session ID from the User Profile. In each case, the User Profile requests from the User authorization to share gaming data. The Assessment and Adaptation component also asks for User's profile data, in addition to the Session ID, since such data can be used in the Adaptation recommendations as well. These authorizations should happen only once in the beginning of the game, and should be valid for enough time so that a gaming session is not interrupted. Furthermore, the user interface of the User Profile component could request the User for all authorizations at once, to make it less cumbersome and user-friendly. The starting sequence can be seen in Figure 5.19.

Once the game has initialized, the gameplay can start. The process depicted in Figure 5.20 happens repeatedly until gameplay ends. Each time the player performs a relevant action in the game, the Game sends a Game Event to the Events Logger component. The event is translated into an xAPI statement and sent to both the Assessment and Adaptation and the User Profile components. In the Assessment and Adaptation component, the statement is processed, and an Adaptation suggestion is generated and sent

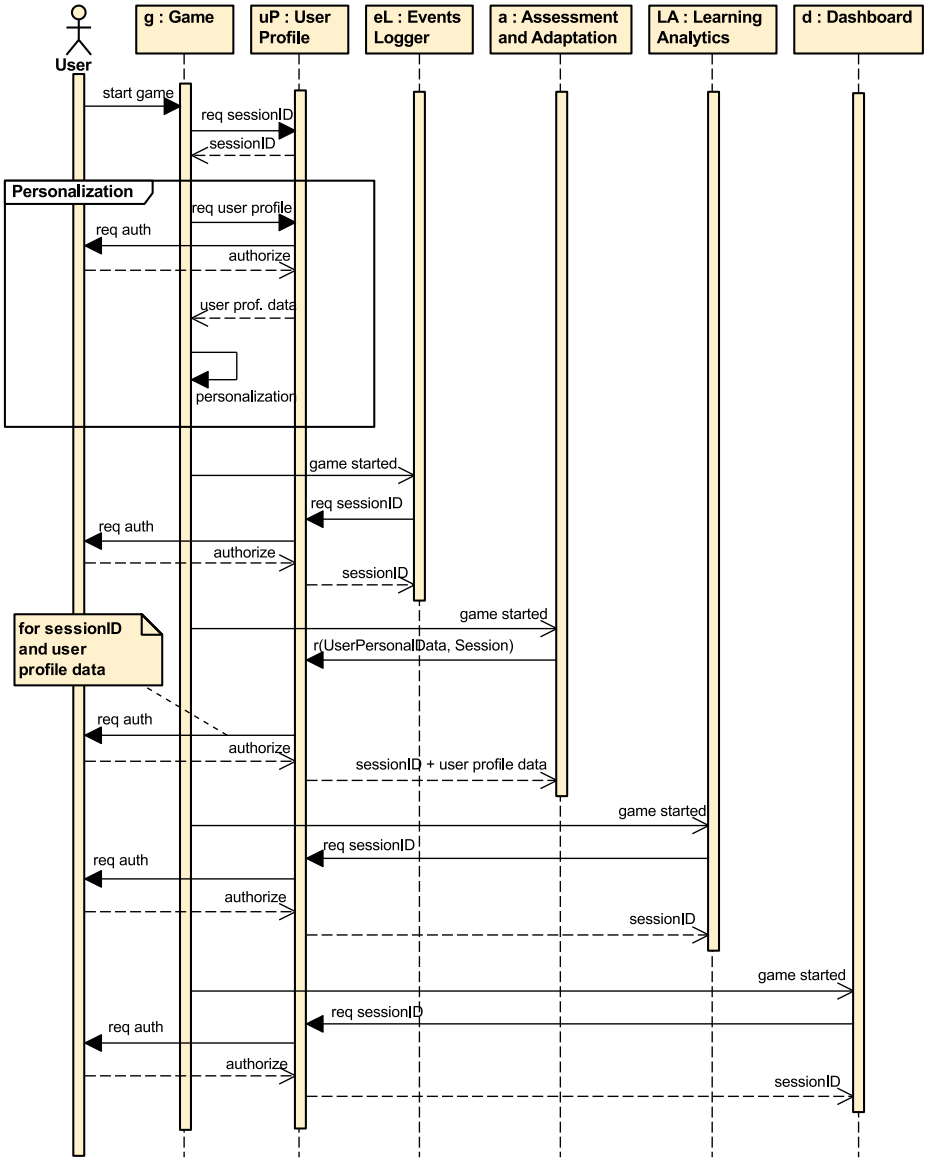


Figure 5.19: Initializing a Game session

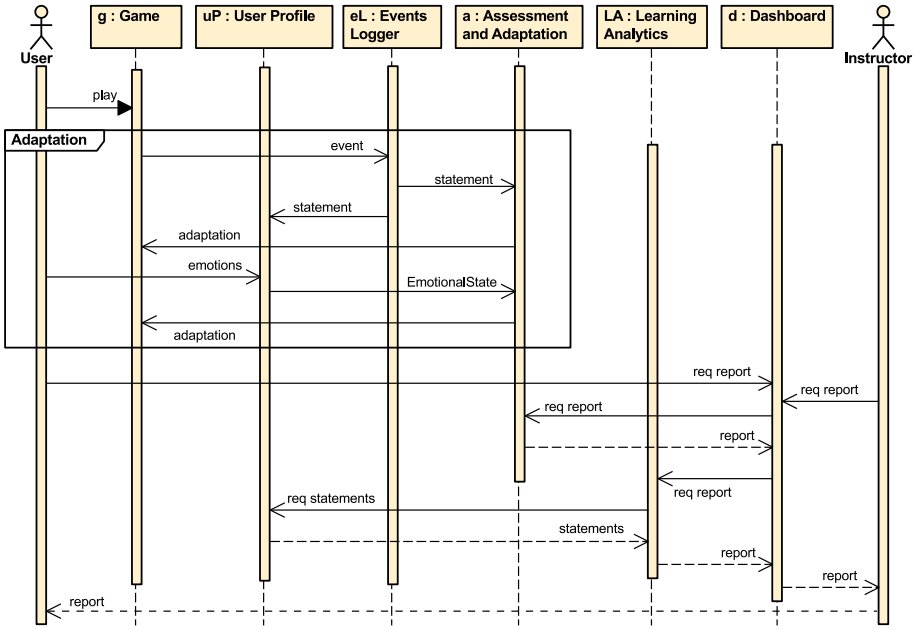


Figure 5.20: During gameplay

back to the game. If there is any Emotional assessment, the User Profile component, via an external component, senses the emotions of the player and sends a the Emotional State representation to the Assessment and Adaptation component, which in turn incorporates the data when generating the Adaptation suggestions.

An Instructor can, at any time during gameplay, request access to reports about the ongoing gaming session via the Dashboard (Figure 5.20). When the request is made, the Dashboard forwards the request to the components that are responsible for generating those reports, namely the Learning Analytics and the Assessment and Adaptation components. The Learning Analytics needs to request the *statementsList* directly from the User Profile. In other words, the Learning Analytics reports do not produce real-time reports during gameplay, but when requested, it can make an updated request to the User Profile to receive an up-to-date list of statements, which can then be used to generate a report on the current state of the gaming session. If authorized to access the User's data, the Dashboard receives and displays the requested reports. Real-time monitoring the user's performance can be achieved by observing the stream of statements directly from the User Profile component, or through the automatic request of updated reports from the Learning Analytics component in short time intervals.

Finally, Figure 5.21 illustrates the process of ending the game. When the User finishes playing, the Game sends an End Game signal to all the components, signaling them

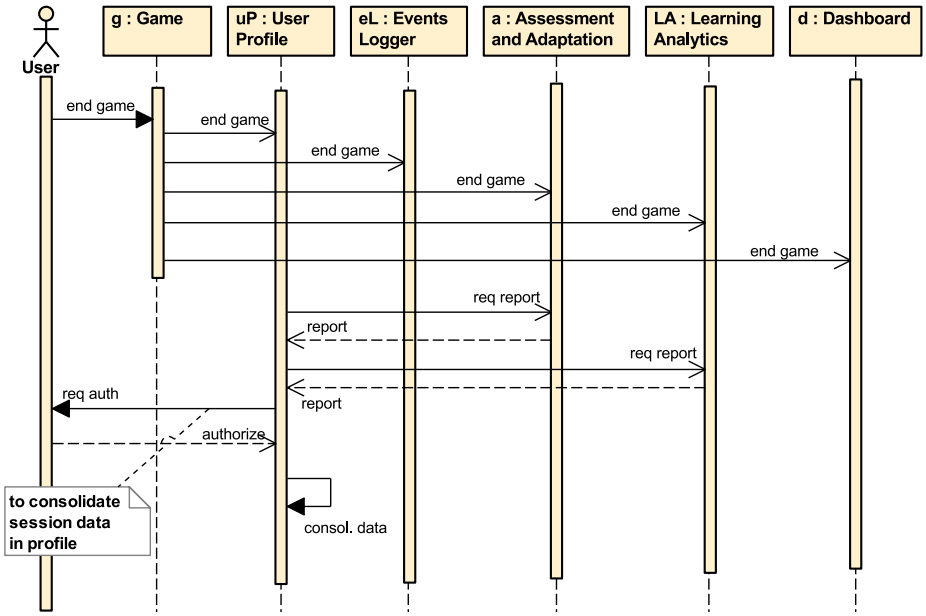


Figure 5.21: Ending the game

to stop communicating with the game and with each other. The User Profile component requests consolidated reports from the Assessment and Adaptation and Learning Analytics components, which can be used to update User Profile data (e.g. updates in learning styles, gaming preferences, knowledge on certain topics). Any update in the User Profile data needs to be verified and authorized by the User.

5.5 Example implementation

We developed an example implementation of the SORASG as part of the process of defining and evaluating the architecture, particularly regarding its buildability. As explained in Section 5.2, we chose to update an existing open source game so that we could focus on aspects of the implementation of the reference architecture, without the overhead of developing a whole game from scratch. Since this exercise was part of the iterative process of defining the SORASG (see Section 5.2), the game reflects an earlier version of the SORASG.

Our prototype implementation of the SORASG is an altered version of an existing open source game called Lix[4]. Lix is inspired by Lemmings, a 1991 digital puzzle game by

[4]Lix: http://asdfasdf.ethz.ch/ ~ simon/index.php

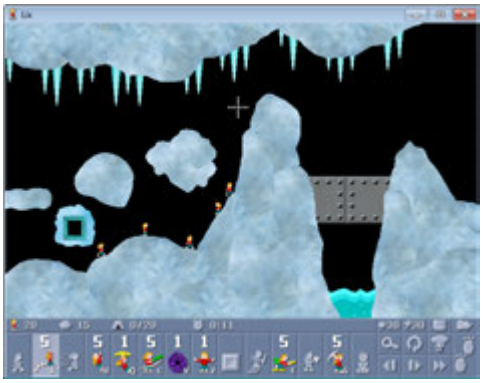


Figure 5.22: Lix game puzzle interface

DMA Design. Although puzzle games are not necessarily considered SGs, they are commonly used for educational purposes (E. Z. F. Liu & Lin, 2009), possibly given its typical reliance on problem-solving and on logical and mathematical intelligence (Becker, 2005). Lix offers an environment of highly constrained interaction, with short and clear feedback loops with clear success criteria. Furthermore, success in the game is not influenced by the player's previous knowledge, contrary to what happens in games with traditional curricular domains. With less interference from external factors in the in-game assessment, the prototype can offer us a simplified environment for the development and evaluation of in-game assessment mechanisms, adaptation, learning analytics and reports. Nevertheless, the system is still close enough to other types of SGs to give useful insights. For subsequent iterations, the SORASG can be tested in more complex game scenarios.

Lix consists of many puzzles, with different levels of difficulty. In each puzzle, the objective is to guide a group of simple characters (called "lixes") to a designated exit. The player must determine how to assign a limited number of skills to specific lixes, which allow the selected lix to alter the landscape, to affect the behavior of other lixes, or to clear obstacles to create a safe passage for the rest of the lixes. The available skills vary between the puzzles and include skills such as climbing, jumping, digging, building bridges, and so on. Each puzzle has a minimum amount of lixes that must be saved; in some more advanced puzzles, a time limit is also enforced. Figure 5.22 shows a screenshot of one puzzle of the game, with the bottom panel that is used to control the characters.

Our altered version of Lix[5] incorporated the following modules of the SORASG: the Game itself, User Profile, Events Logger and a Dashboard. We also implemented an offline Learning Analytics approach using data collected from this setup (Figure 5.23).

[5]The source code is available at http://github.com/carvalhomb.

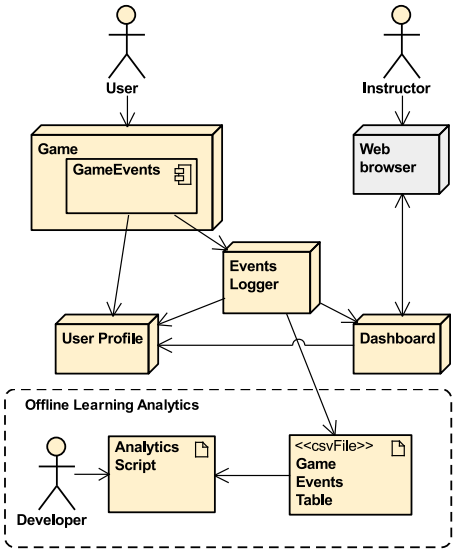


Figure 5.23: Components of the altered version of the game Lix

Game Originally, Lix was developed in C++, using the game programming library Allegro 4[6], which supports basic 2D graphics, text input and output, audio output, among others. The game also has a multiplayer mode; however, in our implementation, we used only the single player mode.

We added an extra component to the game, called "GameEvents" (see Figure 5.23). This component receives, in the form of asynchronous notifications, the game events that must be exposed to the rest of the system. The "GameEvents" component was implemented using the open source library Poco[7], which provides tools for parallelization and networking, among others. The "GameEvents" component runs as a background process in the game so that the game interface does not stop while waiting for the response of its network calls, thus avoiding performance issues. This implementation uses a Poco Notification Queue that collects game events and uses multiple worker threads[8] to submit notifications to the Events Logger component.

The requests sent by the Game to the Events Logger always include the Session ID to which the events belong.

The game events recorded during gameplay are of two types: *game* traces and *meaningful variable* traces (Serrano-Laguna et al., 2014). The game traces indicate timestamps of

[6]Allegro game programming library: http://liballeg.org/
[7]POCO C++ Libraries: http://pocoproject.org/
[8]Three threads were used by default, as testing under normal usage showed this number was enough to handle the load. If necessary, more threads can be included to deal with a more intense stream of events.

when the player started the game, started or restarted a puzzle, paused the game and returned to the menu. The meaningful variable traces consist of a simple record of a timestamp, a short code describing the skill assigned to a character (lix), an internal identifier of the character to which the skill was assigned and an internal measure of game time (called "update"). At the end of each attempt at solving the puzzle, a results entry is recorded, indicating the attempt duration (both in "updates" and in seconds), the number of skills used, and the number of saved lixes (out of the required number). Table 5.9 shows an example of the format of the results recorded and sent to the listening web service.

Timestamp	Action	Level	Upd	Lix	Lix req	Lix saved	Skills used	Secs
2015-12-10T14:29:19Z	STARTLEVEL	14						
2015-12-10T14:30:20Z	ASSIGN=MINER	14	949	1				
2015-12-10T14:31:15Z	ASSIGN=CLIMBER	14	1766	9				
2015-12-10T14:31:23Z	ASSIGN=MINER	14	1890	13				
2015-12-10T14:31:41Z	ENDLEVEL	14	2133					
2015-12-10T14:31:41Z	**RESULT**	**14**			**20**	**0**	**13**	**142**
2015-12-10T15:12:24Z	STARTLEVEL	14						
2015-12-10T15:51:56Z	ASSIGN=JUMPER	14	1594	2				

Table 5.9: Example of game events recorded and sent to the EventsLogger service

Timestamps are recorded in the standardized format ISO 8601, in the UTC (Coordinated Universal Time) time zone, to avoid confusion between services hosted in different places.

