Journal of
Virtual Worlds Research
jvwresearch.org ISSN: 1941-8477

Assembled 2016

April 2016 (Part 1)
Volume 9 No. 1

Volume 9, Number 1
Assembled 2016 (Part 1)

April 2016

Journal of Virtual Worlds Research
jvwresearch.org ISSN: 1941-8477

Volume 9, Number 1
Assembled 2016 (1)
April, 2016

Gamecloud – A Platform for Connecting Video Games

Janne Parkkila
Lappeenranta University of Technology, Finland

Kati Järvi
Hanken School of Economics, Finland

Timo Hynninen **Jouni Ikonen** **Jari Porras**
Lappeenranta University of Technology, Finland

Abstract

The growing video game markets, especially the mobile market, have caused problems in terms of new game products being found by players. Cross-promotion and in-game advertising have been used to promote video games inside each other. However, the digital nature of video games as an interactive medium enables deeper collaboration between video games, and could be used in a more profound manner. We interviewed Finnish video game companies to understand if they were interested in creating deeper collaboration between their game products, and how such collaboration could take place. Based on the results, we built a platform called Gamecloud for connecting games together. We present the platform architecture and demonstrate it in use, with examples of connecting games with other games and connecting games with the real world, alongside an example of physical exercising.

Introduction

The video game industry has seen significant growth in the last decade, mainly due to the introduction of mobile gaming and digital distribution (Sotamaa & Karppi, 2010). Even though digital application stores receive hundreds of new games every month, only a small portion of games bring profits to their creators. One of the most important tasks for game companies is to tackle the problem of visibility – how players can find the game, when it is competing with a huge number of other video games (Prata, de Moraes, & Quaresma, 2012). One possible solution to the discovery problem of video games is the cross-promotion of video games. Marketing professionals have recognized both co-branding (Washburn, Till, & Priluck, 2000) and cross-promotion (Tang, Newton, & Wang, 2007) as feasible approaches in other fields of business.

A good example of joining multiple franchises to create a success story is a famous video game called *Kingdom Hearts* (http://www.kingdomhearts.com/) by Square Enix (http://www.square-enix.com) and Disney (http://disney.com). The game combines two different story universes into a single game. *Kingdom Hearts* introduces a world where Disney's Mickey Mouse, Donald Duck and other characters meet with the famous Final Fantasy characters of Square Enix. The game was a success and has thus received multiple sequels (Square Enix, 2013).

Acknowledging the power of virtual worlds and combining them together are significant steps in virtual world's development. As stated by Rehm, Goel and Crespi (2015), finding common tools and approaches for linking different services together is vital for the development of virtual worlds. Often, the business aspects play a major role in technology adoption, making connections between games an interesting research direction for the industry.

Creating new game brands and introducing games to new audiences is hard and requires a significant marketing budget. However, leveraging an already known brand through cross-promotion, and connecting games together, could provide a valuable boost to the marketing efforts. Game developers are currently exploring new opportunities to connect games together to attain business benefits. A good example of such efforts is the cross-promotion of Supercell's *Clash of Clans* (http://supercell.com/en/games/clashofclans/) and GungHo's *Puzzle & Dragons* (http://www.gunghoonline.com/games/puzzle-dragons/) (Grubb, 2013), which, among other benefits, enabled Supercell to expand its market reach to the Japanese markets (Handrahan, 2013).

Connecting two games together has so far been rather limited to a few examples of characters with cameo roles in other games and giving in-game currency to players who watch advertisements for other games. We continued the train of thought a little bit further by asking ourselves the following questions: could it be possible for characters to move from one game to another? How would it be possible for players' decisions to have an effect on another game? How could items be moved across games? Could the exchange of information be automatized or at least standardized in some way?

In this paper, we explore the possibility of creating a generic platform to connect multiple video games together. Currently, no generic technological solution exists to enable the transfer of video game content, such as characters, items, player-made decisions, or game events, between separate game products. We claim that in addition to cross-promotion, the connections between games could be more profound. In this respect, we explore two distinct opportunities for connected gaming: (1) linking a game to another game and (2) linking a game to reality. Linking a game to another game refers to sharing knowledge of game content (characters, items, player-made decision, and game events) between video games. Linking a game to reality refers to sharing knowledge of real-world content (items, decisions, and events) with a game. For example, tracking a person running with a sport tracker program could enhance the endurance of his/her players in a football

game for the duration of that day, or visiting Egypt could unlock a special level with the Pyramids of Giza in *Angry Birds*.

To better grasp the challenges and requirements for connecting games, we approached the problem in a user-driven manner. We interviewed Finnish video game companies to understand whether such a platform for connecting separate games together would be valuable, and what challenges might be involved in creating such a platform. In the interviews, we approached the respondents as potential users of the technological platform, and thus the analysis had focused on identifying what types of connections could exist between games, why connections between the games would be beneficial, and how games could be connected.

Based on the gathered customer needs, we created a platform, labelled Gamecloud, to support intelligent information exchange between video games. Standards for information exchange are needed (Jakobs, 2009), and even though suggestions for standards in this respect exist (Gelissen & Sivan, 2011), they are not widely used by video game developers, and do not serve all the needs for the transfer of video game content, such as characters, items, player-made decisions, and game events.

The research and development process is further presented as follows: Section 2 presents the research design and the results of the analysis of the video game developer interviews. Section 3 introduces the technology platform for connecting video games together and describes the architectural solution. Section 4 presents the testing of the technology platform with four different example games that were connected using the platform. Finally, section 5 concludes the study by drawing the results together and suggesting further research directions for connecting video games.

1. Research Design

In order to discover the requirements for the technological platform, we conducted eight semi-structured face-to-face expert interviews in 2013–2014. Both the first and the second author were present in the interviews, and the number of interviewees varied from one respondent to three respondents. Our respondents, the representatives of the game development companies, included CEOs, technology directors, and producers. We purposefully sampled the respondents by employing maximum variation criteria (Patton, 1990) to capture different perspectives. All of our respondents (representatives of game development companies) have published games, and these include games for PC, video game consoles, mobile phones, and tablets. In addition to seeking variation in the platforms where the games have been published, we sought variation in the size of the company, which the respondents represent. Thus, our sample includes both small and medium-sized businesses, as well as large companies.