As described in Section 5.4, the communications between the Game and the Events Logger feature a Session ID as unique identifier. The Game requests the Session ID from the User Profile component at the start of a gaming session.

User profile The interface of the User Profile component is implemented as a Representation State Transfer (REST) web service. It was developed in Python, using an open source framework called Flask [9].

The current version implements a simple User Profile that holds a username, a user ID, a password and the list of active and inactive gaming sessions associated with that user. The User Profile creates a new gaming session when it receives a request from the Game. Currently, it does not validate session requests from the other components. Future development will include complete authorization, to confirm that the user has authorized each component to access to his or her data.

Events logger Like the User Profile, the Events Logger component is also a REST web service developed using Flask.

[9]Flask web development: http://flask.pocoo.org/

When receiving a new request from the Game, the Events Logger checks with the User Profile if the Session ID is valid before it starts to accept game events.

In the current version, the Events Logger service is used to generate a downloadable file with the list of game events associated with a session, which can be downloaded by the Developer. It also sends the events directly to the Dashboard, which then displays the stream of events to the Instructor. This behavior is not in line with the most up-to-date version of the SORASG, which specifies that the Events Logger sends processed xAPI statements directly only to the Assessment and Adaptation and to the User Profile components. The Dashboard would not display a stream of events, but instead, it would show reports on the player's activities. Future development will alter the system according to the most up-to-date specifications of the reference architecture.

Dashboard The dashboard implemented is a simplified version, which simply lists the game events associated with a certain session ID.

In a future implementation, the dashboard will be able to receive formatted reports from the Assessment and Adaptation and the Learning Analytics services, providing visualizations of the playing session and graphical representations of the state of the player's knowledge map. Possibly, it will be able to display aggregate graphics from a larger demographic group also.

Offline learning analytics Currently, the learning analytics is performed offline, as a stand-in functionality for a future online learning analytics component. The analysis uses a Comma-Separated Values (CSV) file containing raw game events. This file is downloadable from the Events Logger component (see Figure 5.23). The Developer could process the file to extract meaningful data. Details of the first learning analytics approach applied to the data can be found in Vahdat et al. (2016).

Authentication and authorization Clients authenticate each other via exchange of credentials followed by token authentication. The requesting Client sends its credentials in a handshake request. If the receiving client recognizes the credentials, it sends back an authentication token with an expiration time chosen by the developer. The token is sent as a header in all subsequent requests. If a session continues after the expiration, it is possible to request a new token to replace the expired one. This procedure is a simplified method that does not require an authentication server.

The system does not fully implement user control of access to personal data. Instead, we developed placeholder (stub) functions without actual authentication, which indicate how the authentication flow would happen through the defined interfaces. Future versions would require the implementation of such authorization to conform fully to the SORASG specifications.

❧

Our alterations on the game Lix were based on an earlier version of the SORASG. In fact, as explained in Section 5.2, the architecture was updated according to the lessons learned during the development of the game. Consequently, the connections between the components depicted in Figure 5.23 do not reflect the architecture as it is described in Section 5.4. Namely, the Dashboard and Learning Analytics components should not be connected directly to the Events Logger, but rather to the User Profile component, which would then send game data to the other components. Also, the current implementation of the Events Logger does not perform the translation of raw game events into xAPI statements, which is also a key characteristic of the SORASG.

Figure 5.24 shows how the Lix architecture should be updated to conform to the latest version of the SORASG.

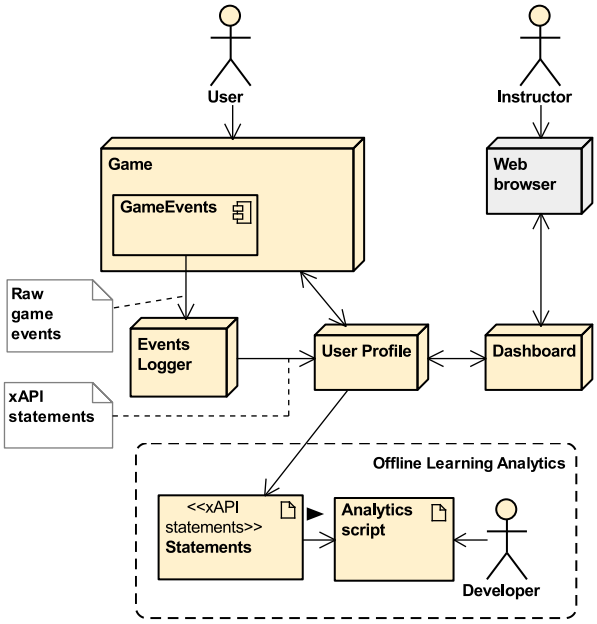


Figure 5.24: How the components of the altered version of the game Lix should be implemented to conform to the SORASG

The game, as it is now, does not include an Assessment and Adaptation component. While not compulsory, such a component is considered an important addition in SGs for educational purposes. For this reason, a future version of this game should include Assessment and Adaptation functionalities, possibly implementing two layers of adaptation. One layer would be intra-puzzle, with hints for players depending on their performance; the other layer would be in-between puzzle adaptation, in which the game would be able to implement a path of different puzzles to be tried that would always be in line with a player's current abilities.

5.6 Evaluation

In this section, I present the results of our evaluations of the SORASG. First, I discuss the functional completeness of the architecture, comparing the functional requirements of the architecture (Subsection 5.3.2) with the modules of the SORASG. Subsequently, I present the architectural analysis phase of the ATAM evaluation. Next, I summarize the results of the ATAM evaluation in the form of a list of risks, non-risks, sensitivity points and trade-off points identified in the analysis. Finally, I discuss the buildability of the SORASG, as demonstrated by the example implementation described in Section 5.5.

5.6.1 Functional completeness

To evaluate the functional completeness of the SORASG, I compare the functional requirements identified in Subsection 5.3.2 with the modules of the SORASG, described in Subsection 5.4.4. In this way, it is possible to check the architecture's compliance with the functional requirements.

Functional requirement	Module(s)
[FR1] Between-players interaction	-
[FR2] Student-instructor interaction	-
[FR3] Information storage and retrieval	-
[FR4] Feedback	Dashboard; Assessment and adaptation; Learning analytics
[FR5] Assessment	Assessment and adaptation; Learning analytics; User Profile
[FR6] Adaptivity	Assessment and adaptation
[FR7] Game connectors	Game; Events logger; Configuration UI
[FR8] User profiling	User profile/User Profile UI
[FR9] Data logging	Events logger

Table 5.10: Functional requirements implemented by each module of the SORASG

Table 5.10 shows the mapping between the functional requirements and the modules of the SORASG.

The mapping between functions User profiling (FR5) and Data logging (FR6) is straightforward, with each of these functionalities being directly supported by one module of the architecture.

The function Feedback (FR1) is implemented most directly by the Dashboard module, via its ability to display reports to the Instructor (who can then use them to give feedback to the User) or directly to the User. Nevertheless, the Reports displayed by the Dashboard are generated by the Assessment and Adaptation and Learning Analytics modules, so they are also listed as providers of the Feedback functionality.

Assessment (FR2) is implemented mainly by the Assessment and Adaptation and the Learning Analytics modules, which generate reports on the learning progress of the User. The Assessment and Adaptation module is formed by internal components, one of which is responsible for providing learning assessment. This learning assessment component is used internally in the system that supports adaptation. If the system supports emotional or engagement assessment, this functionality is attached to the User Profile module.

The Assessment and Adaptation module is the provider of the Adaptivity (FR3) functionality, via one of its internal components that contains the adaptation logic (represented by an InterventionModel).

The Game connectors (FR4) functionality is represented by the components that link the game and the components providing the system functionalities. The Game module itself implements part of this link, by exposing its relevant game events to external components. The Events Logger has a crucial role in providing this functionality, as it is the component responsible for translating raw game events into xAPI statements that can be used by the other components. Finally, the Configuration UI module provides a way for the Developer to provide game-specific configuration files, which are important for linking the components defined in the architecture.

The architecture does not include all possible functional domains that a SG could implement. For example, interactions between-players, interactions between the student and the instructor, and external modules for information storage and retrieval are not present in the architecture. Nevertheless, the SOA approach encourages that, whenever relevant, a game makes requests to external auxiliary modules. These modules could be, for example, databases holding relevant information to the game (e.g. factual information databases, exercises databases), modules to help in processing information for the game (e.g. modules to parse natural language queries), or social networks to retrieve information about the User's friends and their performances in the game. The modular, service-based architecture allows the developer to reuse existing software more easily, even components that are not explicitly defined in the SORASG.

5.6.2 Analysis of architectural approaches

I performed the ATAM analysis of the architecture to evaluate to which extent the SORASG supports the desired system QAs. As explained in Subsection 4.1.2, an ATAM evaluation consists of a systematic analytical comparison between the desired characteristics of the architecture and the architectural approaches that are used in the architecture.

From the QAs and scenarios identified in Subsection 5.3.3 and summarized in the utility tree (Table 5.7), the QAs identified as high priority were selected to be examined in detail against the architectural approaches described in Subsection 5.4.2. The results are described in the tables below. Each table shows the risks (R), non-risks (NR), sensitivity points (SP) and trade-offs points (T) that were identified in the analysis; each item is explained after the tables. A summary and a discussion of the outcomes of this analysis are presented in Subsection 5.6.3.

Scenario S1				
Description: Different coding teams can work at different parts of the system at the same time, with minimal coordination.				
Attribute: Development distributability (QA1)				
Environment: Development time				
Architectural Decisions	**R**	**NR**	**SP**	**T**
[A1] Modularization and layers	-	NR1	-	-
[A2] Service orientation	R1	-	-	-

Scenario S2				
Description: It is possible to reuse existing components in the game, even if developed by third parties.				
Attribute: Development distributability (QA1)				
Environment: Development time				
Architectural Decisions	**R**	**NR**	**SP**	**T**
[A1] Modularization and layers	-	NR2	-	-
[A2] Service orientation	R1, R2	-	SP1	-

Scenario S3				
Description: The system can be easily reconfigured for use with alternative components providing similar functionalities.				
Attribute: Modifiability (QA2)				
Environment: Development time				
Architectural Decisions	**R**	**NR**	**SP**	**T**
[A2] Service orientation	R2	NR2	-	-

Scenario S4				
Description: Related games, with similar characteristics or content, can be created with minimal effort.				
Attribute: Modifiability (QA2)				
Environment: Development time				
Architectural Decisions	**R**	**NR**	**SP**	**T**
[A1] Modularization and layers	-	NR1	-	-
[A8] Use of configuration files	-	NR5	-	T4

Scenario S5				
Description: The game can be changed in order to incorporate new content or tasks.				
Attribute: Modifiability (QA2)				
Environment: Development time				
Architectural Decisions	**R**	**NR**	**SP**	**T**
[A8] Use of configuration files	-	NR5	-	-

Scenario S6				
Description: The game is capable of exchanging information with connected components, and the data is automatically interpreted.				
Attribute: Interoperability (QA3)				
Environment: Runtime, normal operation				
Architectural Decisions	**R**	**NR**	**SP**	**T**
[A2] Service orientation	R2	-	-	-

Scenario S7				
Description: Game components can run in different platforms.				
Attribute: Interoperability (QA3)				
Environment: Runtime, normal operation				
Architectural Decisions	**R**	**NR**	**SP**	**T**
[A2] Service orientation	R1	-	SP1	-

Scenario S8				
Description: Player's data is stored in an open standard format. The data can eventually be imported and interpreted by third-party tools (e.g. LMS).				
Attribute: Interoperability (QA3)				
Environment: Development time				
Architectural Decisions	**R**	**NR**	**SP**	**T**
[A7] Use of xAPI specification	-	-	SP2	-

Scenario S9				
Description: Players' personal data is not disclosed to third parties. Storage and transfer of players' personal data is minimized.				
Attribute: Privacy and security (QA4)				
Environment: Runtime, normal operation				
Architectural Decisions	**R**	**NR**	**SP**	**T**
[A5] Limitation of exposure of user data	R3	-	-	T1

Scenario S10				
Description: No personal data is disclosed without users' explicit consent. Users can revoke rights to their data at any time.				
Attribute: Privacy and security (QA4)				
Environment: Runtime, normal operation				
Architectural Decisions	**R**	**NR**	**SP**	**T**
[A6] Explicit authorization	R3	-	-	T1, T2

Scenario S11				
Description: The player's gaming experience is not affected by the software response times.				
Attribute: Performance (QA5)				
Environment: Runtime, normal operation				
Architectural Decisions	**R**	**NR**	**SP**	**T**
[A3] Choreography	-	NR3	-	-
[A4] Asynchronous messaging	R4	NR4	SP1, SP3	T3

Risks

[R1] Degraded performance: With the service layer making the intermediation between the game and the service components, the possibility of network delays affecting the response is higher than if the component was developed as part of the game, or if the game accessed it directly.

[R2] Badly defined interfaces or poor documentation may hinder reuse of third party components.

[R3] Malicious service providers may permanently store data that should only be requested when needed (e.g. the connection between session ID and user ID).

[R4] Asynchronous messaging has the risk that the game does not receive adaptation suggestions on time. When the delays are too big, the adaptation suggestions become useless.

Non-risks

[NR1] Logical separation of modules with well-defined responsibilities facilitates development.

[NR2] Logical separation in layers makes it possible to use components even when they were not developed specifically with games in mind (e.g. third-party LRSs, emotion recognition components).

[NR3] A choreography configuration for service composition is more complex than orchestration, and eventual changes in the communication between components might bring high maintenance cost. Nevertheless, the only module that might require changes in the sequence of interaction is the Assessment and Adaptation, which has a Controller class (AssessAdaptController, Figure 5.16) that serves the purpose of concealing changes and making sure these changes do not need to be reflected in the rest of the system.

[NR4] A timestamp is included in the Game events and Adaptation messages to enable the game to judge the timeliness of the suggestions, so that the game is able to ignore outdated adaptation suggestions.

[NR5] Unforeseen content-types, tasks, and/or adaptation requirements may prevent the use of the architecture with the current division of modules and layers or currently defined configuration files. However, this is a non-risk since the SORASG can evolve to incorporate unforeseen game configurations and content types.

Sensitivity points

[SP1] Response time is sensitive to the number of clients using a component at the same time. It is necessary to make sure that the component can provide an acceptable level of

quality of service.

[SP2] The quality of the Translator file (Figure 5.11) that converts raw game events into xAPI specification is crucial to ensure a usable file. It is important that the Translator use predefined statements as much as possible.

[SP3] The nature of the game, and particularly the number of calls to the Events Logger, may affect response times.

Trade-off points

[T1] There is a tradeoff between not disclosing the key between a session ID and a user ID to third parties to prevent unauthorized access to user data, and being able to assess a user for longer periods than just the current gaming session in the dashboard.

[T2] Assessment reports that involve long-term usage of the game are more difficult to generate and depend on user's explicit authorization (they might require multiple interventions from the user).

[T3] The number of game events sent at once (combining multiple events or sending one by one) defines a trade-off between the network load and the responsiveness of the adaptation (how often the game can adapt). In environments in which network quality cannot be guaranteed, the game might need to rely on less frequent adaptations, and/or implement mechanisms to be able to wait for responses without affecting the player's experience.

[T4] Configuration files are more complex than a graphical interface for a developer to provide the system with required settings. They also have a steeper learning curve. However, they offer the possibility of making many changes at once, minimizing possible down times. The disadvantages from the usability viewpoint can be reasonably compensated by providing clear documentation on how to write the configuration file. Another possibility is to create a simple graphical user interface that generates and tests the configuration file before it can be fed into the system.

5.6.3 Risks, non-risks, sensitivity points, and trade-off points

In this section, I summarize the risks, non-risks, sensitivity points and trade-off points that were identified in the analysis of architectural approaches (Subsection 5.6.2), grouping them according to their general themes.

The architecture addresses well three main quality attributes, namely Development Distributability, Modifiability and Interoperability. The SOA approach, with its logical separation of layers and clear interface definitions, was deemed as a facilitator factor for distributed development and interoperability. The architecture is also favorable when modifying existing games or creating new ones, due to the layered architecture

that compartmentalizes changes to specific components, preventing a ripple effect when changes need to be made. Furthermore, the use of mapping files between game events and statements, knowledge maps and intervention models is intended to provide a way for concentrating the configurations that would need to be changed for different types of games. Unforeseen content types, tasks, and/or adaptation requirements may prevent the use of the architecture with the current division of modules and layers. However, the SORASG can always evolve to incorporate unforeseen game configurations and content types.

Configuration files offer a trade-off between system simplicity and usability for the developer. Configuration files are more complex than a graphical interface for a developer to provide the system with required settings. They also have a steeper learning curve. However, they offer the possibility of making many changes at once, minimizing possible down times. The disadvantages from the usability viewpoint can be reasonably mitigated with clear documentation and possibly graphical user interfaces to generate and test the configuration files.

The quality attribute Performance has been addressed with the application of a choreography configuration, eliminating a centralizing orchestrator, and thus removing a possible bottleneck in the interaction. Furthermore, time-stamped asynchronous messaging prevents synchronization issues that could affect the gaming experience. However, the analysis indicated that there is still the risk of slow network responses due to the communication loops that add extra network and computing demands when compared to a non-layered architecture. The architecture's response to this issue prevents that the regular gaming experience is affected. In cases where network delays happen, delayed messages can be ignored by the game – in which case adaptivity would simply not happen. The rest of the system setup is unaffected, which is considered a positive characteristic of the architecture. Nevertheless, this issue highlights the need for developers who adopt the SORASG to consider the tradeoffs carefully. They should consider using techniques to minimize and speed up network traffic (e.g. combining and compressing messages before sending). The decision on when and how often adaptation should occur is also important so that problems with the timing of adaptations are reduced (e.g. performing adaptations between levels or game activities, including animations or pauses in the game while waiting for responses from the server). Finally, developers need to choose third-party components that offer an acceptable level of quality of service, and that can deal with the demands of the system.

Finally, the QA of Security of user's data presents a risk that developers need to be aware. While using third-party components brings the benefits already explained previously (e.g. reuse of specialized components, always up-to-date, no need for maintenance), the SORASG highly depends on the quality and trustworthiness of these components. Quality, because by employing components developed by third-parties, the development team depends on the quality of the interfaces and documentation, and on the provided quality of service. Trustworthiness, because while the architecture limits the exposure of user data and user IDs to third party components by using session IDs

as identifiers (thus avoiding exposing more information than strictly necessary for processing by the third-parties), the risk of exposure of user data is not eliminated by this technique. For the proper functioning of some components of the system, particularly those that are responsible for generating aggregating reports or displaying information to the user or the instructor, some user data may have to be shared. For example, the architecture defines that the Dashboard would not hold the User ID related to a given Session ID, but only request it once the Instructor needs to access a report. The assumption is that, once this information is no longer needed, the Dashboard would discard it. A malicious component could keep this information instead of discarding it, building over time a complete mapping of session IDs and user IDs. In other words, while the architecture makes it harder to disclose to whom the playing and learning data belongs, it does not completely prevent it. Employing the usual security measures, such as using only trustworthy components and employing encryption methods in the communication, is still necessary to ensure user privacy.

5.6.4 Buildability

Building a prototype of a reference architecture is a way to prove its buildability (Angelov et al., 2008). Although the example implementation described in Section 5.5 refers to an earlier version of the SORASG, it was a valid exercise in demonstrating how new and existing games could be altered to conform to the specifications of the reference architecture.

The example implementation showed that only a small alteration was needed in the original game: the development of a new component that runs in the background collecting relevant game events and sending them to an external component. This simple alteration made it possible to incorporate two relevant functionalities to the game: an instructor could follow a player in real-time from another machine, and a developer could perform analyses on game data to extract interaction patterns. This demonstrates the modifiability (QA2) of the game architecture.

The experience also confirmed the expected interoperability (QA3) between components, since the game was developed in one programming language (C++) and the other components were developed in another (Python). The fact that the communication between them happened through a commonly used and standardized web service interface (REST) ensured interoperability. It also facilitated development, in which we had no problems in finding existing open source libraries and frameworks. Being able to use open source software significantly sped up the development.

Nevertheless, the example implementation was a simplified version of the architecture, which did not implement two key aspects of the SORASG. The use of xAPI statements has not been implemented yet, so we could not evaluate how much the translation from raw game events would impact the rest of the system regarding processing times. Furthermore, we have not tested the use of xAPI statements in learning analytics to observe if important game data is lost in the translation. Finally, since we did not implement

the Assessment and Adaptation component, we also could not assess if the format of the adaptation messages would be useful for the game, nor if the response times would be acceptable.

5.7 Discussion

5.7.1 Quality and applicability of the architecture

The evaluation of the SORASG considered three main criteria: functional completeness, achievement of desired quality attributes, and buildability.

As the SORASG is a preliminary facilitation architecture (Type 5.1, see Subsection 5.4.1), its goal is to provide guidance and inspiration, not to establish a standard. SG developers have the choice expand the architecture – and alter it, when needed – to suit their projects' characteristics. As such, the functional completeness of the architecture should be judged according to the functionalities identified in the requirements, even if these functionalities are a subset of those that will be found in an actual SG implementation.

The SORASG implements all six functionalities identified as relevant during the requirements elicitation phase. Three functionalities were identified in the analysis of the business domain but were not included in the specifications of the SORASG (Between-players interaction, Student–instructor interaction and Information Storage and Retrieval). These functionalities were not considered universally relevant nor homogeneous enough to be included in the scope of an overarching reference architecture. The functionalities that were included in the SORASG, conversely, define a general framework for implementing assessment, adaptation, monitoring and feedback in games, independently of the content and game genre.

Regarding the analysis of quality attributes of the SORASG, our conclusion, based on the ATAM analysis, is that the reference architecture addresses well the needs of SG developers. Specifically, it provides structures that: are easier to be developed in teams; allow for easier reuse of existing parts; and are easier to maintain and modify when needed. Such flexibility, however, comes with two main costs: performance and security. While the SORASG provides acceptable solutions to minimize the impact of a distributed architecture in the game performance and data security, these still represent risks that need to be evaluated by the stakeholders in face of each project's technical and non-technical requirements and available resources.

The SORASG can be regarded by SG developers as a tool to incorporate evolutionary prototyping in their practices, allowing them to iteratively improve their products without increasing costs. The SORASG facilitates rapid development of games, in which third-party solutions can be incorporated quickly in the development of prototypes. These prototypes can be tested with real users in real usage settings. Once the game matures and reaches a stable version, developers can reassess the quality needs of the

components and, if necessary, replace them with other components that offer better quality of service. The replacement of components is facilitated by the SOA approach, since the interface definitions and standardized protocols ensure that the components will be able to communicate, independently of where the service provider is deployed. It is possible to configure components to run locally to ensure performance and security standards with minimal changes in the deployment configurations.

We managed to develop, within less than two months, with solely one developer, a simple implementation of the architecture. We were able to use open source libraries to adapt an existing game to include several functionalities contemplated by the reference architecture, which attests to the compatibility of the architecture with legacy code. Although it is not possible to estimate precisely the extent of the workload saved, the experience satisfactorily illustrates the buildability of the SORASG in a real application setting.

Despite our efforts in evaluating the SORASG systematically, a real assessment can only be achieved by measuring its impact in the domain – this is true for any preliminary facilitation (Type 5.1) reference architecture. When other researchers and developers implement SGs using the SORASG, it will be possible to assess the reference architecture objectively against its goals (i.e. reduce development costs while maintaining quality, promote reuse of existing technological solutions, and promoting interoperability via the use of open standards). Real-world implementations will also allow us to evaluate the technical quality of implemented games using objective performance and usability measurements.

5.7.2 Connecting ATMSG and SORASG

The two main contributions presented in this work – the ATMSG and the SORASG – are targeted mainly at two different groups. The ATMSG model can be used by SG designers and instructors to analyze existing SGs or as a tool to support SG design. The SORASG, conversely, is targeted at game software developers and software engineers, who may use the reference architecture in their projects directly.

These two contributions are connected in two main ways.

The first way in which the two contributions are connected has already been described in Subsection 5.3.2: the ATMSG model served as the conceptual structure that allowed the definition of a comprehensive taxonomy of SG elements that connects existing gaming, learning, and instructional elements into one. This taxonomy is relevant to a wide variety of SGs, since there are no assumptions about the nature of the game or its instructional elements. The taxonomy gave an overview of the elements that can form different games, and as such it was a valuable source at the initial stage of the development of the SORASG.

The second way is the following. Applying the ATMSG model can also help game

developers incorporate the SORASG in their work. The SORASG is still a template that needs to be further refined, and the ATMSG model can help that process. At the prototyping stage, an ATMSG analysis can be performed. Using the output table (e.g. Figure 3.5 on page 36), the SG designer can map the components that are related to the functionalities provided by the SORASG, using Table 5.5 (page 70) as a guide. This analysis can help the designer in deciding which of the functionalities of the SORASG are relevant, and then communicate the requirements to the SG developers. Furthermore, the ATMSG analysis can assist in defining which game events are relevant to be exposed to rest of the architecture, by highlighting which are the educationally relevant parts of the game and how they are matched to the elements of the gaming activity (again, see Figure 3.5 on page 36). Such visualization can also be helpful in generating the mapping between the gaming actions and the xAPI statements.

Chapter 6

Concluding remarks

In this chapter, I summarize and discuss the work presented in this thesis, going back to the original research questions and problem statement posed in the Introduction and arguing about the implications of the findings for the field of serious game (SG) design and development. I also reflect on the limitations of the research and list pointers for future work.

6.1 Research conclusions

Current evidence indicates that SGs have a promising role to play in education. However, SG development costs are still quite high, and producing high-quality and effective games is a difficult task. In this work, I have addressed this issue, in an attempt to present ways to reduce the costs associated with SG development (i.e. effort required to develop or maintain software), while fulfilling the game's educational and entertainment goals.

Throughout this thesis I have presented two main contributions: a theoretical one, in the form of the Activity Theory-based Model for Serious Games (ATMSG), and a technical one, the Service-Oriented Reference Architecture for Serious Games (SORASG). Combining theory and practice allowed me to propose a solution for a software engineering problem (i.e. high SG development costs) that is grounded on what we currently know about the educational and entertainment aspects of SGs and game-based learning (GBL).

To conclude this thesis, I address each of the research questions formulated earlier separately.

RQ1 *How does a game realize its educational and entertainment objectives through its concrete mechanics?*

The field of SG design is undoubtedly advancing. The early days in which researchers needed to argue that using games for learning is a good idea are gone; the general understanding nowadays is that games can indeed be good learning tools. Nevertheless, using SGs in educational and training settings is, more often than not, a clumsy endeavour. On the one hand, SGs commonly fail to engage players, as these games still lack the engaging elements that players have come to expect from commercial off-the-shelf (COTS) games. On the other hand, COTS games are sometimes used to make for captivating gaming sessions in the classroom; but then, how can educators make sure they fulfill the desired educational objectives? In other words, we are still left with questions about how exactly we can build and use games to support learning.

In this scenario, understanding how a SG realizes its educational and entertainment objectives through its mechanics is important. It allows us to look inside this "conceptual black-box" (the serious game) and make sense of what is going on that might explain the success (or failure) of a game, which in turn can help us build better SGs.

The issue of understanding SGs in more depth has drawn some attention from the research community. Many of the current models, methodologies, and frameworks for the analysis and design of SG focused on high-level aspects of what makes SG effective, but only a few provided an investigation into the concrete mechanics of the game. Specifically, they could guide a SG designer on what should be expected of an effective SG, but they would offer very limited insights into which mechanics would enable a game to achieve those goals. Two of these models (i.e. Game Object Model II (GOM II) and Learning Mechanics–Game Mechanics (LM-GM)) did offer a way to look into the concrete mechanics, but then the link to the high-level aspects was lost.

To fill that gap and give a proper answer to RQ1, I proposed the Activity Theory-based Model for Serious Games (ATMSG) model (Chapter 3). The model is built over the conceptual framework of activity theory, which is a line of research in the social sciences that studies different forms of human practices and development processes. The hierarchical structure of the activity provides a way to reason about the relationships among the SGs components and between these components and the educational goals of the game.

One of the main contributions of the ATMSG is identifying the main activities involved in the use of SGs for education: the gaming activity, the learning activity, and the instructional activity that is further decomposed into intrinsic instruction and extrinsic instruction. While the learning activity corresponds to the viewpoint of the learner, the instructional activity depicts the side of the instructor. Previous models and frameworks only distinguish between gaming and learning aspects, but do not provide fine-grained distinctions among educational aspects of SGs. These distinctions clarify differences between the learner's and the instructor's goals, and also which parts of the instruction is given by the game and which ones require the intervention of the instructor. This is a richer representation of the SG, in which the game is a tool used by people with different objectives and expectations. It allows the observer to think about how these different objectives can interact in the game and affect its learning and entertainment outcomes.

Another important contribution to the understanding of the SG objectives through its mechanics is the extensive taxonomy of SG elements, which unifies existing vocabularies into one single taxonomy, with almost 400 items classified in 36 categories. The taxonomy helps identify and classify SG components according to their characteristics and roles in a SG. It is a valuable tool not only for understanding existing games, but also as inspiration for SG designers, who can refer to this comprehensive list when deciding which actions, tools and goals can best support their desired objectives in the game.

The ATMSG model is accompanied by a practical guide for users who want to apply this theoretical model in practice when evaluating or designing SGs. This thorough analysis of existing games results in a series of tables and diagrams that show explicitly the relationship between high-level objectives and concrete game mechanics as the game unfolds. Despite its complexity, an ATMSG analysis provides detailed insights about a SG, which can be useful, for example, when adapting games in specific learning settings or when detailing, identifying and cataloging design patterns. The ATMSG model can also be used as a tool in SG design, supporting the analysis of low-fidelity prototypes at early stages of development and generally providing reference and inspiration through the observation of successful patterns in other games.

I conclude, then, that the ATMSG model is a valuable tool to understand how the inner workings of a SG are articulated to achieve its educational and entertainment goals. It has the potential to contribute to a better understanding of SGs and to support the development of games that fulfill their intended objectives. Nevertheless, it is still a theorical – and thus somewhat abstract – contribution. Since the overall goal of this work is to concretely make SG development less costly, with RQ2 I directed my attention to issues related to software engineering.

RQ2 *To which extent can SG development be simplified by reusing existing technological solutions, even the ones that were not created specifically for SGs?*

SG development is a complex and often costly process. In many fields of software engineering, software complexity is managed by finding ways to optimize development, particularly through component reuse. Reuse strategies have been applied extensively in the entertainment game industry for more than a decade, mainly through the widespread use of game engines. In SG development, so far, the effort has been concentrated in creating games that decouple content from underlying software, or creating genre-dependent authoring platforms. That is, the SG designer must conform to what the platform provides.

RQ2 expresses the desire to optimize SG development further by trying to revert that dependency. Instead of conforming the game to the available tools, we look at a way to incorporate the tools (or reusable assets) into the game. However, it is important to investigate which components can be successfully reused without sacrificing game quality. Once again, I highlight the importance of linking back to the theory behind

GBL, so that the choices made by the developers have higher chances of successfully supporting the educational and entertainment goals of the game.

Using the taxonomy of SG elements defined with the ATMSG model, I extracted the elements that are common among SGs of different genres and topics (Subsection 5.3.2). The elements identified fall into nine functional domains, namely: feedback, assessment, personalization and adaptivity, game connectors, user profiling, data logging, between-players interaction, student-instructor interaction, and information storage and retrieval. Not all of these domains have the same level of importance for all games, but they represent functionalities that can potentially be incorporated into SGs in the form of reusable components. Three of these domains – assessment, feedback, and personalization and adaptivity – represent crucial characteristics of successful learning environments, as discussed in the theoretical foundations (Section 2.2). As such, facilitating their inclusion in SGs is a significant contribution to increasing the quality of existing games with relatively low impact on the overall cost of the development.