Our semi-structured interview guide included the following open-ended questions, which allowed latitude for further follow-up questions:

- How are games being developed?
- What inspires you to develop games?
- Where do the ideas come from?
- What is the general process for developing plots in games?
- Are there any products or events from the real world that are presented in your games? Are your games presented in real-world products or events?
- How do cultural products such as books, comics, movies, and other games, influence game development? Have other cultural products, such as books, comics, movies, and other games, been influenced by your games?

- Have you developed games that are connected or linked?
- What factors (could) enable the development of connected or linked games?
- What will the commercial benefits of connected or linked games be?
- Please describe such a solution that will enable the development of connected or linked games.

After such questions, we prompted the respondents with ideas and suggestions regarding the architecture of the technological platform for the purpose of collecting feedback for our development work. The duration of the interviews varied from 48 minutes to 95 minutes. All interviews were recorded and transcribed verbatim.

We analysed the collected data with thematic analysis through NVivo software (http://www.qsrinternational.com/product). The purpose of the analysis was to identify the requirements (themes) for the future development of the technological platform. The thematic analysis combined aspects of deduction and induction; that is, we partly relied on an established template for the analysis and categorization of the interview data, but some themes also emerged from our analysis (Cassell, 2008). The predetermined themes included "benefits", "nature of the connection", and "technological solution", whereas the themes of "boundary conditions" and "challenges" were categories that emerged based on our analysis. In other words, the emergent categories of "boundary conditions" and "challenges" provided us with important information for the development of the platform, although we did not directly seek such information. We will next elaborate on the results of the thematic analysis that corresponded to the requirements for the platform, and will discuss what the technical solutions to the identified requirements were.

1.1 Thematic Analysis: Requirements

In the interviews, we approached the respondents as potential users of the technological platform, and thus the analysis focused on identifying what types of connections could exist between games, why connections between games would be beneficial, and how games could be connected.

First, we classified the instances of connecting games into four distinct categories: advertising, cross-promotion, linking games with each other, and linking games with reality. Advertising refers to product placement in a game. For example, a game could be promoted in another game by a character reading a newspaper where an advertisement for that game is placed. In cross-promotion, a character from one game can appear in another game. In addition to featuring characters from the previous games of that specific game developer, games can also introduce characters from games developed by other game developers. To give an example, Duke Nukem appears in *Death Rally* by Remedy, and characters from Supercell's *Clash of Clans* and GungHo's *Puzzle & Dragons* have appeared across the games. In addition, such an instance would be, for example, the remake of *Death Rally* for iOS, where the Mighty Eagle from *Angry Birds* by Rovio makes an appearance.

Advertising and cross-promotion are already utilized means for game developers to increase customer retention and to expand in the market. Other means to connect games to achieve such benefits would be linking games with each other and linking games with reality. To describe what these concepts mean, we start by differentiating these instances from advertising and cross-promotion. Whereas advertising and cross-promotion refer to static appearances of characters and graphics in general, linking games with each other and linking games with reality mean that, for example, the player could play with a character from another game. Whereas advertising and cross-promotion are somewhat commonplace, game-to-game linkages or reality-to-game linkages are rare. Regarding game-to-game linkages, *Eve Online – Dust 514* is one of the few examples where the

games share a universe and where the actions of the player in one game affect the political and economic landscape in the other game[1].

In terms of linking games with other games and linking games with reality, we identified five different themes in the analysis of the interview data: 1) the benefits of the technological solution, 2) proposed ideas on how the games could be connected (issues related to game mechanism and game design), 3) proposed ideas on how the technological solution would operate, 4) boundary conditions for the technology, and 5) challenges associated with the development and utilization of the technology. These identified themes and the questions related to them are shown in Table 1.

The business benefits identified by our respondents are a lot like those provided by advertising and cross-promotion, and include customer retention, market expansion, brand extension, and user acquisition. Such benefits could be realized, for example, by sharing achievements between games or sharing statistics. Sharing achievements means that an achievement in one game, for example, could be transported into another game, where the player benefits from the achievement. When considering reality-to-game linkages, an achievement in real-world sports could also benefit the player in the game. An example we suggested to the respondents in this regard was linking a running achievement with a sports game; for instance, a player could go for a run, record the distance of the run with Nike+, and that real-world achievement could then make an ice hockey player faster in an NHL game.

Sharing statistics was mentioned as an alternative way to connect games. For example, statistics from real-world ice hockey games could influence the skills that a character has in an NHL video game. Regarding supporting linkages between games or linkages between real-world activities and games - the technological solution was envisioned as one that would provide support for multiple platforms by the respondents, similar to how the Unity game engine (https://unity3d.com/) works.

The respondents also extensively discussed the boundary conditions for the technological solution and described the challenges that were associated with their development and utilization. By default, the respondents envisioned that, in technological terms, the solution would not be difficult. Instead, they saw that the boundary conditions and challenges arise from the game design, game mechanics, and gaming experience. In terms of boundary conditions, recognizability was highlighted. This means that the player of the connected games should be able to understand how to play the game with a specific character from another game.

Furthermore, the visual style of a character should be maintained in order for the player to recognize the game and the character. In addition, the utilization of real-world achievements in a game should not be a requirement for the player. This means that the real-world achievement should be a bonus for the player. For example, running a certain distance with Nike+ should not be required from the player to be able to play the game or to proceed in the game. Also, when creating games that are connected, inter-organizational issues such as coordination of the game development between developers and timing of the game launch, should be taken into consideration.

[1] The interaction between *Eve Online* and *Dust 514* is briefly described in the developer FAQ at http://dust514.com/game/faq/.

Table 1. Results of the analysis of the interview data

	Game-to-game or reality-to-game linkages
Description	A game that is linked to another game, sharing knowledge of game content (characters, items, player-made decisions, and game events) between games A game that is related to reality, sharing knowledge of real-world content (items, decisions, and events) with a game
Benefits	Brand extension Customer retention Market expansion User acquisition
Nature of the connection	Sharing achievements Sharing statistics
Technological solution	Support for multiple platforms
Boundary conditions	Coordination No requirements for the player Recognizability
Challenges	Asymmetry in development partnerships Fragmentation of technological platforms Loss of control over brand Loss of performance Intellectual property (IP) rights and game publishers Security

These aforementioned boundary conditions could be considered as the frame within which the development of game-to-game or reality-to-game linkages could be held. However, challenges might remain. On inter-organizational collaboration between game developers, asymmetry challenges were mentioned as an issue that might create difficulties in developing game-to-game linkages. Asymmetry arises firstly from the different sizes of the game developers (e.g. when small indie developers are collaborating with big game development companies) but also from the asymmetric distribution of received business benefits. More generally, our respondents also expressed concerns over fragmentation of technological platforms, loss of control over the brand, loss of performance, game publishers' rights over IP, and security. As games are developed for many different kinds of technological platforms (mobile, tablet, PC, and console), ensuring interoperability for games that are developed for different technological platforms was mentioned as a challenge. Unresolvable interoperability issues could further cause loss of performance in the connected games. Especially for larger game development companies, games and game characters are brands that companies want to retain control over. As many businesses utilize publishers in bringing their games to the market, some companies may not even hold IP rights in their possession, making it extremely difficult to allow game characters to be utilized for other games, even though the developers might wish to do so. Reality-to-game linkages would require game development companies to collect personal data on the players, and that data should be stored and utilized in a secure manner, raising security as a boundary condition for the technological development as well.