Thus, from a systematic analysis of the domain that used the ATMSG model as a starting point, I conclude that there are elements of SGs that are desirable in different genres of games, in different domains, and that are generic enough that their implementation could be abstracted from the game and incorporated as external modules. In some cases the functionalities are so general (e.g. emotional assessment, information retrieval) that it is reasonable to assume that it would be possible to reuse solutions that were not created specifically for SGs. Nevertheless, this reuse must be supported by an architecture that successfully incorporates those solutions in the SG, not only from a technical standpoint, but also employing those functionalities to support the learning and entertainment goals of the game. This leads us to the last research question addressed in this work.

RQ3 *How can SG developers incorporate reusable components into their software development projects?*

While the idea of incorporating reusable assets in a SG is appealing, the fact is that it can be difficult for developers to learn new ways of building software. This difficulty is particularly true when it involves a significant shift in the programming paradigm, as is the case with Service-Oriented Architecture (SOA).

To ease the transition and to guide SG developers on how to better incorporate reusable assets into their games, I proposed the Service-Oriented Reference Architecture for Serious Games (SORASG) (Chapter 5). A reference architecture is a template solution for a software domain that provides major guidelines for defining the software architecture of the project, facilitating decision-making early in the development lifecycle. The SORASG was elaborated in an iterative process that included a systematic analysis of the field – represented mainly by the identification of the functional requirements with the help of the ATMSG model – and consultation with SG designers, developers, and researchers. The process also included small-scale evaluations during the design,

the development of an example implementation, and a complete Architecture Trade-off Analysis Method (ATAM) evaluation at the end.

The SORASG represents a significant contribution to the field in which it identifies important SGs functionalities, and shows, via detailed technical specifications, how to incorporate them into the development of SGs of different genres and in different learning domains. At the time of this writing, one European project in relatively initial phases aims to provide a development ecosystem for SGs and applied games in general (RAGE, 2016). In a work published recently by the group, the RAGE Architecture for Reusable Serious Gaming Technology Components is described (Vegt et al., 2016). This architecture highlights general architectural aspects of the components themselves, without describing their functionalities. In other words, it does not describe what each of the components should do, but only how they should be built to be compatible with other components in the RAGE ecosystem. The SORASG, in contrast, was created using a theoretical model as its starting point so that it can provide guidance not only on purely technical aspects, but also on how to implement desirable SG characteristics such as assessment, adaptation and feedback. To the best of my knowledge, the SORASG is the first reference architecture with these characteristics.

The SORASG suggests, but does not prescribe, which components must be incorporated in the system. In addition, it is not restricted to certain game genres or learning objectives. It gives freedom to the development team in defining the learning and entertainment goals of the game and in deciding which mechanics will be used to achieve those goals. At the same time, it facilitates the adoption of key features for efficient learning, namely assessment, feedback, and adaptation.

Moreover, the service-oriented nature of the SORASG allows for the reuse of components that have been developed by third-parties, even those that were not developed specifically for SGs. For example, components created for digital learning environments (e.g. Learning Record Stores (LRSs), competence assessment services, learning analytics services) and for engagement detection (e.g. real-time monitoring of physiological signals) can be incorporated directly into the architecture. This flexibility increases the choices available to the developer, who can more easily benefit from technological advances in related fields such as automatic emotion recognition, natural language processing, learning analytics, among others.

The SORASG takes advantage of the benefits of its service-oriented nature while minimizing the potential pitfalls of its reliance on external services. The ATAM evaluation demonstrated that indeed the SORASG supports the development of flexible architectures that are easier to develop in teams, easier to reuse existing parts, easier to maintain and that can be modified when needed. However, this flexibility comes at the potential cost of performance and security, which are naturally important issues when deploying SGs in real settings. As a result, I argue that the SORASG is particularly suitable for developing SG high-fidelity prototypes that can later be enhanced towards a final version progressively. By reducing game development time, it is possible to test concepts quickly, enabling an iterative process in which it is possible to test early and often, to

achieve a design that meets both educational and entertainment goals. Furthermore, the reference architecture can also be used in final (production) versions of games even when there are strict performance and security requirements: the components can be deployed on a local server, to mitigate performance and security problems and to comply with eventual data storage policies and restrictions.

The example implementation presented in this work (Section 5.5), albeit simple, demonstrated that existing games can also be altered to conform to the specifications of the reference architecture. The experience confirmed that interoperability between components is possible, even when developing components in different programming languages and deploying them on different servers. Finally, the example implementation was developed by relying on existing open source libraries, which can significantly speed up development time and potentially reduce costs.

One point to note is that the evaluations of the SORASG provided are still mostly theoretical. Given its novelty and its preliminary and facilitation nature, the SORASG can be more concretely evaluated only later on, after its release to the research and development community. Then, it will be possible to assess the SORASG in terms of both its impact in the research community and the quality of the games that employ it.

The SORASG, then, is a very practical response to RQ3, giving SG developers an extensive guide on how to incorporate reusable components in their SGs. This guide considers not only purely technical aspects (e.g. the importance of interoperability with other learning tools and the potentially sensitive nature of the user data exchanged between the components), but it also takes into consideration issues that are particular to SG development, such as the central role of assessment, adaptation and feedback for the learning objectives of the game, and the importance of involving instructors in the process, at least by letting them monitor players' progress and evaluate their performance.

The SORASG is a novel and useful tool that can be used by SG developers as a reference of best practices in SOA-based game development. It informs companies and research institutes that may be interested in developing components to be incorporated in such gaming architectures. It can also support the retrofitting of educational capabilities in entertainment games, with small changes required in the original game, as was shown in the example implementation. Indirectly, the SORASG benefits also the end user, bringing the possibility of the increased availability of games that incorporate key functionalities for more efficient learning.

❧

Finally, after having discussed how this work answered the research questions posed in the Introduction, I return to the original problem statement:

PS *How can we reduce costs associated with SG development, while fulfilling the game's educational and entertainment goals?*

The problem statement above addresses a wide-reaching and long-term problem – the cost of development of SGs. This problem is composed of different aspects that range from the technical and conceptual complexity of SG development to the current maturity of the field. In this thesis, I have chosen to focus on the technical complexity of the task, and to explore how to reduce this complexity without sacrificing the pedagogical quality of the SG. The three research questions I posed reflect this framing, and guide the exploration of possible solutions that can contribute towards solving the problem.

Investigating how SGs achieve their learning and entertainment objectives through game mechanics gave me the basis to identify the elements that are common among SGs of different genres and topics. These elements served as input to create a SOA-based reference architecture that helps reduce SG development costs by providing a starting point for development and by promoting the reuse of components in the game architecture. Both main contributions of this work – the ATMSG model and the SORASG – represent high-level solutions that can be employed in practice by developers who would want to reduce SG development costs while maintaining quality. The former gives the theoretical understanding that is essential to guide the conceptual design of an effective game, while the latter can serve as a direct cost-saving resource in the SG development cycle.

To be effective, the effort of reducing the high costs of SG development must be undertaken by the SG research and development community as a whole. This thesis represents my contribution towards this goal. But, as any tool proposed for a research and practice community, it needs to be embraced by it. The challenge now is, then, to implement the tools proposed here – the modules described by the SORASG in particular – and use them in real research and business settings, assess the reference architecture for its cost-saving characteristics, and evaluate games created with it for their gaming and learning quality. With this, the tools I propose in this work can evolve to reflect the ever-changing needs of the SG research and development community.

6.2 Limitations and future work

Evaluation and improvement of the ATMSG model The ATMSG model was evaluated concerning its perceived usability by the target audience, also in comparison with another model for SG analysis. Further evaluation studies would be needed, with a few modifications. Firstly, this study focused on the analysis of existing games only, but I would also like to verify the applicability of the model in the conceptual design of SGs. Also, in the current study, a usability scale was used for the measurements, since the purpose of the evaluation was to improve the tool itself iteratively. The SUS scale measured how easy or difficult it was to use and understand the model, but it did not allow us to investigate the model's usefulness. Future studies should use scales that measure user satisfaction instead, in addition to collecting open-ended comments and analyzing the quality of the ATMSG analyses produced by the users. Furthermore, considering that

ATMSG seems to be more useful to expert users, the next studies should target specifically this user group, performed preferably in contexts in which there is a real need for an analysis tool.

Future work on the ATMSG model also includes the development of a piece of software to facilitate the application of the model, offering an adequate and usable interface to generate the diagrams, choose components and put them in the appropriate place in the game representation. Such a tool could support non-expert users in applying ATMSG, helping minimize the challenges identified in the evaluation of the model. Furthermore, the analyses produced by the tool could be stored in an interchangeable format, which will allow the analyses to be archived in repositories for serious game studies, such as the Serious Games Studies Database (Serious Games Society, 2013b), and used as input material for cataloging game-based learning patterns.

Evaluation of the SORASG reference architecture Despite its relative simplicity, the example implementation gave indications of the feasibility of implementing the architecture. Future work should include a complete implementation of the SORASG that can provide enough evidence for the buildability of the architecture when it incorporates assessment and adaptation elements.

Another limitation of this work is that the evaluation of the architecture was done in a systematic, but still theoretical way. A final assessment of the usefulness and buildability of the SORASG can only be made when other researchers and developers can use the work, implement games, and evaluate their performance against objective measurements.

Multiplayer games One of the limitations of this work is that both the ATMSG and the SORASG refer to single-player games only. Multiplayer games were out of the scope of this investigation.

The ATMSG model does not contemplate the underlying social structures that mediate the relationship between the subject and the object with the community. This limitation reflects the features offered by the vast majority of state-of-the-art SGs, but should be addressed in the near future. In particular, the ATMSG model should incorporate the analysis of collaboration and cooperation aspects in a serious game.

The SORASG model also does not support multiplayer aspects. It can be used in the development of games that can be played simultaneously by many users, and monitored simultaneously by one or more instructors. However, it does not consider interaction between gamers explicitly, not inside the game nor in associated social networks. This is another limitation that should be addressed in the future.

Combining different types of assessment The assessment and adaptation cycle (Figure 5.16) foresees the possibility of combining different types of assessment such as learning assessment and engagement assessment coming from real-time monitoring of physiologi-

cal signals, as discussed in Subsection 2.2.2). However, the architecture does not explain how the inner parts of the Assessment and Adaptation component would achieve this goal. Such functionality is an interesting avenue for future work, since making a SG able to respond to multiple dimensions of the user's gaming/learning experience could significantly improve learning and engagement outcomes in SGs. However, this is not a trivial task. Deciding how the factors can be combined is one of the challenges, as it involves a complex decision model that must take many variables into account, such as player's preferences, game genre, learning domain, instructor involvement, and so on. Technical aspects of such implementation, for example how to configure such service and how to provide an intervention model to the assessment and adaptation service, should then be included in future versions of the SORASG.

Format of the configuration files The SORASG, as it is currently, does not specify the format of its configuration files, i.e. the Translator file (to convert raw game events into xAPI statements), the KnowledgeMap (the tree structure that establishes the learning domain) and the InterventionModel (the model that allows the game to react to the player's performance). The format of these files needs to be defined and tested in games of different genres, to ensure that they can be applicable in a variety of conditions.

Appendices

Appendix A

Interview summaries

This appendix reports in more detail the data collected during the group and individual interviews with stakeholders, as described in Subsection 5.2.3.

Group interviews

In the group interviews, participants were asked about the usefulness of the preliminary version of the reference architecture and the features that they would like the architecture to have. However, their feedback focused on characteristics relevant to non-educational games (marketing, pervasive games). They expressed their desire that the architecture incorporate external events, which are particularly relevant in pervasive games. This feedback reflected the participants' backgrounds and interests, which were more related to the use of pervasive games for advertisement and engagement.

Participants indicated elements of the first reference architecture that were not defined very clearly, particularly a section that was listed as "auxiliary modules". They suggested that this section could be described in more detail. This feedback was incorporated into the second version of the architecture.

When asked specifically about the desired quality attributes (QAs) for a serious game developed following a service-oriented architecture, the participants listed two main concerns: quality of service, particularly related to possible latency in network communications; and privacy issues, which have implications according to privacy laws of different countries, particularly in Europe. Participants also thought that the discussion of business models and return on investment when providing games or game components as services are important issues to be considered and addressed by a reference architecture.

QAs collected:

- performance
- security
- mobility (for pervasive games)

Individual interviews

From the individual interviews with game developers and game researchers, the feedback obtained was more technical. The participants expressed their considerations on functionalities that should be addressed by the reference architecture.

User profiling At the heart of a reference architecture for serious games is user profiling, which is a topic that is researched intensively without a clear answer. How to represent a user profile and which fields a user profile should contain is an important topic that needs to be addressed by the reference architecture.

Reflection on the learning process A reference architecture for serious games should ideally support self-reflection on the learning process. In other words, feedback can occupy an important role, and the user/player's reactions to the feedback could also provide interesting and valuable data for the instructors.

Configuration One important issue for a reference architecture is the need for easy configuration of game-specific settings, particularly configuring how a game can react differently to the same kinds of signals coming from the game and/or the learner. Selecting the relevant game events and how external services can interpret game events is particular to each game and configuring it should be supported by the reference architecture. In other words, an important QA mentioned is modifiability.

In summary, then, the interviews confirmed the importance of a centralized user profile module that is generic enough to be relevant to several games, but without restricting the functionalities of the game. They also highlighted the role of learning feedback in the game. And with regard to non-functional requirements, modifiability was highlighted as a desirable QA.

QAs collected:

- modifiability

Functionalities collected:

- centralized user profiling
- learning feedback in the game

Appendix B

Online questionnaire

Title: Serious games software requirements

Understanding general functional requirements and quality attributes for serious games, especially games for learning.

Thank you for your availability to help in this study. Your feedback is really appreciated!

In our research, we are developing tools to facilitate serious game design and development. With this questionnaire, we aim to collect common functional requirements and quality attributes that are relevant to serious games of different genres and in different learning domains.

Your responses to this questionnaire are anonymous. At the end of the questionnaire, we will ask you for a contact email if you agree to help us further in a follow-up interview.

By clicking "Next", you agree that your anonymized responses may be used in academic publications (journals, conferences, theses) and open datasets for the research community. In reports and data sets that may result from this work, no information will be made public that could identify you.

This questionnaire should take you 25-30 minutes to complete.

❧

Part I – Demographics

1. What is your age? *

2. What is your sex? * [Female / Male]

3. What is your educational level? Please inform the level (or equivalent) that you are currently pursuing or the last level that you have completed. Please choose only one of the following: *

- Primary school
- Secondary school
- Professional/vocational/trade school
- Bachelor's degree
- Master's degree
- Doctoral degree
- No formal education
- Other

4. Please inform your current occupation and industry: *

5. What is your country of origin? *

6. What is your country of current residence? *

7. Please indicate your level of familiarity with digital games. Please choose only one of the following: *

- I have never played digital games
- I have played digital games only once or twice
- I have played digital games a few times
- I play digital games every now and then
- I am a gamer/I play digital games regularly

8. Please indicate your level of familiarity with Serious Games / games for learning. Please choose only one of the following: *

- I don't know what Serious Games or games for learning are
- I have played a Serious Game or game for learning only once or twice
- I have played a Serious Game or game for learning a few times
- I play Serious Game or games for learning every now and then
- I play Serious Games regularly -or- I work with Serious Games or games for learning

9. What is your previous experience with software development in practice? (Mark the bottom-most item that applies.) *

- I have never developed software.
- I have developed software on my own.
- I have developed software as a part of a team, as part of a course.
- I have developed software as a part of a team, in industry or open-source project, one time.
- I have worked on multiple projects in industry or open-source projects
- Other

10. Please explain your answer above, including years of relevant experience. (E.g. "I worked for 10 years as a programmer in industry"; "I worked on one large project in industry"; "I developed software as part of a class project"; etc...)

Part II – Requirements and goals in serious games

In software engineering, a functional requirement defines a function of a system and its components. The set of functional requirements defines what a system is supposed to accomplish. Examples of functional requirements are: "send messages to a mailing list automatically"; "allow a user to edit personal data"; "log all interaction in a database".

11. In your opinion, which are the three most important functional requirements for serious games or games for learning? Please think about functionalities that are independent of game genre or learning topic.

- 11a. Most important functional requirement: *
- 11b. 2nd most important functional requirement:
- 11c. 3rd most important functional requirement:

12. Think about a serious game or game used for learning purposes project that you participated recently, directly or indirectly. If you have not participated in any such project, think about any existing serious game or game for learning that you are familiar with. If you cannot disclose the name of the project, just add a placeholder name for the project that you will recognize during this questionnaire. Please write the name of the project or game: *

13. Please list up to three main gaming, learning and/or teaching goals of the project you informed in question 12. Examples of goals: "helping parents encourage their children to learn Mathematics", or "raise awareness about the bad situation for a certain population in a certain country", or "collecting data from students to allow teachers to understand their habits better".

- 13a. Most important goal: *
- 13b. 2nd most important goal:
- 13c. 3rd most important goal:

Part III – Evaluating functional requirements and goals

14. In your opinion, how well did the project you selected in question 12 implement the functional requirements and fulfill the goals you listed in the previous questions? Please rate it on a scale from 1 to 7. Rating it "1" means: "the game did not implement this requirement or did not fulfill this goal at all". Rating it "7" means: "the game completely succeeded at implementing this requirement or fulfilling this goal". Please choose the appropriate response for each item:

- Requirement informed in 11a
- Requirement informed in 11b
- Requirement informed in 11c
- Goal informed in 13a
- Goal informed in 13b
- Goal informed in 13c

15. If you want, you can explain your ratings here:

Part IV – Quality attributes of serious games

A quality attribute is a non-functional requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. Quality attributes can be used to evaluate the success of the design and the overall quality of the software application.

16. Which quality attributes do you consider relevant to the project you selected in question 12? Please choose all that apply: *

- Availability (readiness to perform a task when needed)
- Interoperability (ability to exchange meaningful information via interfaces)
- Modifiability (ability to easily change the software)
- Performance (ability to meet timing requirements)
- Security (ability to protect data and information from unauthorized access)
- Testability (ability to demonstrate software faults through testing)
- Usability (how easy it is for the user to accomplish the desired task)
- Variability (ability to support production of set of variants of the system)
- Portability (ability to run on different platforms)
- Development distributability (ability to support distributed software development)
- Scalability/ Elasticity (ability to deal with changes in demand during runtime)
- Deployability (how easy it is for the executable software to arrive on a host platform)
- Mobility (ability to support mobile platforms, e.g. phones, tablets, laptops)

- Monitorability (ability to monitor the system during execution)
- Safety (ability to avoid entering states that cause or lead to damage, injury or loss of life)
- Other:

Part V – Ranking quality attributes

17. Consider the quality attributes you selected in the previous question. Please rank them in order of importance for the project you selected in question 12.

Part VI – Scenarios

Scenarios are often used in the design of software architecture to clarify quality attribute requirements. In this context, a scenario is a real world situation which illustrates the desired behavior of a system. It comprises a related stimulus, an environmental condition, and response (preferably quantifiable).

Examples:

- *A developer wishes to modify an element of the user interface of a system at design time. He/she can make the modification without affecting other functionalities of the system, in less than 20 hours of work.*
- *An internal component of the system fails. The system can recognize a failure of an internal component and has strategies to compensate the fault.*

18. Please create a scenario for the following quality attributes:

- 18a. The quality attribute you ranked in position 1 in question 17:
- 18b. The quality attribute you ranked in position 2 in question 17:
- 18c. The quality attribute you ranked in position 3 in question 17:

Part VI – Conclusion

19. Would you be available for a follow-up interview (via e-mail or video conference) regarding your answers to this questionnaire? If yes, please inform a contact e-mail address below. Your e-mail will not be disclosed to third parties. It will only be used for the purpose of contacting you for a follow-up interview.

20. If you have any comments about this questionnaire or this research, please write them below.

❧

This is the end of this questionnaire. Thank you very much for your participation!

Appendix C

Requirements from stakeholders

From the questionnaires, we collected a list of relevant requirements, in the participants' opinion, as described in Subsection 5.2.3. The resulting requirements are shown in Table C.1.

	Item	Occurrences
1	Data logging	4
2	Adaptivity to player's skills	3
3	Provide learning feedback	3
4	Be replayable	2
5	Save state	2
6	Social features	2
7	User can control own data	2
8	User profiling and stealth assessment	2
9	Interactivity	1
10	Mailing functions	1
11	Model learning content	1
12	Provide information on demand	1

Table C.1: Functionalities mentioned by participants in free text questions.

From this list, *data logging* was included as a functional requirement for the architecture, since it was not explicitly contemplated in our list extracted from the analysis of the domain.

The requirements *adaptivity to player's skills*, *provide learning feedback*, *social features*, *user profiling and stealth assessment*, and *model learning content* were already contemplated

within our analysis of the domain Subsection 5.3.2. The functionalities *be replayable*, *save states*, *interactivity* and *provide information on demand*, while relevant for serious games (SGs), do not have a direct impact in the game architecture and thus are not included in the functional requirements of the reference architecture.

Appendix D

Quality attributes definitions

Availability The software's readiness to perform a task when needed.

Deployability How easy it is for the executable software to arrive in a host platform.

Development distributability The software's ability to support distributed software development.

Interoperability The software's ability to exchange meaningful information via interfaces.

Mobility The software's ability to support mobile platforms, e.g. phones, tablets, laptops.

Modifiability The software's ability to easily change the software.

Monitorability The software's ability to monitor the system during execution.

Performance The software's ability to meet timing requirements.

Portability The software's ability to run on different platforms.

Safety The software's ability to avoid entering states that cause or lead to damage, injury or loss of life.

Scalability/elasticity The software's ability to deal with demand changes during runtime.

Security The software's ability to protect data from unauthorized access.

Testability The software's ability to demonstrate software faults through testing.

Usability How easy it is for the user to accomplish the desired task using the software.

Variability The software's ability to support the production of a set of variants of the system.

Appendix E

Scenarios

As explained in Subsection 5.2.3, we asked participants to write scenarios illustrating the top three quality attributes (QAs) that they selected and ranked for a project that they had participated in before. These scenarios served as input for us to write the consolidated scenarios shown in the Utility Tree (Table 5.7).

Table E.1 below lists the original scenarios described by the participants for the three most important QAs that they identified in their selected project (rank 1 is the most important, rank 3 is the least).

Project	Ranked	Attribute	Scenario
Timo's Adventure	3	availability	Once the patients arrived, the game shall be ready to be played, instead of being set up or configured in front of the patients.
Timo's Adventure	2	deployability	The target user group is not only the children playing the game, but the medical professionals to observe and to review the log data. The system shall be easily deployed and maintained by these medical professionals in their professional environment.
LawVille	2	deployability	IT infrastructures in Italian schools have large deficits. Game developers have to deal with old machines, network failures, etc..
Una Aventura por el Cauca	3	deployability	The game is easy to execute. Even a child can install it and execute it.
Lix	2	interoperability	There is a networking game mode. The game reads/writes line-based text files and images.

Project	Ranked	Attribute	Scenario
Rice game	1	mobility	Interface is designed in such away that the display scales to the device screen, but is identical on all devices - ie not like the bbc news site, or facebook, where certain tasks are confusing or impossible on the mobile version of the site.
Una Aventura por el Cauca	2	mobility	A teacher or the leader in a playful-centre wishes to execute the game in the tablets provided by the school. These are cheaper and generally more available than desktop PCs
Bosque Interactivo	3	mobility	A product needs to support "x" amount of devices perfectly and have clearly state the requirements for the best functionality of the program.
Lix	1	modifiability	The project is released to the public domain. We use git, C++03, and make, which are very common tools. All library dependencies are open source. The research team behind this survey was able to build and modify my game, adapting it to their needs during research.
LawVille	1	modifiability	A new teaching requirement arises and the game storyboard can be changed in order to meet the new need
Timo's Adventure	3	modifiability	The game's parameters must be easy to modify by a moderator/examiner. The parametes will be effected when the game restart.
Timo's Adventure	1	performance	The game is to identify the time of the user playing with certain game elements in reaction, waiting and being focused. The software must perform well to lower the time taken by other system components.
Bosque Interactivo	2	performance	Performance of the team: When planning a project, multiply the plan by x 2.5. Performance of the system: Needs a very clear concept design to be able to implement it ideally.
Rice game	3	portability	The program should provide the same experience to the end user on each platform with the exception of OS specific interactions - e.g. In windows the double-click for in should be retained, whilst Mac would not require this. Non-expert users get very confused when their OS doesn't behave as expected.
Lix	3	portability	The game builds on Windows, all common Linux distributions, and Mac. I don't write platform-dependent code myself, but use a low-level wrapping library.

Project	Ranked	Attribute	Scenario
Timo's Adventure	1	safety	The game shoudl be designed to be free from violence. And the game must not be placed within the body space of a player.
Code Red Triage	1	testability	User interactions should be predictable or explainable-after-the-fact based on gathered data.
Bosque Interactivo	1	usability	The challenge I have with the previous examples given for this step of the questionnaire are: the expertise level of the team and the expertise of the programmer. In the case of usability, the task can be divided on two aspects: a) what the technology allows the player to do? b) what would be the ideal behavior of the player within learning game to achieve the learning goal? According to where the problem relies we want to have a concept designer and a programmer working together to solve the challenge in the best manner.
Timo's Adventure	2	usability	The game should be designed to be easy enought to understand by a player. There were introductions of each sub-games to be ensure that this requirement was met.
Rice game	2	usability	Target user demographic should be able to complete any given challenge in the game withou repeating the sequence more than 3 times.
Code Red Triage	2	usability	A user should not get stuck and be unsure what to do for a period longer than three minutes. Users should have a degree of autonomy, but the game should be structured in such a way that most users follow the logical path.
Una Aventura por el Cauca	1	variability	A teacher wishes to create new municipalities or add information to the game

Table E.1: Scenarios created by participants

Bibliography

Aalst, W. van der, Beisiegel, M., Hee, K. M. V., Konig, D., & Stahl, C. (2007). A SOA-based architecture framework. *International Journal of Business Process Integration and Management*, *2*(2), 91. doi:10.1504/IJBPIM.2007.015132

Adams, E. & Dormans, J. (2012). *Game mechanics: advanced game design*. New Riders.

Adcock, A. & Eck, R. van. (2012). Adaptive game-based learning. In N. M. Seel (Ed.), *Encyclopedia of the sciences of learning*. Heidelberg: Springer-Verlag. doi:10.1007/978-1-4419-1428-6

Advanced Distributed Learning. (2015). xAPI technical specification. Github repository. Retrieved April 9, 2016, from https://github.com/adlnet/xAPI-Spec

Ahn, R., Barakova, E., Feijs, L., Funk, M., Hu, J., & Rauterberg, M. (2014). Interfacing with adaptive systems. *Automation, Control and Intelligent Systems*, *2*(4), 53–61. doi:10.11648/j.acis.20140204.12

Almerico, G. M. & Baker, R. K. (2004). Bloom's taxonomy illustrative verbs: developing a comprehensive list for educator use. *Florida Association of Teacher Educators Journal*, *1*(4), 1–10.

Amazon Web Services. (2014). AWS game development and operation. Retrieved July 22, 2014, from http://aws.amazon.com/game-hosting/

Amory, A. (2007). Game Object Model version II: a theoretical framework for educational game development. *Educational Technology Research and Development*, *55*(1), 51–77. doi:10.1007/s11423-006-9001-x

Amory, A., Naicker, K., Vincent, J., & Adams, C. (1999). The use of computer games as an educational tool: identification of appropriate game types and game elements. *British Journal of Educational Technology*, *30*(4), 311–321. doi:10.1111/1467-8535.00121

Anderson, L. W., Krathwohl, D. R., & Bloom, B. S. (2001). *A taxonomy for learning, teaching, and assessing: a revision of Bloom's taxonomy of educational objectives*. Longman.

Angelov, S., Grefen, P., & Greefhorst, D. (2012). A framework for analysis and design of software reference architectures. *Information and Software Technology*, *54*(4), 417–431. doi:10.1016/j.infsof.2011.11.009

Angelov, S., Trienekens, J. J. M., & Grefen, P. (2008). Towards a method for the evaluation of reference architectures: experiences from a case. In *Proceedings of the 2nd*

European Conference on Software Architecture – ECSA 2008 (pp. 225–240). Paphos, Cyprus: Springer. doi:10.1007/978-3-540-88030-1_17

Angelov, S., Trienekens, J., & Kusters, R. (2013). Software reference architectures – exploring their usage and design in practice. In *Proceedings of the 7th European Conference on Software Architecture – ECSA 2013* (pp. 17–24). Montpellier, France: Springer. doi:10.1007/978-3-642-39031-9_2

Arnab, S., Lim, T., Carvalho, M. B., Bellotti, F., Freitas, S. de, Louchart, S., Suttie, N., Berta, R., & De Gloria, A. (2015). Mapping learning and game mechanics for serious games analysis. *British Journal of Educational Technology*, *46*(2), 391–411. doi:10.1111/bjet.12113

Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., & Holley, K. (2008). SOMA: a method for developing service-oriented solutions. *IBM Systems Journal*, *47*(3), 377–396. doi:10.1147/sj.473.0377

Arslan, F. (2012). Towards service oriented architecture (SOA) for massive multiplayer online games (MMOG). In *Proceedings of the International Conference on Computer Modelling and Simulation – UKSim* (pp. 538–543). IEEE. doi:10.1109/UKSim.2012.82

Azadegan, A., Riedel, J. C. K. H., & Baalsrud Hauge, J. (2012). Serious games adoption in corporate training. In *Serious games development and applications* (pp. 74–85). Springer. doi:10.1007/978-3-642-33687-4_6

Barros, A., Dumas, M., & Oaks, P. (2006). Standards for web service choreography and orchestration: status and perspectives. In *Revised selected papers of the business process management international workshops – BPM 2005* (pp. 61–74). Nancy, France: Springer. doi:10.1007/11678564_7

Bass, L., Clements, P., & Kazman, R. (2012). *Software architecture in practice* (3rd ed.). SEI Series in Software Engineering. Addison-Wesley.

Batko, M. (2016). Business management simulations – a detailed industry analysis as well as recommendations for the future. *International Journal of Serious Games*, *3*(2). doi:10.17083/ijsg.v3i2.99

Becker, K. (2005). How are games educational? learning theories embodied in games. In *Proceedings of the Digital Games Research Association International Conference – DiGRA 2005: Changing views – worlds in play*.

Bellotti, F., Berta, R., & De Gloria, A. (2010). Designing effective serious games: opportunities and challenges for research. *International Journal of Emerging Technologies in Learning (iJET)*, *5*(SI3), 22–35. doi:10.3991/ijet.v5s3.1500

Bellotti, F., Berta, R., De Gloria, A., D'Ursi, A., & Fiore, V. (2012). A serious game model for cultural heritage. *Journal on Computing and Cultural Heritage*, *5*(4), 1–27. doi:10.1145/2399180.2399185

Bellotti, F., Berta, R., De Gloria, A., & Primavera, L. (2009). Adaptive experience engine for serious games. *IEEE Transactions on Computational Intelligence and AI in Games*, *1*(4), 264–280. doi:10.1109/TCIAIG.2009.2035923

Bellotti, F., Berta, R., De Gloria, A., & Primavera, L. (2010). Supporting authors in the development of task-based learning in serious virtual worlds. *British Journal of Educational Technology*, *41*(1), 86–107. doi:10.1111/j.1467-8535.2009.01039.x

Bellotti, F., Berta, R., De Gloria, A., & Zappi, V. (2008). Exploring gaming mechanisms to enhance knowledge acquisition in virtual worlds. In *Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts – DIMEA 2008* (pp. 77–84). New York, New York, USA: ACM. doi:10.1145/1413634.1413653

Bellotti, F., Kapralos, B., Lee, K., Moreno-Ger, P., & Berta, R. (2013). Assessment in and of serious games: an overview. *Advances in Human-Computer Interaction*, *2013*, 1–11. doi:10.1155/2013/136864

Bente, G. & Breuer, J. (2009). Making the implicit explicit: embedded measurement in serious games. In U. Ritterfield, M. J. Cody, & P. Vorderer (Eds.), *Serious games: mechanisms and effects* (pp. 322–343). New York, NY, USA: Routledge.