1.2 From Technical Requirements to the Solution

The results of the thematic analysis of the interview data reveal that the respondents envisioned the technological solution to offer benefits such as brand extension, customer retention, market expansion, and user acquisition. The connections between games or between games and reality could

be built upon sharing achievements and/or sharing statistics. As boundary conditions, the proposed technological solution should enable coordination, should not create any additional requirements for the player, and should ensure recognizability of the games connected. The respondents also brought up the challenges that they saw being associated with connecting games or games and reality, such as asymmetry in development partnerships, fragmentation of technological platforms, loss of control over the brand, loss of performance, IP rights and game publishers, and security.

The interviewed developers and company representatives stated that data privacy and security were seen as important parts when exchanging information between different games. Thus, the system must provide good security and not share unnecessary information of the players. Also, the developers must be given the power to decide what information of their games is shared and with whom it is shared. As already mentioned, the problem of asymmetry between well-known products and more-unfamiliar products needs to be taken into account.

However, as the developers were discussing interoperability, support for multiple platforms and support for development that is independent of other developers were mentioned as necessary requirements. Based on our previous research (Parkkila, Hynninen, Ikonen, Porras, & Radulovic, 2015; Parkkila, Ikonen, & Porras, 2015), ontologies were selected as a tool for enabling interoperability among separate video games. The term 'ontology' refers to the common conceptualization of information (Shadbolt, Berners-Lee, & Hall, 2006) that can be understood by machines. Ontology is a solution to ensure the interoperability of different software applications and, in addition, enables reasoning based on the modelled data. In the interviews, we touched upon the technical concept of ontologies, but the respondents were not familiar with it. Thus, using ontologies as is would not be a viable approach, as it would require developers to learn entirely new concepts and tools to use them. Instead, the system must be able to abstract ontologies from developers, provide a simple interface for developers to model game information, and provide simple interfaces for games to communicate with other games.

In addition, we interpreted the following technical requirements from the interviews: The system must be 'always on', meaning that it has to be available and accessible from anywhere. Thus, we chose to utilize cloud technology. The technological solution must allow game developers to choose what they want to share in order to enable them to monitor and define their contents. Also, the technological solution must preserve the actual logic of the game, meaning that connecting games through Gamecloud must not break the logic of the games to be connected.

Based on the analysis of the interview data, we translated the aforementioned themes into technical requirements for the technological solution. Following the principles of system information design, we developed the first version of the technological solution based on these technical requirements. Moving further, we developed games and tested the technological solution with the developed games. Thus, our design procedure was iterative in its nature and had multiple versions. The platform presented here only portrays the final version of the process, not the actual design process itself.

2. The Technology Platform and Its Architecture

In this section, we present the Gamecloud platform architecture, based on the requirements and features obtained from the developer interviews, as presented in the previous section.

The goal of this research was to design and implement a technology platform that enables different service providers related to the field of video games to connect and to exchange information, independent of the used technologies. Even though there are some suggested standards (Han, Han, Kim, & Kim, 2013), they are not without criticism (Garcia & LeMasters, 2009). Also, at the time of writing, we did not find any ontology for the transfer of video game content, such as

characters, items, player-made decisions, and game events. Thus, a separate Video Game Ontology (http://purl.org/net/VideoGameOntology) was created and used as a tool for enabling interoperability between games. The platform must be accessible everywhere; thus, the decision was made to use cloud technology (hence the name – Gamecloud).

As already mentioned in the previous section, the system can have potential impacts on different stakeholders: players, game companies, and third parties, such as real-world retail stores or smart digital services. The platform is not only for connecting games together but also for enabling external services to use the stored information and to provide non-game-related digital information. An overall architecture of how different stakeholders are related to the Gamecloud platform is shown in Figure 1.

This section explains the technology behind the developed system, explaining the developed architecture, describing the user interface for game developers, and briefly discussing the development of the system's security.

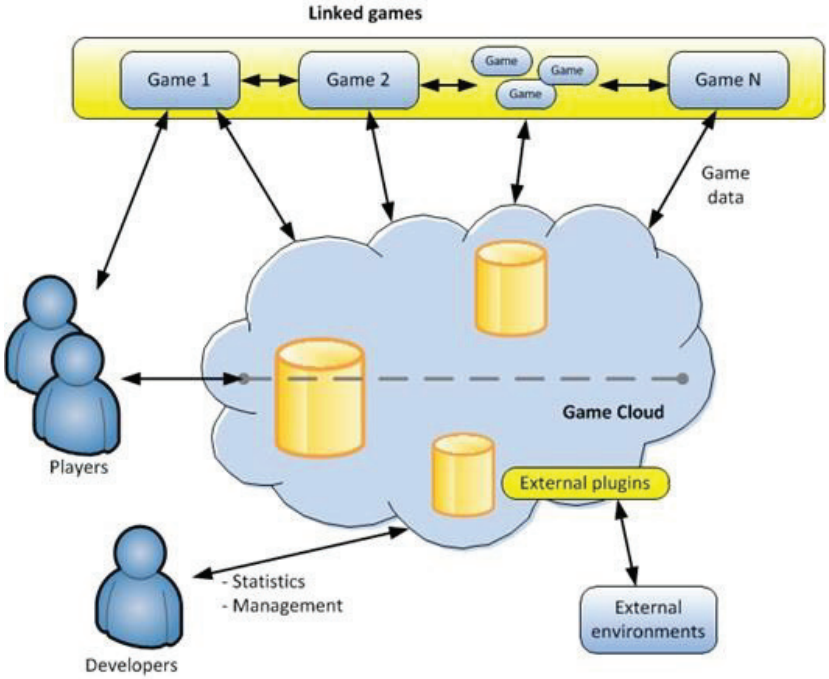

Figure 1. Overview of Gamecloud

2.1 Platform Architecture

The architecture of the Gamecloud platform was split into three different parts: 1) REST API (representational state transfer application programming interface), which is used to communicate between the various games and Gamecloud; 2) Server middleware, which handles authentication and required data validations, transformations, and query building; and 3) The Ontology Engine, which stores the modelled ontology data and performs reasoning based on the given queries. The Gamecloud architecture stack and its main functions are shown in Figure 2.

The server architecture was created to support all the required use cases and to abstract ontology from the developers. Based on the interviews with the game developers, we understood that they were not familiar with the concept of ontologies and decided to create the system in a manner that does not require knowledge of ontologies. Thus, we created middleware for the server that hides

the actual ontology implementation from the developers and enables the use of simpler requests for the developers, lowering the amount of time required to integrate a video game into the Gamecloud platform and also lowering the technological knowledge requirements for using the platform.

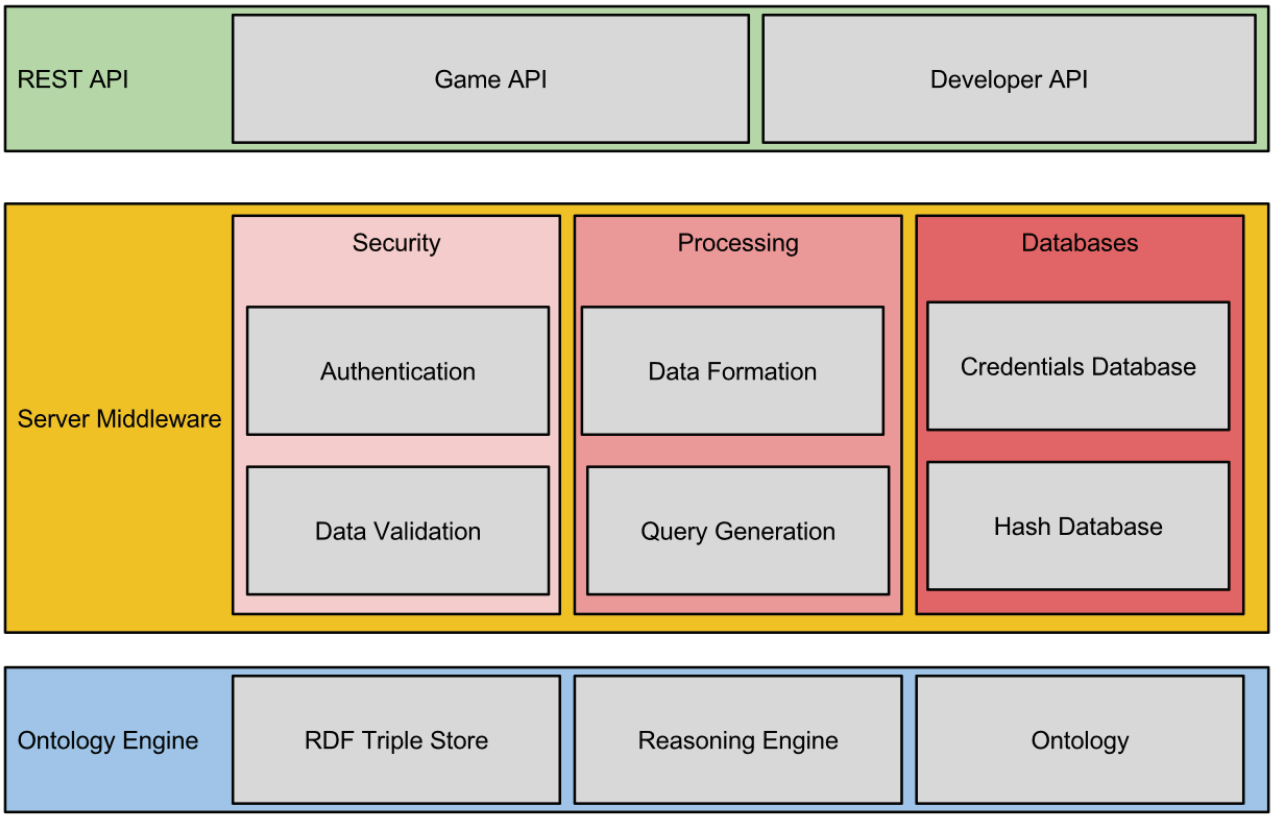

Figure 2. Overview of the Gamecloud server stack

2.1.1 REST API

The communication between server and client is done through a common REST API. The API allows stateless connectivity, meaning that the server will always respond to requests in the same way, independent of previous communication. This makes the server functionality more predictable and requires fewer computational resources, as there is no need to keep the state of each client transaction in memory. As one of the requirements was support for multiple platforms, we decided to use REST APIs for communication, as the only requirement for client–server communication is HTTP capability (i.e. Internet access) from the client. Usually, a device that can connect to the Internet can thus interact with the Gamecloud platform. In addition, all the data is received using JSON (JavaScript object notation), which is a common, lightweight data-interchange format (http://www.w3schools.com/json/).

The API layer itself is divided into two different APIs: first for the developers and second for the video games. The APIs are separated from each other for two reasons: first, to limit access to certain functions of the platform, and second, to simplify the API usage for the developers.

The developer API is meant for inputting developer-related information into the system, such as modelling the ontology information of games and to ask for game-related statistics. The developer API was created to decouple the developer UI (user interface) from the actual Gamecloud, making the UI open for development and enabling even third parties to create interfaces for the system. A more specific explanation of how the developer UI is used is described in a later part of this section.

The game API is used by the video games to store players' gameplay information, for example to store knowledge of newly gained items or achievements to Gamecloud. The game API accepts short string hashes with player-related information, which are then later converted to the actual SPARQL (ontology) queries at the server middleware. This process is explained more specifically in the section 'Server Middleware'.

To demonstrate the extension of the developer API, we created supporting tools that take advantage of the REST API. Based on the interview results and the fact that the Unity engine is quite popular among game developers, we created a Gamecloud helper tool inside the Unity editor. The Gamecloud helper tool takes advantage of the developer API implementation. The created Unity plugin enables the definition and creation of new ontology entries to Gamecloud inside the Unity editor, without the need for the developer to use another development tool. This allows developers to use already familiar tools to define content in the platform, lowering the barrier for adoption. In addition, we created a helper JavaScript library for Gamecloud, which also reduces the development time for JavaScript-based games by providing helper functions to integrate games with the Gamecloud platform.