Berta, R., Bellotti, F., De Gloria, A., Pranantha, D., & Schatten, C. (2013). Electroencephalogram and physiological signal analysis for assessing flow in games. *IEEE Transactions on Computational Intelligence and AI in Games*, *5*(2), 164–175. doi:10.1109/TCIAIG.2013.2260340

BinSubaih, A. & Maddock, S. (2007). G-factor portability in game development using game engines. In *Proceedings of the 3rd International Conference on Games Research and Development* (pp. 163–170).

Bloom, B. S. (1984). The 2 Sigma problem: the search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, *13*(6), 4–16.

Blunt, R. (2009). Do serious games work? results from three studies. *eLearn*, *2009*(12), 1. doi:10.1145/1661377.1661378

Bogost, I. (2007). *Persuasive games: the expressive power of videogames*. MIT Press.

Boot, W. R., Kramer, A. F., Simons, D. J., Fabiani, M., & Gratton, G. (2008). The effects of video game playing on attention, memory, and executive control. *Acta Psychologica*, *129*(3), 387–398. doi:10.1016/j.actpsy.2008.09.005

Bosch, N., D'Mello, S., Baker, R., Ocumpaugh, J., Shute, V. J., Ventura, M., Wang, L., & Zhao, W. (2015). Automatic detection of learning-centered affective states in the wild. In *Proceedings of the 20th International Conference on Intelligent User Interfaces – IUI 2015*. Association for Computing Machinery (ACM). doi:10.1145/2678025.2701397

Boyle, E. A., Hainey, T., Connolly, T. M., Gray, G., Earp, J., Ott, M., Lim, T., Ninaus, M., Ribeiro, C., & Pereira, J. (2016). An update to the systematic literature review of empirical evidence of the impacts and outcomes of computer games and serious games. *Computers & Education*, *94*, 178–192. doi:10.1016/j.compedu.2015.11.003

Brockmyer, J. H., Fox, C. M., Curtiss, K. A., McBroom, E., Burkhart, K. M., & Pidruzny, J. N. (2009). The development of the Game Engagement Questionnaire: a measure of engagement in video game-playing. *Journal of Experimental Social Psychology*, *45*(4), 624–634. doi:10.1016/j.jesp.2009.02.016

Brooke, J. (1996). SUS – a quick and dirty usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & A. L. McClelland (Eds.), *Usability evaluation in industry*. London: Taylor & Francis.

Brusilovsky, P., Peylo, C. et al. (2003). Adaptive and intelligent web-based educational systems. *International Journal of Artificial Intelligence in Education*, *13*(2-4), 159–172.

Bull, S. & Kay, J. (2007). Student models that invite the learner in: the SMILI :) open learner modelling framework. *International Journal of Artificial Intelligence in Education*, *17*(2), 89–120.

Bull, S. & Kay, J. (2013). Open learner models as drivers for metacognitive processes. In *International handbook of metacognition and learning technologies* (pp. 349–365). Springer. doi:10.1007/978-1-4419-5546-3_23

Bull, S., Kickmeier-Rust, M. D., Vatrapu, R., Johnson, M., Hammermueller, K., Byrne, W., Hernandez-Munoz, L., Giorgini, F., & Meissl-Egghart, G. (2013). Learning, learning analytics, activity visualisation and open learner model: confusing? In D. Hernández-Leo, T. Ley, R. Klamma, & A. Harrer (Eds.), *Scaling up learning for sustained impact SE – 51* (Vol. 8095, pp. 532–535). Lecture Notes in Computer Science. Springer Berlin Heidelberg. doi:10.1007/978-3-642-40814-4_51

Bura, S. (2006). A game grammar. Retrieved February 14, 2014, from http://www.stephanebura.com/diagrams/

Caillois, R. & Barash, M. (1961). *Man, play, and games*. University of Illinois Press.

Carini, R. M., Kuh, G. D., & Klein, S. P. (2006). Student engagement and student learning: testing the linkages. *Research in Higher Education*, *47*(1), 1–32. doi:10.1007/s11162-005-8150-9

Carmines, E. & Zeller, R. (1979). *Reliability and validity assessment*. Quantitative Applications in the Social Sciences. SAGE Publications.

Carvalho, M. B. (2015a). Comparison of two models for serious games analysis. Dataset. Retrieved April 1, 2015, from http://persistent-identifier.nl/?identifier=urn:nbn:nl:ui:13-r5t6-mn

Carvalho, M. B. (2015b). SG Study Github repository. Repository. Retrieved April 1, 2015, from https://github.com/carvalhomb/sgmodels_study

Carvalho, M. B., Bellotti, F., Berta, R., Gloria, A. D., Gazzarata, G., Hu, J., & Kickmeier-Rust, M. D. (2015). A case study on Service-Oriented Architecture for serious games. *Entertainment Computing*, *6*, 1–10. doi:10.1016/j.entcom.2014.11.001

Carvalho, M. B., Bellotti, F., Berta, R., Gloria, A. D., Sedano, C. I., Baalsrud Hauge, J., Hu, J., & Rauterberg, M. (2015). An activity theory-based model for serious games analysis and conceptual design. *Computers and Education*, *87*, 166–181. doi:10.1016/j.compedu.2015.03.023

Carvalho, M. B., Bellotti, F., Hu, J., Baalsrud Hauge, J., Berta, R., Gloria, A. D., & Rauterberg, M. (2015). Towards a Service-Oriented Architecture framework for educational serious games. In *Proceedings of the 15th IEEE International Conference on Advanced Learning Technologies – ICALT 2015*. Hualien, Taiwan.

Carvalho, M. B., Hu, J., Bellotti, F., De Gloria, A., & Rauterberg, M. (2015). Service-oriented architecture (SOA) development for serious games. In *Proceedings of the 14th International Conference on Entertainment Computing – ICEC 2015*. Trondheim, Norway.

Cheng, M.-T., Chen, J.-H., Chu, S.-J., & Chen, S.-Y. (2015). The use of serious games in science education: a review of selected empirical research from 2002 to 2013. *Journal of Computers in Education*, *2*(3), 353–375. doi:10.1007/s40692-015-0039-9

Csikszentmihalyi, M. (1990). *Flow: the psychology of optimal experience*. Harper and Row.

Devane, B. & Squire, K. D. (2012). Activity theory in the learning technologies. In D. H. Jonassen & S. M. Land (Eds.), *Theoretical foundations of learning environments* (Chap. 10, pp. 242–267). New York: Routledge.

Djaouti, D., Alvarez, J., Jessel, J.-P., & Methel, G. (2007). Towards a classification of video games. In *Proceedings of the Artificial and Ambient Intelligence convention (Artificial Societies for Ambient Intelligence) – AISB (ASAMi) 2007*.

Djaouti, D., Alvarez, J., Jessel, J.-P., & Rampnoux, O. (2011). Origins of serious games. In *Serious games and edutainment applications* (pp. 25–43). Springer. doi:10.1007/978-1-4471-2161-9_3

Dormans, J. (2009). Machinations. Retrieved February 25, 2014, from http://www.jorisdormans.nl/machinations/

Doswell, J. & Harmeyer, K. (2007). Extending the 'serious game' boundary: virtual instructors in mobile mixed reality learning games. In *Proceedings of the Digital Games Research Association International Conference – DiGRA 2007* (pp. 524–529).

Dunwell, I. & Freitas, S. de. (2011). Four-dimensional consideration of feedback in serious games. In S. Freitas & P. Maharg (Eds.), *Digital games and learning*. Continuum Publishing.

Eck, R. van. (2006). Digital game-based learning: it's not just the digital natives who are restless... *EDUCAUSE Review*, *41*(2), 1–16. doi:10.1145/950566.950596

Egenfeldt-Nielsen, S. (2006). Overview of research on the educational use of video games. *Digital Kompetanse*, *1*(3), 184–213.

Engeström, Y. (1987). *Learning by expanding: an activity-theoretical approach to developmental research*. Orienta-Konsultit Oy.

Engeström, Y. (2001). Expansive learning at work: toward an activity theoretical reconceptualization. *Journal of Education and Work*, *14*(1), 133–156.

Erhel, S. & Jamet, E. (2013). Digital game-based learning: impact of instructions and feedback on motivation and learning effectiveness. *Computers & Education*, *67*, 156–167. doi:10.1016/j.compedu.2013.02.019

Erl, T. (2005). *Service-Oriented Architecture: concepts, technology, and design*. Upper Saddle River, NJ, USA: Prentice Hall PTR.

Erradi, A., Anand, S., & Kulkarni, N. (2006). SOAF: an architectural framework for service definition and realization. In *Proceedings of the IEEE International Conference on Services Computing – SCC 2006* (pp. 151–158). IEEE.

Facebook. (2014). Games Developer Center. Retrieved October 20, 2014, from https://developers.facebook.com/docs/games/

Fink, L. D. (2003). *Creating significant learning experiences: an integrated approach to designing college courses*. John Wiley & Sons.

Floryan, M. & Woolf, B. P. (2011). Optimizing the performance of educational web services. In *Proceedings of the 11th IEEE International Conference on Advanced Learning Technologies Advanced Learning Technologies – ICALT 2011* (pp. 399–400). doi:10.1109/ICALT.2011.126

Folmer, E. (2007). Component based game development – a solution to escalating costs and expanding deadlines? In *Component-based software engineering* (pp. 66–73). Springer. doi:10.1007/978-3-540-73551-9_5

Foreman, S. (2013). The xAPI and the LMS: what does the future hold? Learning Solutions Magazine. Retrieved April 9, 2016, from http://www.learningsolutionsmag.com/articles/1271/the-xapi-and-the-lms-what-does-the-future-hold

Freitas, S. de & Oliver, M. (2006). How can exploratory learning with games and simulations within the curriculum be most effectively evaluated? *Computers & Education*, *46*(3), 249–264.

Fu, F.-L., Su, R.-C., & Yu, S.-C. (2009). EGameFlow: a scale to measure learners' enjoyment of e-learning games. *Computers & Education*, *52*(1), 101–112. doi:10.1016/j.compedu.2008.07.004

Gagné, R. (1985). *The conditions of learning and theory of instruction*. CBS College Publishing.

Games Enhanced Learning. (2010). Educational game design patterns. Retrieved February 13, 2014, from http://amc.pori.tut.fi/educational-game-design-patterns/

Gardner, J. (Ed.). (2012a). *Assessment and learning* (2nd ed.). SAGE Publications.

Gardner, J. (2012b). Assessment and learning: introduction. In J. Gardner (Ed.), *Assessment and learning* (2nd ed., pp. 1–8). SAGE Publications.

Glahn, C. (2013). Using the ADL experience API for mobile learning, sensing, informing, encouraging, orchestrating. In *Proceedings of the 7th International Conference on Next Generation Mobile Apps, Services and Technologies*. IEEE. doi:10.1109/ngmast.2013.55

Green, C. S. & Bavelier, D. (2007). Action-video-game experience alters the spatial resolution of vision. *Psychological Science*, *18*(1), 88–94. doi:10.1111/j.1467-9280.2007.01853.x

Green, C. S., Li, R., & Bavelier, D. (2010). Perceptual learning during action video game playing. *Topics in Cognitive Science*, *2*(2), 202–216. doi:10.1111/j.1756-8765.2009.01054.x

Guillén-Nieto, V. & Aleson-Carbonell, M. (2012). Serious games and learning effectiveness: the case of It's a Deal! *Computers & Education*, *58*(1), 435–448. doi:10.1016/j.compedu.2011.07.015

Gunter, G. A., Kenny, R. F., & Vick, E. H. (2006). A case for a formal design paradigm for serious games. *The Journal of the International Digital Media and Arts Association*, *3*(1), 93–105.

Guy, E. S. (2005). *From rollout to appropriation: changing practices of development and use during a groupware project* (PhD Thesis, University of Brighton).

Harlen, W. & James, M. (1997). Assessment and learning: differences and relationships between formative and summative assessment. *Assessment in Education: Principles, Policy & Practice*, *4*(3), 365–379. doi:10.1080/0969594970040304

Hartrell, G. (2013). Introducing Google Play game services. Retrieved July 22, 2014, from http://googledevelopers.blogspot.com/2013/05/introducing-google-play-game-services.html

Hasan, H. (1999). Integrating IS and HCI using activity theory as a philosophical and theoretical basis. *Australasian Journal of Information Systems*, *6*(2), 44–55.

Hassan, M. M., Hossain, M. S., Alamri, A., Hossain, M. A., Al-Qurishi, M., Aldukhayyil, Y., & Ahmed, D. T. (2012). A cloud-based serious games framework for obesity. In *Proceedings of the 1st ACM Multimedia International Workshop on Cloud-based*

Multimedia Applications and Services for E-health – CMBAS-EH 2012 (pp. 15–20). New York, USA: ACM. doi:10.1145/2390906.2390912

Haynes, J. A., Underwood, J. S., Pokorny, R., & Spinrad, A. (2014). What is adaptivity? Does it improve performance? In *Foundations of augmented cognition. Advancing human performance and decision-making through adaptive systems* (pp. 224–235). Springer. doi:10.1007/978-3-319-07527-3_21

Heemstra, F. (1992). Software cost estimation. *Information and Software Technology*, *34*(10), 627–639. doi:10.1016/0950-5849(92)90068-z

Houten, S.-P. A. van & Jacobs, P. H. M. (2004). An architecture for distributed simulation games. In *Proceedings of the 36th Conference on Winter Simulation* (pp. 2081–2086).

Hruska, M., Long, R., Amburn, C., Kilcullen, T., & Poeppelman, T. (2014). Experience API and team evaluation: evolving interoperable performance assessment. In *Proceedings of the Interservice/Industry Training, Simulation & Education Conference – I/ITSEC 2014*.

Huizinga, J. (1949). *Homo ludens*. Routledge & Kegan Paul.

Hummels, C., Vinke, D., Frens, J., & Hu, J. (2011). Competency-centered education for designing interactive and intelligent products. *Creation and Design*, *13*(2), 4–17.

Hunicke, R., Leblanc, M., & Zubek, R. (2004). MDA: a formal approach to game design and game research. In *Proceedings of the Challenges in Games AI Workshop, 19th National Conference of Artificial Intelligence* (pp. 1–5). San Jose, California.

Hurwitz, J., Bloor, R., Baroudi, C., & Kaufman, M. (2006). *Service oriented architecture for dummies*. For dummies. Wiley.

Ifenthaler, D., Eseryel, D., & Ge, X. (2012). Assessment for game-based learning. In *Assessment in game-based learning* (pp. 1–8). Springer. doi:10.1007/978-1-4614-3546-4_1

IJsselsteijn, W., De Kort, Y., Poels, K., Jurgelionis, A., & Bellotti, F. (2007). Characterising and measuring user experiences in digital games. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology* (Vol. 2, p. 27).

Illinois Central College. (2011). Revised Bloom's taxonomy. Retrieved July 30, 2014, from http://www.icc.edu/innovation/PDFS/assessmentEvaluation/RevisedBloomsChart_bloomsverbsmatrix.pdf

Ionita, M. T., Hammer, D. K., & Obbink, H. (2002). Scenario-based software architecture evaluation methods: an overview. In *Proceedings of the Workshop on Methods and Techniques for Software Architecture Review and Assessment (SARA) at the International Conference on Software Engineering – ICSE 2002*. Orlando, Florida, USA.

Islas Sedano, C. (2012). *Hypercontextualized games* (Doctoral dissertation, University of Eastern Finland).

Jackson, L. A., Witt, E. A., Games, A. I., Fitzgerald, H. E., Eye, A. von, & Zhao, Y. (2012). Information technology use and creativity: findings from the children and technology project. *Computers in Human Behavior*, *28*(2), 370–376. doi:10.1016/j.chb.2011.10.006

Jennett, C., Cox, A. L., Cairns, P., Dhoparee, S., Epps, A., Tijs, T., & Walton, A. (2008). Measuring and defining the experience of immersion in games. *International Journal of Human-Computer Studies*, *66*(9), 641–661. doi:10.1016/j.ijhcs.2008.04.004

Jonassen, D. H. & Rohrer-Murphy, L. (1999). Activity theory as a framework for designing constructivist learning environments. *Educational Technology Research and Development*, *47*(1), 61–79. doi:10.1007/BF02299477

Kaptelinin, V. (1996). Context and consciousness: activity theory and human-computer interaction. In B. A. Nardi (Ed.), (Chap. Activity theory: Implications for human-computer interaction, pp. 103–116). The MIT Press Cambridge, MA.