2.1.2 Server Middleware

The server middleware is the part that handles the majority of data processing, such as user authentication, data validation, and processing and transforming RDF (resource description framework) data into simpler queries for the developer and game APIs to use. Authentication and data validation are common parts of any service platform. However, the data formation and query generation parts are the most interesting functions of the server middleware.

Based on the respondent interviews, the developers were not familiar with ontologies, and the design decision was to abstract the ontologies from the developers to lower the adoption barrier of the service. This was undertaken with a query generator that converts SPARQL (SPARQL protocol and RDF query language) ontology queries into hashes that are then stored in a hash database. Each hash corresponds to a certain SPARQL query that can be used to either retrieve or store information in the ontology engine. The developer is only shown the short hash string, which is converted into the actual SPARQL query when a client query with the matching hash is received. This way, the developer is not required to learn a new language for storing or retrieving information in/from the platform, yet the platform provides a solution for taking advantage of the ontologies.

The process of abstracting the ontology from the developer is portrayed in an example scenario in Figure 3. For example, adding a magical sword via the developer API generates an ontology definition of the item, for example "the magical sword is a one-handed sword that can be equipped by a character; it has a magical property and it belongs to the example game." This ontology definition is then saved to the RDF store at the ontology engine. In addition, the query builder generates SPARQL query templates for giving the magical sword item to the player in the specific game, for taking it away from the player (e.g. the player selling the weapon in the game), and for asking if a player has the item in question. All of these query templates are then stored in a database with a matching hash for the query. The hashes are then returned to the developer, who can use the hashes to perform queries in the Gamecloud, instead of writing long SPARQL commands.

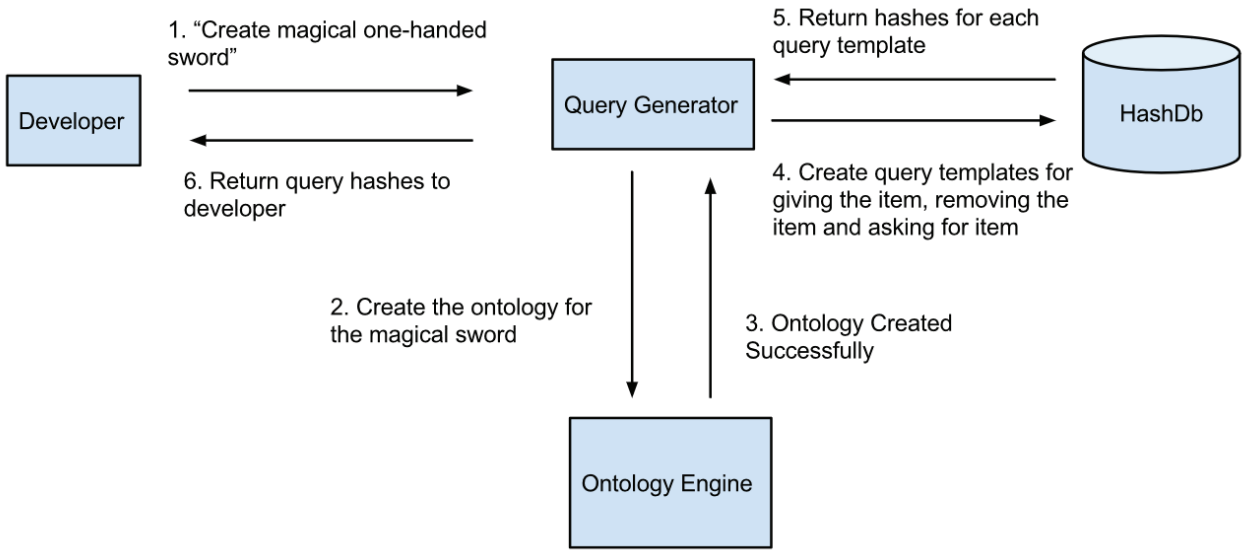

Figure 3. Example of query generator workings of server middleware

2.1.3 Ontology Engine

The ontology engine is the core of the Gamecloud platform. The ontology engine handles the storage, retrieval, and reasoning of the modelled video game data in the system. It consists of three parts: the actual Video Game Ontology (http://purl.org/net/VideoGameOntology), a triple store (for storing the data), and a reasoning engine, which handles all the ontological reasoning. The Video Game Ontology contains the model for all game-related events, such as the player gaining an item or an achievement, the player making an in-app purchase in a game, and the player's gameplay session information.

All the game information entered in the Gamecloud is modelled according to the Video Game Ontology. Each ontology entry (i.e. the implementation of an ontology 'class' definition) is then saved in the triple store. A triple store is a database that consists of subject-verb-object graphs, such as "Michael is a person". These triples can be used to explain concepts and their relations. An example of defining an achievement of gaining 1,000 coins, which is a collection type of achievement, is modelled as shown in Figure 4.

The reasoning engine is in charge of handling all the queries in the triple store. In addition to working as an interchange language between different games, the ontology enables reasoning to answer complex questions. For example, the system can answer such questions as "how many players who have played video games of the roleplaying genre during the last 24 hours have made an in-app purchase in this game?" or "how many health potions have been consumed by players who have completed level 3 and have not purchased any magical weapons?" All these questions are possible, as long as the equivalent events have been modelled to the system (e.g. an event for completing level 3 and an event for making an in-app purchase.) This possibility for making powerful queries enables in-depth analytics, allowing game developers to understand their players even better.

However, as ontologies are not very well known among developers and they require some experience to be used, we decided to abstract the ontology layer to an easier interface. The next section describes how the developer experience works, making the modelling of ontologies to Gamecloud a much easier task that requires a lot less-specific technical knowledge.

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix vgo: <http://purl.org/net/videogameontology#> .

<http://www.example.com/Collect1000Coins>
  a vgo:Collection;
  rdfs:comment “An achievement gained from collecting 1,000 coins in a game”.
}
```

Figure 4. Example of how the ‘collecting 1,000 coins achievement’ looks when modelled according to the Video Game Ontology.

2.2 Developer UI for Modelling Data

To enable video game developers to use the system easily, a UI was developed to allow the developers to model information about their games to the Gamecloud platform (Paraschou, Ikonen, Heikkinen, & Parkkila, 2014). The UI works completely in the web browser, using the Gamecloud REST API to communicate between the UI and Gamecloud. Even though we have created one type of developer UI for the system, the Gamecloud enables anyone to develop a different kind of UI to suit their needs (or to implement the developer functionality into any developer tool, as noted later in this section on Unity implementation.)

The developer UI displays all games created by the developer and shows all the items, events, and achievements modelled to the system. The UI allows the developer to model new knowledge about their games via a step-by-step wizard that guides the developer to enter game data properly into the system. After successfully entering the game data into the system, the developer is shown an instructions page, which explains how to implement the commands related to the added entry into the game program code.

In order to help with the integration process of Gamecloud, we created a set of REST API tools, which can be used to communicate with Gamecloud. There are currently Gamecloud API libraries for JavaScript, Python, and the Unity engine. The API libraries require the programmer only to call an equivalent function, passing the hash and other required arguments, making the integration process even easier. An example of giving an achievement to a particular player with the use of the JavaScript Gamecloud library is shown in Figure 5. The process of modelling information to Gamecloud and using it in the developed game is shown in Figure 6.

```
var Gamecloud = require(‘./gamecloud’);

Gamecloud.giveAchievement(<developerKey>, <hash>, <playerId>);
```

Figure 5 – An example of using the Gamecloud API library in JavaScript