Kaptelinin, V. & Nardi, B. A. (2006). *Acting with technology: activity theory and interaction design*. Cambridge, MA, USA: The MIT Press.

Kazman, R., Klein, M., & Clements, P. (2000). *ATAM: method for architecture evaluation* (tech. rep. No. CMU/SEI-2000-TR-004). Product Line Systems. Retrieved from http://www.sei.cmu.edu/reports/00tr004.pdf

Kebritchi, M., Hirumi, A., & Bai, H. (2010). The effects of modern mathematics computer games on mathematics achievement and class motivation. *Computers & Education*, *55*(2), 427–443. doi:10.1016/j.compedu.2010.02.007

Keller, J. M. (1987). Development and use of the ARCS model of instructional design. *Journal of Instructional Development*, *10*(3), 2–10.

Kevan, J. M. & Ryan, P. R. (2016). Experience API: flexible, decentralized and activity-centric data collection. *Technology, Knowledge and Learning*, *21*(1), 143–149. doi:10.1007/s10758-015-9260-x

Kickmeier-Rust, M. D. & Albert, D. (2007). The ELEKTRA ontology model: a learner-centered approach to resource description. In *6th International Conference Advances in Web Based Learning – ICWL 2007 – revised papers* (pp. 78–89). Edinburgh, UK. doi:10.1007/978-3-540-78139-4_8

Kickmeier-Rust, M. D. & Albert, D. (2012a). *An alien's guide to multi-adaptive educational computer games* (M. D. Kickmeier-Rust & D. Albert, Eds.). Santa Rosa, CA: Informing Science Press.

Kickmeier-Rust, M. D. & Albert, D. (2012b). Educationally adaptive: balancing serious games. *International Journal of Computer Science in Sport*, *11*(1), 1–10.

Kickmeier-Rust, M. D., Marte, B., Linek, S. B., Lalonde, T., & Albert, D. (2008). The effects of individualized feedback in digital educational games. In *Proceedings of the 2nd European Conference on Games Based Learning – ECGBL 2008* (pp. 227–236).

Kickmeier-Rust, M. D., Peirce, N., Conlan, O., Schwarz, D., Verpoorten, D., & Albert, D. (2007). Immersive digital games: the interfaces for next-generation e-learning? In *Proceedings of the 4th International Conference on Universal Access in Human-Computer Interaction – UAHCI 2007* (pp. 647–656). Beijing, China. doi:10.1007/978-3-540-73283-9_71

Kiili, K. (2005). Digital game-based learning: towards an experiential gaming model. *The Internet and higher education*, *8*(1), 13–24.

Kiili, K. (2010). Call for learning-game design patterns. In *Educational games: design, learning, and applications*. Nova Publishers.

Kim, C.-H., Weston, R. H., Hodgson, A., & Lee, K.-H. (2003). The complementary use of IDEF and UML modelling approaches. *Computers in Industry*, *50*(1), 35–56. doi:10.1016/S0166-3615(02)00145-8

Kivikangas, J. M., Chanel, G., Cowley, B., Ekman, I., Salminen, M., Järvelä, S., & Ravaja, N. (2011). A review of the use of psychophysiological methods in game research. *Journal of Gaming & Virtual Worlds*, *3*(3), 181–199. doi:10.1386/jgvw.3.3.181_1

Kivikangas, J. M., Nacke, L., & Ravaja, N. (2011). Developing a triangulation system for digital game events, observational video, and psychophysiological data to study emotional responses to a virtual character. *Entertainment Computing*, *2*(1), 11–16. doi:10.1016/j.entcom.2011.03.006

Knight, J. F., Carley, S., Tregunna, B., Jarvis, S., Smithies, R., Freitas, S. de, Dunwell, I., & Mackway-Jones, K. (2010). Serious gaming technology in major incident triage training: a pragmatic controlled trial. *Resuscitation*, *81*(9), 1175–9. doi:10.1016/j.resuscitation.2010.03.042

Kolb, D. A. (1984). The process of experiential learning. In *Experiential learning: experience as the source of learning and development* (pp. 20–38). Englewood Cliffs, NJ: Prentice-Hall. doi:10.1016/B978-0-7506-7223-8.50017-4

Koster, R. (2005a). A grammar of gameplay. Game atoms: can games be diagrammed. Retrieved January 1, 2014, from http://theoryoffun.com/grammar/gdc2005.htm

Koster, R. (2005b). *Theory of fun for game design*. Paraglyph Series. O'Reilly Media.

Koster, R. (2011). Social mechanics – the engines behind everything multiplayer. Retrieved July 31, 2014, from http://www.raphkoster.com/gaming/gdco2010/socialmechanics.pdf

Kuhl, F., Weatherly, R., & Dahmann, J. (1999). *Creating computer simulation systems: an introduction to the high level architecture*. Prentice Hall PTR.

Kuutti, K. (1995). Activity theory as a potential framework for human-computer interaction research. In B. Nardi (Ed.), *Context and consciousness: activity theory and human-computer interaction* (pp. 17–44). Cambridge: MIT Press.

Law, C. Y., Grundy, J., Cain, A., & Vasa, R. (2015). A preliminary study of open learner model representation formats to support formative assessment. In *Proceedings of the 39th IEEE Annual Computer Software and Applications Conference*. IEEE. doi:10.1109/compsac.2015.112

Leahy, S. & Wiliam, D. (2012). From teachers to schools: scaling up professional development for formative assessment. In *Assessment and learning* (2nd ed., pp. 49–71). SAGE Publications.

Leont'ev, A. N. (1978). *Activity, consciousness and personality*. Englewood Cliffs, NJ: Prentice-Hall.

Lewis, M. & Jacobson, J. (2002). Games engines in scientific research. *Communications of the ACM*, *45*(1), 27–31.

Liu, E. Z. F. & Lin, C. H. (2009, January). Developing evaluative indicators for educational computer games. *British Journal of Educational Technology*, *40*(1), 174–178. doi:10.1111/j.1467-8535.2008.00852.x

Liu, J. H. (2012). DragonBox: algebra beats Angry Birds. Retrieved July 31, 2014, from http://archive.wired.com/geekdad/2012/06/dragonbox/

Loh, C. S. (2012). Information trails: in-process assessment of game-based learning. In D. Ifenthaler, D. Eseryel, & X. Ge (Eds.), *Assessment in game-based learning: Foundations, innovations, and perspectives* (pp. 123–144). Springer. doi:10.1007/978-1-4614-3546-4_8

Lumos. (2014). Lumos Powered. Retrieved July 22, 2014, from https://www.lumospowered.com

Mandryk, R. L., Inkpen, K. M., & Calvert, T. W. (2006). Using psychophysiological techniques to measure user experience with entertainment technologies. *Behaviour & Information Technology*, *25*(2), 141–158. doi:10.1080/01449290500331156

Marne, B., Wisdom, J., Huynh-Kim-Bang, B., & Labat, J.-M. (2012). The six facets of serious game design: a methodology enhanced by our design pattern library. In *Proceedings of 7th European Conference of Technology Enhanced Learning – EC-TEL 2012* (Vol. 7563, pp. 208–221). Saarbrücken, Germany. doi:10.1007/978-3-642-33263-0_17

Marsh, T. (2006). Game development for experience through staying there. In *Proceedings of the ACM SIGGRAPH symposium on videogames – Sandbox 2006* (Vol. 1, *212*, pp. 83–89).

Marsh, T. (2010). Activity-based scenario design, development and assessment in serious games. *Gaming and cognition: Theories and practice from the learning sciences*, 213–225.

Marsh, T. & Nardi, B. A. (2014). Spheres and lenses: activity-based scenario / narrative approach for design and evaluation of entertainment through engagement. In *Proceedings of the 13th International Conference on Entertainment Computing – ICEC 2014* (pp. 42–51). Springer.

Maxwell, E. (2006). Open standards, open source, and open innovation: harnessing the benefits of openness. *Innovations: Technology, Governance, Globalization*, *1*(3), 119–176. doi:10.1162/itgg.2006.1.3.119

Megliola, M., De Vito, G., Sanguini, R., Wild, F., & Lefrere, P. (2014). Creating awareness of kinaesthetic learning using the experience API: current practices, emerging challenges, possible solutions. In *Proceedings of the 4th workshop on Awareness and Reflection in Technology-Enhanced Learning, in conjunction with the 9th European Conference on Technology Enhanced Learning – ECTEL 2014)* (Vol. 1238, pp. 11–22). Retrieved from http://oro.open.ac.uk/41697/

Mislevy, R. J., Behrens, J. T., Dicerbo, K. E., Frezzo, D. C., & West, P. (2012). Three things game designers need to know about assessment. In D. Ifenthaler, D. Eseryel, & X. Ge (Eds.), *Assessment in game-based learning: Foundations, innovations, and perspectives* (pp. 59–81). Springer. doi:10.1007/978-1-4614-3546-4_5

Mislevy, R. J., Steinberg, L. S., & Almond, R. G. (2003). Focus article: on the structure of educational assessments. *Measurement: Interdisciplinary Research & Perspective*, *1*(1), 3–62. doi:10.1207/s15366359mea0101_02

Moreno-Ger, P., Sierra, J. L., Martínez-Ortiz, I., & Fernández-Manjón, B. (2007). A documental approach to adventure game development. *Science of Computer Programming*, *67*(1), 3–31. doi:10.1016/j.scico.2006.07.003

Naarmann, S. (2011). Lix. https://github.com/SimonN/Lix. GitHub.

Newman, S. (2015). *Building microservices*. O'Reilly Media.

Nicol, D. J. & Macfarlane-Dick, D. (2006). Formative assessment and self-regulated learning: a model and seven principles of good feedback practice. *Studies in Higher Education*, *31*(2), 199–218. doi:10.1080/03075070600572090

Ning, J. (1996). A component-based software development model. In *Proceedings of the 20th International Computer Software and Applications Conference – COMPSAC 1996*. IEEE. doi:10.1109/cmpsac.1996.544597

OATH. (2007). *OATH reference architecture, release 2.0*. Initiative for Open AuTHentication.

Object Management Group. (2015). *Unified Modeling Language version 2.5* (tech. rep. No. Formal/15-03-01). Object Management Group (OMG).

Oei, A. C. & Patterson, M. D. (2014). Playing a puzzle video game with changing requirements improves executive functions. *Computers in Human Behavior*, *37*, 216–228. doi:10.1016/j.chb.2014.04.046

Oliveira, V., Coelho, A., Guimarães, R., & Rebelo, C. (2012). Serious game in security: a solution for security trainees. *Procedia Computer Science*, *15*, 274–282. doi:10.1016/j.procs.2012.10.079

Oostendorp, H. van, Spek, E. D. van der, & Linssen, J. (2014). Adapting the complexity level of a serious game to the proficiency of players. *EAI Endorsed Transactions on Game-Based Learning*, *1*(2), e5. doi:10.4108/sg.1.2.e5

Papazoglou, M. P. & Heuvel, W.-J. van den. (2006). Service-oriented design and development methodology. *International Journal of Web Engineering and Technology*, *2*(4), 412–442.

Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2007). Service-oriented computing: state of the art and research challenges. *Computer*, *40*(11), 38–45. doi:10.1109/MC.2007.400

Paraskeva, F., Mysirlaki, S., & Papagianni, A. (2010). Multiplayer online games as educational tools: facing new challenges in learning. *Computers & Education*, *54*(2), 498–505. doi:10.1016/j.compedu.2009.09.001

Peachey, P. (2010). The application of 'activity theory' in the design of educational simulation games. In *Design and implementation of educational games: theoretical and practical perspectives* (1988, pp. 154–167). IGI Global. doi:10.4018/978-1-61520-781-7.ch011

Peltz, C. (2003). Web services orchestration and choreography. *Computer*, *36*(10), 46–52.

Popescu, M. M., Romero, M., & Usart, M. (2012). Using serious games in adult education serious business for serious people-the metavals game case study. In *Proceedings of the 7th International Conference on Virtual Learning – ICVL 2012* (pp. 68–72).

Pranantha, D., Bellotti, F., Berta, R., & De Gloria, A. (2012). Puzzle-it: an HTML5 serious games platform for education. In S. Göbel, W. Müller, B. Urban, & J. Wiemeyer (Eds.), *E-learning and games for training, education, health and sports* (Vol. 7516, pp. 134–143). Lecture Notes in Computer Science. Springer Berlin Heidelberg. doi:10.1007/978-3-642-33466-5_15

Prensky, M. (2001). *Digital game-based learning*. New York: McGraw-Hill.

Qazdar, A., Cherkaoui, C., Er-Raha, B., & Mammass, D. (2015). AeLF: mixing adaptive learning system with learning management system. *International Journal of Computer Applications*, *119*(15).

RAGE. (2016). RAGE – realizing an applied gaming ecosystem. Website. Retrieved August 1, 2016, from http://rageproject.eu

Rauterberg, M. & Felix, D. (1996). Human errors: disadvantages and advantages. In *Proceedings of the 4th Pan Pacific Conference on Occupational Ergonomics – PPCOE 1996* (pp. 25–28). Ergonomics Society Taiwan. Hsinchu.

Reed, P. (2002). Reference architecture: the best of best practices. Retrieved January 28, 2016, from http://www.ibm.com/developerworks/rational/library/2774.html

Rice, J. W. (2007). New media resistance: barriers to implementation of computer video games in the classroom. *Journal of Educational Multimedia and Hypermedia*, *16*(3), 249.

Ricordel, P.-M. & Demazeau, Y. (2000). From analysis to deployment: a multi-agent platform survey. In *Revised papers of the 1st International Workshop Engineering Societies in the Agents World – ESAW 2000* (pp. 93–105). Berlin, Germany. doi:10.1007/3-540-44539-0_7

Riedel, J. C. K. H., Feng, Y., Hauge, J. M. B., Hansen, P. K., & Tasuya, S. (2015). The adoption and application of serious games in corporate training – the case of manufacturing. In *2015 IEEE International Conference on Engineering, Technology and Innovation/ International Technology Management Conference (ICE/ITMC)*. doi:10.1109/ice.2015.7438684

Rustici Software. (2016). Who's using the Tin Can API? Webpage. Retrieved April 9, 2016, from http://experienceapi.com/adopters/

Sabourin, J. L. & Lester, J. C. (2014). Affect and engagement in Game-Based Learning environments. *IEEE Transactions on Affective Computing*, *5*(1), 45–56. doi:10.1109/t-affc.2013.27

Sadler, D. R. (1989). Formative assessment and the design of instructional systems. *Instructional Science*, *18*(2), 119–144. doi:10.1007/bf00117714

Sadler, D. R. (1998). Formative assessment: revisiting the territory. *Assessment in Education: Principles, Policy & Practice*, *5*(1), 77–84. doi:10.1080/0969595980050104

Salen, K. & Zimmerman, E. (2004). *Rules of play: game design fundamentals*. MIT press.

Sawyer, B. & Rejeski, D. (2002). *Serious games: improving public policy through game-based learning and simulation*. Woodrow Wilson International Center for Scholars. Washington, DC.

Scacchi, W. & Cooper, K. M. (2015). Research challenges at the intersection of computer games and software engineering. In *Proceedings of the conference Foundations of Digital Games – FDG*. Pacific Grove, CA.

Schell, J. (2008). *The art of game design: a book of lenses*. Morgan Kaufmann. Morgan Kaufmann.