```
1. Design the game (as normal)
2. Define the game content to Gamecloud via developer UI
        * Gamecloud will automatically generate a function to be sent
        to Gamecloud to track the wanted entry status
3. Implement provided function in the game program code
```

Figure 6 – Example of the developer ontology modelling process in Gamecloud

In addition to the web UI, we created a Gamecloud plugin for the Unity game engine. Unity is a widespread development tool for games and was used by the interviewed developers. For integrating Unity games into Gamecloud, we created a separate plugin (https://github.com/lut-projects/GamecloudAPI) that enables the modelling of game data, as well as adding Gamecloud queries to the game, inside the Unity editor, without the need to use other external tools.

2.3 Security

One of the most important factors that arose during the interviews was information security and how to maintain it, especially, keeping the trust of players and preserving their privacy. The information exchange between games and the real world cannot reveal any unnecessary information about the players. In addition, the players need to be aware of what information is collected from him/her and who is processing this information. For the game, it is important that the external signals cannot interfere with the game or the balance of the game world, as for the developer it is important that the business logic cannot be altered in a harmful manner. Also, for the sake of Gamecloud, it is important that all transferred data be coherent (integrity) and reliable so that both players and developers can trust the services offered by the platform.

In order to maintain and enable good privacy, the platform was designed following the privacy-by-design approach (Cavoukian, 2009). Privacy by design was taken as part of the development process straight from the design phase, and special care was given to privacy. The approach enabled both security and privacy to be taken into account throughout the whole process and also streamlined the development process (Laakkonen, Parkkila, Jäppinen, & Ikonen, 2014). An example of information protection can be seen in Figure 7, which shows the access rights of different actors to the different parts of the service.

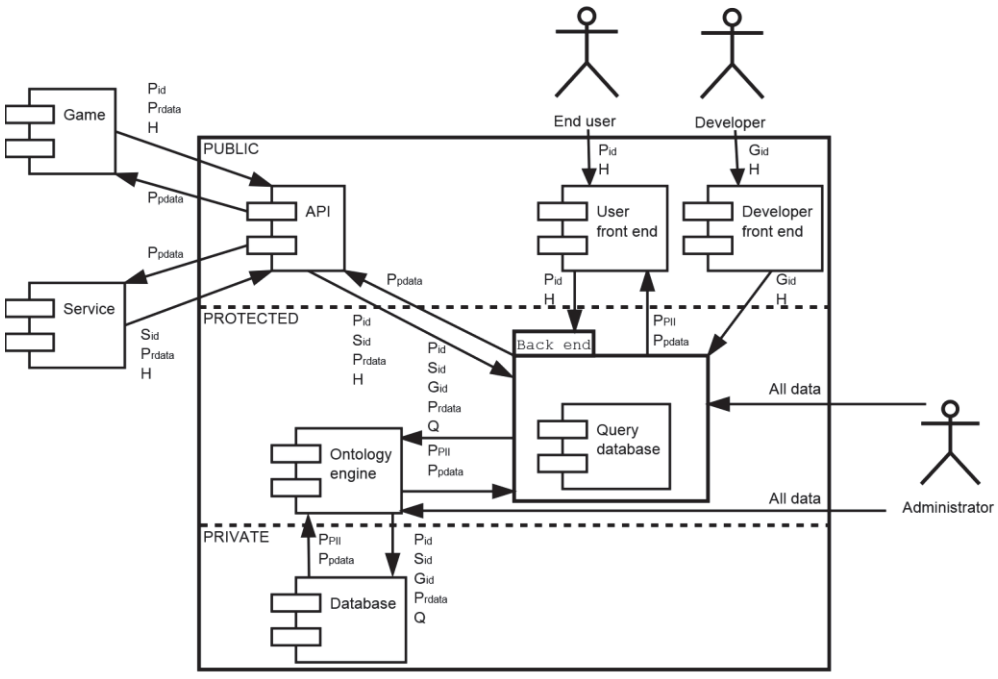

Figure 7. Overview of Gamecloud security model (Laakkonen et al., 2014)

3. Testing

To evaluate the implemented architecture and features of the Gamecloud platform, we decided to test the platform for use with different games. The starting requirement for testing was to find a

collection of games that could be connected to communicate with each other through the Gamecloud platform. The goal of testing was to connect games together, in order to demonstrate and evaluate the links between multiple games. The games had to be able to communicate with each other so that the interaction with the platform could be evaluated.

We contacted several game companies (the ones interviewed and five others) to test their existing games with the Gamecloud platform. However, getting commercial game companies to test their existing products with the Gamecloud platform turned out to be a challenging task. Instead, we decided to use existing games and to develop them to suit our needs. We took three open-source games (*Asteroids*, *Coil*, and *BrowserQuest*), which were modified to work with Gamecloud. In addition, we created a fourth game (*BugBear*) to test the possibility of merging physical activities with video games. The games use Gamecloud in storing and linking game data to test and demonstrate how Gamecloud works, as well as to gather player information and experiences on such connected games[2].

The games were tested from a simple usage of modelling only the core concepts (Example 1 – *Asteroids*) with Gamecloud to a more complex one (Example 3 – *BrowserQuest*). An overall view of all tests undertaken, their results, and the lessons learned is shown in Table 2. The rest of this section explains the tests made with each game and their results in more detail.

Table 2. An overview of Gamecloud platform tests done with different video games

Game	Purpose of test	Results of test	Lessons learned
Test 1 – *Asteroids*	Validating service connections, logging, and integration process	The game worked with Gamecloud and required only a few additional lines of code and preparation to integrate the system	System works, yet a simple arcade game does not demonstrate the full potential of Gamecloud
Test 2 – *Coil*	Testing cross-platform (tablet–PC) interactions and gathering gameplay interaction data	Gamecloud can gather even fine-grained and detailed gameplay data (every finger tap and player action)	For analytics purposes, collecting everything might not be important, but the Gamecloud system enables modelling of even the finest of details
Test 3 – *BrowserQuest*	Testing creation of new game content based on actions in other games and working on linking games together	Gamecloud can keep the player inventory properly up to date. Also, linking games with the player's previous gameplay history can be implemented with Gamecloud	Showing/giving game content to players according to their previous games works, and even a multiplayer game scenario does not provide problems. However, providing extra content still requires developer effort to implement the content in program code and graphical asset forms
Test 4 – *BugBear*	Evaluating the possibility of rewarding players based on physical activities (e.g. exercising)	Combining games with physical exercising is feasible and can be achieved with Gamecloud	Designing games that interact with non-game content is possible yet requires some design thinking to achieve wanted results

[2] The three first Gamecloud platform test games can be found at http://www.gamecloudgames.com/

3.1 Game 1 – *Asteroids*

The first test game was created by editing an open-source game called *Asteroids (*http://www.gamecloudgames.com/HTML5-Asteroids/). The game is a browser-based arcade game with the goal of destroying and dodging asteroids floating around the player. The player is granted points by destroying asteroids. When an asteroid is hit by a bullet, it divides into smaller asteroids that continue floating in the game field. If an asteroid hits the player's ship, the player loses one life. The game is over when the player has lost three lives.

The goal of the first test was to validate that the Gamecloud APIs (both developer and game APIs) work properly with games. In addition, we developed a JavaScript helper library to support usage of the Gamecloud platform with HTML5-based games. We modelled basic game events, such as starting a new game, destroying an asteroid, and game over, to the Gamecloud with the developer UI. We then tested implementing the sending of each event with its equivalent hash information to Gamecloud. Implementing the communication with the Gamecloud platform with the helper JavaScript library only required the addition of one line of program code to functions that are related to the event in question. Furthermore, displaying the Gamecloud player login component on the game's webpage had to be added to the game's main screen. In total, adding Gamecloud support to the existing *Asteroids* game required fewer than ten lines of program code.

The first test validated that Gamecloud can be integrated into a simple browser-based arcade game with a small developer effort. Also, we were able to model game events to Gamecloud and to exchange the modelled gameplay information with the Gamecloud platform. However, more testing was required to evaluate the platform in depth.

3.2 Game 2 – *Coil*

The second test game, *Coil* (http://www.gamecloudgames.com/Coil/), is a browser-based game with the goal of encircling blue balls by moving the user's finger on a touchscreen or with a mouse. Each ball is only visible for a short moment before it turns into a yellow ball and eventually explodes, taking some of the player's energy with it. In addition, red balls appear on the playing field, and encircling those causes the player to lose energy. As the game continues, more and more balls appear on the playing field, until the player can no longer encircle blue ones fast enough and he/she loses all the energy, leading to game over.

The goal of the second test was to evaluate Gamecloud's performance when dealing with large amounts of data. As one of the main offerings of the platform for game developers was the possibility of analysing players' gameplay data, much information can be gathered during a single playing session. For this test, we modelled every event of new balls appearing on the playing field (e.g. a red ball appears, a blue ball appears, a blue ball is encircled, and a red ball is encircled) and player actions (e.g. the player starts drawing a circle and the player ends drawing a circle). This way, we were able to create hundreds of gameplay events even from a short playing session, which later could be analysed by the developers for improving the gameplay experience.

The second test result showed that modelling even the most-specific player actions is possible with the Gamecloud platform. We were able to retrieve anonymised session information, in order to see what happened during each session and how players acted. Even though the Gamecloud platform enables the retrieval of anonymised gameplay session data, the data itself currently needs to be analysed using different third-party tools. For future development purposes, the Gamecloud platform could offer simple analytics tools, to enable basic analysis of the available data.

3.3 Game 3 – BrowserQuest