Schmidt, R., Emmerich, K., & Schmidt, B. (2015). Applied games – in search of a new definition. In *Proceedings of the 14th International Conference on Entertainment Computing – ICEC 2015* (pp. 100–111). Trondheim, Norway. doi:10.1007/978-3-319-24589-8_8

Serious Games Society. (2013a). Serious games web services catalog. Retrieved April 4, 2014, from http://services.seriousgamessociety.org/

Serious Games Society. (2013b). SG knowledge management system. Retrieved July 30, 2014, from http://studies.seriousgamessociety.org/

Serrano-Laguna, Á., Martínez-Ortiz, I., Haag, J., Regan, D., Johnson, A., & Fernández-Manjón, B. (2017, February). Applying standards to systematize learning analytics in serious games. *Computer Standards & Interfaces*, *50*, 116–123. doi:10.1016/j.csi.2016.09.014

Serrano-Laguna, Á., Torrente, J., Moreno-Ger, P., & Fernández-Manjón, B. (2014). Application of learning analytics in educational videogames. *Entertainment Computing*, *5*(4), 313–322. doi:10.1016/j.entcom.2014.02.003

Shaikh, A., Sahu, S., Rosu, M.-C., Shea, M., & Saha, D. (2006). On demand platform for online games. *IBM Systems Journal*, *45*(1), 7–19. doi:10.1147/sj.451.0007

Shute, V. J. (2008). Focus on formative feedback. *Review of Educational Research*, *78*(1), 153–189. doi:10.3102/0034654307313795

Shute, V. J., D'Mello, S., Baker, R., Cho, K., Bosch, N., Ocumpaugh, J., Ventura, M., & Almeda, V. (2015). Modeling how incoming knowledge, persistence, affective states, and in-game progress influence student learning from an educational game. *Computers & Education*, *86*, 224–235. doi:10.1016/j.compedu.2015.08.001

Shute, V. J. & Ke, F. (2012). Games, learning, and assessment. In D. Ifenthaler, D. Eseryel, & X. Ge (Eds.), *Assessment in game-based learning: Foundations, innovations, and perspectives* (pp. 43–58). Springer. doi:10.1007/978-1-4614-3546-4_4

Shute, V. J. & Kim, Y. J. (2013). Formative and stealth assessment. In *Handbook of research on educational communications and technology* (pp. 311–321). Springer. doi:10.1007/978-1-4614-3185-5_25

Shute, V. J., Ventura, M., Bauer, M., & Zapata-Rivera, D. (2009). Melding the power of serious games and embedded assessment to monitor and foster learning. In *Serious games: mechanisms and effects* (Vol. 2, pp. 295–321). Philadelphia, PA: Routledge/LEA.

Shute, V. J., Ventura, M., & Ke, F. (2015, January). The power of play: the effects of portal 2 and lumosity on cognitive and noncognitive skills. *Computers & Education*, *80*, 58–67. doi:10.1016/j.compedu.2014.08.013

Sicart, M. (2008). Defining game mechanics. *Game Studies*, *8*(2). Retrieved July 31, 2014, from http://gamestudies.org/0802/articles/sicart

Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I., & Postma, E. (2006). Adaptive game AI with dynamic scripting. *Machine Learning*, *63*(3), 217–248.

Sprott, D. & Wilkes, L. (2004). Understanding service-oriented architecture. *The Architecture Journal*, *1*(1), 10–17.

Staalduinen, J.-P. van & Freitas, S. de. (2011). A game-based learning framework: linking game design and learning. In M. S. Khine (Ed.), *Learning to play: exploring the future of education with video games* (pp. 29–54). Peter Lang. doi:10.3726/978-1-4539-0084-0

Stanescu, I. A., Stanescu, A. M., Moisescu, M., Sacala, I. S., Stefan, A., & Baalsrud Hauge, J. (2014). Enabling interoperability between serious game and virtual engineering ecosystems. In *Proceedings of the ASME international design engineering technical conferences and computers and information in engineering conference*. Buffalo, New York, USA. doi:10.1115/DETC2014-35418

Steiner, C. M., Kickmeier-Rust, M. D., Mattheiss, E., Göbel, S., & Albert, D. (2012). Balancing on a high wire: adaptivity key factor of future learning games. In *An alien's guide to multi-adaptive educational computer games* (p. 43). Informing Science.

Stevens, M. (2005). Understanding Service-Oriented Architecture. Retrieved June 27, 2014, from http://www.developer.com/design/article.php/10925_2207371_3/Understanding-Service-Oriented-Architecture.htm

Streicher, A. & Roller, W. (2015). Towards an interoperable adaptive tutoring agent for simulations and serious games. In *Proceedings of the 9th Multi Conference on Computer Science and Information Systems – MCCSIS 2015* (pp. 21–24).

Susi, T., Johannesson, M., & Backlund, P. (2007). *Serious games – an overview*. University of Skövde, School of Humanities and Informatics. Skövde, Sweden.

Sweetser, P. & Wyeth, P. (2005). Gameflow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)*, *3*(3), 3–3.

Terenzini, P. T. (1989). Assessment with open eyes: pitfalls in studying student outcomes. *The Journal of Higher Education*, *60*(6), 644. doi:10.2307/1981946

The Open Group. (2011). *TOGAF version 9.1*. The Open Group. Retrieved January 26, 2016, from http://www.opengroup.org/architecture/togaf9-doc/arch

Tijs, T. J. W., Brokken, D., & IJsselsteijn, W. A. (2008). Dynamic game balancing by recognizing affect. In *Fun and games* (pp. 88–93). Springer. doi:10.1007/978-3-540-88322-7_9

Torrente, J., Blanco, Á. D., Moreno-Ger, P., Martínez-Ortiz, I., & Fernández-Manjón, B. (2009). Implementing accessibility in educational videogames with <e-adventure>. In *Proceedings of the 1st ACM international workshop on Multimedia Technologies for Distance Learning – MTDL 2009* (pp. 57–66). New York, NY, USA. doi:10.1145/1631111.1631122

Vahdat, M., Carvalho, M. B., Funk, M., Rauterberg, M., Hu, J., & Anguita, D. (2016). Learning analytics for a puzzle game to discover the puzzle-solving tactics of players. In *Proceedings of the 11th European Conference on Technology Enhanced Learning – EC-TEL 2016*. Lyon, France.

Vegt, W. van der, Westera, W., Nyamsuren, E., Georgiev, A., & Martínez-Ortiz, I. (2016). RAGE architecture for reusable serious gaming technology components. *International Journal of Computer Games Technology*, *2016*, 1–10. doi:10.1155/2016/5680526

Ventura, M., Shute, V. J., & Kim, Y. J. (2012). Video gameplay, personality and academic performance. *Computers & Education*, *58*(4), 1260–1266. doi:10.1016/j.compedu.2011.11.022

WeWantToKnow. (2012). DragonBox. Retrieved July 31, 2014, from http://dragonboxapp.com

Wiebe, E. N., Lamb, A., Hardy, M., & Sharek, D. (2014). Measuring engagement in video game-based environments: investigation of the user engagement scale. *Computers in Human Behavior*, *32*, 123–132. doi:10.1016/j.chb.2013.12.001

Wiliam, D. & Black, P. (1996). Meanings and consequences: a basis for distinguishing formative and summative functions of assessment? *British Educational Research Journal*, *22*(5), 537–548. doi:10.1080/0141192960220502

Wouters, P., Nimwegen, C. van, Oostendorp, H. van, & Spek, E. D. van der. (2013). A meta-analysis of the cognitive and motivational effects of serious games. *Journal of Educational Psychology*, *105*(2), 249.

Wu, H. (2002). *A reference architecture for adaptive hypermedia applications* (Doctoral dissertation, Eindhoven University of Technology).

Yorke, M. (2001). Formative assessment and its relevance to retention. *Higher Education Research and Development*, *20*(2), 115–126. doi:10.1080/758483462

Zagal, J. P., Mateas, M., Fernández-Vara, C., Hochhalter, B., & Lichti, N. (2005). Towards an ontological language for game analysis. In *Proceedings of the Digital Games Research Association International Conference – DiGRA 2005: Changing views – worlds in play*.

Zaphiris, P., Wilson, S., & Ang, C. S. (2010). Computer games and sociocultural play: an activity theoretical perspective. *Games and Culture*, *5*(4), 354–380. doi:10.1177/1555412009360411

Acknowledgement

This dissertation is the result of work carried out in two partner universities, in two different countries, and under the guidance of four supervisors.

I am forever grateful to my supervisors, who were always helpful and supportive of my work throughout this PhD. At the University of Genoa, prof. Francesco Bellotti was a tireless mentor, always available to discuss my work and my ideas. His excitement for research is contagious, and it certainly affected my PhD experience in a very positive way. I would also like to thank my first promotor, prof. Alessandro De Gloria, for giving me the chance to get involved in the Games and Learning Alliance project, which was an incredible professional opportunity. At the Eindhoven University of Technology, I was lucky to study under the supervision of my second promotor, prof. Matthias Rauterberg, whose knowledge and experience were invaluable sources of inspiration. Sometimes I would get too tangled in the technicalities, but Matthias always made sure I did not lose sight of the very human dimension of my work. My deepest gratitude to prof. Jun Hu, whose advice and remarks were always thoughtful and spot on. Jun helped me craft a coherent story from the work carried out throughout these years. He also helped me remain calm and focused during the toughest times.

During these years, I had the privilege to meet and work with amazing researchers. Jannicke Baalsrud Hauge, Carolina Islas Sedano, Michael Kickmeier-Rust, Erik van der Spek, Ioana Stanescu and Riccardo Berta, I am indebted to you for all the fruitful and inspiring discussions, without which this thesis would not be possible. Mehrnoosh Vahdat, thank you for the opportunity to work with you. I think we made a great research team! To my colleagues at Unige, at TU/e, and from the ICEPhD program, thank you for the research and non-research discussions and companionship throughout these years. It was a privilege to meet you all!

The story of this PhD started many, many years ago, with my admiration for both my parents' academic careers in the field of technology for education. I am grateful to my mother, prof. Fátima Brandão, for her infinite wisdom, patience, love, and unrelenting support. She is my role model! My gratitude also to my father, prof. Marco Carvalho, who has always been an inspiration, in every way possible.

Krešimir Dabčević supported me by being not only my caring and loving life partner, but also my unofficial research mate. He listened to me when I needed to talk about my ideas, even before they made any sense. He helped me with my manuscripts, proof-reading and giving me his always insightful comments. He cheered with me for my achievements and was my rock during the hard days. His companionship was the best gift I could ever wish for.

I would like to extend my gratitude to my siblings, Fernanda, Pedro, and Milla, for their love and companionship; to Luciano, for his support and enthusiasm; to Deborah, for her care and encouragement; and to my grandmother, Dona Edith, the family matriarch and our greatest inspiration. Finally, to all my friends and family, some close and some far away, but all of them always in my heart: thank you!

Publications

Journal articles

Arnab, S., Lim, T., Carvalho, M. B., Bellotti, F., Freitas, S. de, Louchart, S., Suttie, N., Berta, R., & De Gloria, A. (2015). Mapping learning and game mechanics for serious games analysis. *British Journal of Educational Technology*, *46*(2), 391–411. doi:10.1111/bjet.12113

Carvalho, M. B., Bellotti, F., Berta, R., Gloria, A. D., Gazzarata, G., Hu, J., & Kickmeier-Rust, M. D. (2015). A case study on Service-Oriented Architecture for serious games. *Entertainment Computing*, *6*, 1–10. doi:10.1016/j.entcom.2014.11.001

Carvalho, M. B., Bellotti, F., Berta, R., Gloria, A. D., Sedano, C. I., Baalsrud Hauge, J., Hu, J., & Rauterberg, M. (2015). An activity theory-based model for serious games analysis and conceptual design. *Computers and Education*, *87*, 166–181. doi:10.1016/j.compedu.2015.03.023

Baalsrud Hauge, J., Bellotti, F., Nadolski, R., Berta, R., & Carvalho, M. B. (2014). Deploying serious games for management in higher education: lessons learned and good practices. *EAI Endorsed Transactions on Serious Games*, *14*(3), 1–12. doi:10.4108/sg.1.3.e4

Baalsrud Hauge, J., Bellotti, F., Berta, R., Carvalho, M. B., De Gloria, A., Lavagnino, E., Nadolski, R., & Ott, M. (2013). Field assessment of serious games for entrepreneurship in higher education. *Journal of Convergence Information Technology*, *8*(13), 1–12.

Conference proceedings

Vahdat, M., Carvalho, M. B., Funk, M., Rauterberg, M., Hu, J., & Anguita, D. (2016). Learning analytics for a puzzle game to discover the puzzle-solving tactics of players. In *Proceedings of the 11th European Conference on Technology Enhanced Learning – EC-TEL 2016*. Lyon, France.

Baalsrud Hauge, J., Stanescu, I. A., Carvalho, M. B., Stefan, A., Banica, M., & Lim, T. (2015). Integrating SG in VES to support knowledge processes. In *Proceedings of the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference – IDETC/CIE 2015*.

Baalsrud Hauge, J., Stanescu, I., Carvalho, M. B., Lim, T., & Arnab, S. (2015). Serious Games Mechanics and opportunities for reuse. In *Proceedings of the 11th eLearning and Software for Education Conference – eLSE 2015*. Bucharest, Romania.

Carvalho, M. B., Bellotti, F., Hu, J., Baalsrud Hauge, J., Berta, R., Gloria, A. D., & Rauterberg, M. (2015). Towards a Service-Oriented Architecture framework for educational serious games. In *Proceedings of the 15th IEEE International Conference on Advanced Learning Technologies – ICALT 2015*. Hualien, Taiwan.

Carvalho, M. B., Hu, J., Bellotti, F., De Gloria, A., & Rauterberg, M. (2015). Service-oriented architecture (SOA) development for serious games. In *Proceedings of the 14th International Conference on Entertainment Computing – ICEC 2015*. Trondheim, Norway.

Lim, T., Louchart, S., Suttie, N., Baalsrud Hauge, J., Stanescu, I. A., Bellotti, F., Carvalho, M. B., Earp, J., Ott, M., Arnab, S., & Brown, D. (2014). Serious game mechanics, workshop on the ludo-pedagogical mechanism. In *Games for training, education, health and sports* (pp. 186–189). Springer.

Lim, T., Louchart, S., Suttie, N., Baalsrud Hauge, J., Stanescu, I. A., Ortiz, I. M., Moreno-Ger, P., Bellotti, F., Carvalho, M. B., Earp, J., Ott, M., Arnab, S., & Berta, R. (2014). Narrative serious game mechanics (NSGM)–insights into the narrative-pedagogical mechanism. In *Games for training, education, health and sports* (pp. 23–34). Springer.

Islas Sedano, C., Carvalho, M. B., Secco, N., & Longstreet, C. S. (2013). Collaborative and cooperative games: facts and assumptions. In *Proceedings of the International Conference on Collaboration Technologies and Systems – CTS 2013* (pp. 370–376).

Dataset

Carvalho, M. B. (2015). Comparison of two models for serious games analysis. Dataset. Retrieved April 1, 2015, from http://persistent-identifier.nl/?identifier=urn:nbn:nl:ui:13-r5t6-mn

Curriculum Vitae

Maira Brandao Carvalho was born on February 1st, 1983, in Brasilia, Brazil. She received her Bachelor's degree in Communications (Advertising and Journalism) from the University of Brasilia, Brazil, in 2004. Later, in 2011, she received her M. Sc. degree in Interactive Technology from the University of Tampere, Finland, where her work and research focused on the development of applications for low-literacy users.

She has more than eleven years of experience as information architect, interface designer and web developer. She has worked for companies and organizations such as WWF-Brazil, Agência Click, Brazil's National Supply Company (CONAB), and at the headquarters of the International Labour Organization (ILO), in Geneva, Switzerland.

Since October 2012, she has been a doctoral candidate in the Erasmus Mundus Joint Doctorate in Interactive and Cognitive Environments. She spent the first two years of her doctoral studies at the Dipartimento di Ingegneria Navale, Elettrica, Elettronica e delle Telecomunicazioni of the Università degli Studi di Genova, Italy. During the second half of her studies, she was a member of the Designed Intelligence research group, at the Department of Industrial Design of Eindhoven University of Technology, The Netherlands. In her PhD project, she focused on the design and development of serious games, particularly on how to enhance games with centralized user profiles, learning analytics and automatic adaptation to players' skills and affective states.