The third test game, *BrowserQuest* (http://www.gamecloudgames.com/BrowserQuest), is a multiplayer adventure game originally developed by the Mozilla Foundation (https://www.mozilla.org/en-US/foundation/) to demonstrate the possibilities of HTML5 technology. The goal of the game is to adventure around the game world, exploring dungeons, fighting enemies, and collecting better weapons and armour for the main character. The game allows multiple players to adventure in the same world simultaneously.

The goal was to test the implementation of a game more complex than the two previously used arcade games. Gamecloud supports the modelling of items, such as weapons, armour, and consumable potions, and we wanted to test how modelling an inventory of items works on the platform. In addition, we wanted to link games together, giving a small bonus to players who have already played another arcade game. For testing purposes, we edited the open-source game to exchange information with the Gamecloud platform, gathering knowledge of player achievements, owned items, and killed enemies. We implemented blue armour (the 'Armour of Arcade Awesomeness') that is only available to players who have gained at least five achievements in any arcade-type game (i.e. in our first test games: *Asteroids* and *Coil*). In addition, *BrowserQuest* includes an agent character in the starting town of the game. We changed the character replies to contain an advertisement link to the *Asteroids* game. If the player clicks the link, he/she is awarded with an extra life in *Asteroids*, which is then checked by the *Asteroids* game every time a player starts a new game.

As a result of the third test, we were able to track player inventory in Gamecloud. The full range of potions consumed and owned was kept properly in the cloud, and the amounts were properly tracked. Also, the implementation of giving a special item to players, who had gained five achievements in the arcade games, worked properly. Granting special content to players based on their previous experiences proved to be possible with the use of ontologies. However, the game developer (in this test case, the researchers) had to create the implementation of the armour and its graphics in the game program code and assets.

Transferring items with all their graphics and attributes could be possible with Gamecloud but with certain limitations. The attributes could be carried over if they were defined in the system (e.g. a sword has the special feature of dealing ice damage), and if the receiving game has some method for implementing and displaying these attributes (e.g. ice damage) in the game logic. Also, item graphics could be transferred from one game to another if the games use the same game engine (e.g. Unity) and would be stored in Gamecloud as assets of the engine in question. However, this is currently purely hypothetical and would require entirely new research to evaluate and test the implementation.

3.4 Game 4 – *BugBear*

The fourth and last test was about analysing how non-game information could be combined with games. For our tests, we implemented a simple Pokémon (http://www.pokemon.com/) style game called *BugBear* and a simple exercise-monitoring application, which allows users to save their physical exercises and see their exercising history (Ikonen, Ryhänen, Parkkila, & Knutas, 2015). The goal of the *BugBear* game is to train a BugBear to fight against other players, in Pokémon-style gameplay. The game also monitors how many other players are around the player, giving bonuses according to the amount of nearby people.

The exercise-monitoring application and *BugBear* are both connected to the Gamecloud platform, storing and reading player information from the platform. *BugBear* allows the player to play only a certain number of fights per day, as the BugBear runs out of stamina and needs time to recover. The exercise-monitoring application stores knowledge of the player's physical exercising,

which is then used in *BugBear* to replenish the stamina of the BugBear, allowing the player to keep playing longer.

The testing of connecting physical activity tracking with a video game was successfully implemented with the use of the Gamecloud platform. Modelling physical activities was possible with Gamecloud, and adding deeper connections with physical exercising could be done by extending the platform with a more-specific physical activity ontology. The implemented example suggests that non-game-related knowledge can be merged with games, using the Gamecloud platform.

4. Conclusions

The findings of this research are manifold. The technology platform was tested with four example games, verifying that connecting games together is possible with only a small effort by game developers.

However, testing the platform with commercial games turned out to be difficult to arrange. Game companies were not interested in testing out the platform with their existing products. This is not very surprising, as testing new technology with published products is always risky. Nonetheless, we were able to develop four example games that allowed us to test the platform in practice.

In our interviews, the game companies saw linking games with each other and with the physical world as a promising venture. However, collaboration with other developers and keeping games playable without prerequisites for the players were seen as challenges. Many game companies felt that releasing data of their games to other firms posed a risk to their own brands. This is a problem of social capital and trust, which requires further research to evaluate the possibilities on how to create trust among the actors in the field.

We suppose that links between games, co-branding, and co-marketing will quite likely become more frequent shortly. Gamecloud is a research platform, yet large digital game store providers such as Valve and Google already possess platforms for enabling the linking of games among the games in their stores. In addition, large game companies such as Microsoft, Sony, and Ubisoft have resources to link their game products together, which has already happened on a smaller, yet rather limited, scale.

In the first phase of linking games together, it is quite likely that this transition will start among the products of separate companies. This helps to avoid difficult competition situations and the marketing of competing products in own games. However, the linking will not only be among games; real world and game worlds will start to converge with the advances of product digitization, bringing new business opportunities to all parties.

For future research, co-operation between the physical world and game worlds in particular is a field with many possible research directions. The cross-section of the digital and physical worlds has huge potential for both new business ventures and new interaction paradigms. The interviews undertaken during the design phase of the Gamecloud platform raised many discussions and interest towards the real-world links, yet a lot more research is needed to find business opportunities that benefit all parties.

References

Cassell, C. (2008). Template analysis. In *The SAGE Dictionary of Qualitative Management Research* (pp. 221–223).

Cavoukian, A. (2009). Privacy by design: The 7 foundational principles. *Information and Privacy Commissioner of Ontario*.

Garcia, D. L., & LeMasters, G. (2009). Synthetic Excellence: Standards, Play, and Unintended Outcomes. *Journal For Virtual Worlds Research*, *2*(3).

Gelissen, J. H. A., & Sivan, Y. Y. (2011). The Metaverse1 Case: Historical Review of Making One Virtual Worlds Standard (MPEG-V). *Journal For Virtual Worlds Research*. http://doi.org/10.4101/jvwr.v4i3.6066.

Grubb, J. (2013). Clash of Clans, Puzzle & Dragons working together again to further dominate global gaming. Retrieved from http://venturebeat.com/2013/10/21/clash-of-clans-and-puzzle-dragons-developers-collaborate-for-a-new-cross-game-event/.

Han, S., Han, J.-J., Kim, J. D. K., & Kim, C. (2013). Connecting users to virtual worlds within MPEG-V standardization. *Signal Processing: Image Communication*, *28*(2), 97–113.

Handrahan, M. (2013). GungHo and Supercell in cross-promotion deal. Retrieved from http://www.gamesindustry.biz/articles/2013-06-10-gungho-and-supercell-in-cross-promotion-deal.

Ikonen, J., Ryhänen, P., Parkkila, J., & Knutas, A. (2015). Linking physical activities and video games. In *Proceedings of the 16th International Conference on Computer Systems and Technologies* (pp. 120–127). ACM. http://doi.org/10.1145/2812428.2812457.

Jakobs, K. (2009). Real Standards for Virtual Worlds – Why and How? *Journal For Virtual Worlds Research*, *2*(3).

Laakkonen, J., Parkkila, J., Jäppinen, P., & Ikonen, J. (2014). Continuous Development of Gamecloud with Privacy by Design. *International Journal on Information Technologies & Security*, *6*(4), 51–64.

Paraschou, N., Ikonen, J., Heikkinen, K., & Parkkila, J. (2014). Designing a user interface for game developers to enter game specific information. In *Proceedings of the 15th International Conference on Computer Systems and Technologies* (pp. 309–316). ACM. http://doi.org/10.1145/2659532.2659617.

Parkkila, J., Hynninen, T., Ikonen, J., Porras, J., & Radulovic, F. (2015). Towards Interoperability in Video Games. In *Proceedings of the 11th Biannual Conference on Italian SIGCHI Chapter* (pp. 26–29). ACM. http://doi.org/10.1145/2808435.2808454.

Parkkila, J., Ikonen, J., & Porras, J. (2015). Where is the research on connecting game worlds?—A systematic mapping study. *Computer Science Review*, *18*, 46–58. http://doi.org/10.1016/j.cosrev.2015.10.002.

Patton, M. (1990). *Qualitative evaluation and research methods*. SAGE Publications, inc. Thousand Oaks, CA, US.

Prata, W., de Moraes, A., & Quaresma, M. (2012). User's demography and expectation regarding search, purchase and evaluation in mobile application store. *Work*, *41*(Supplement 1), 1124–1131. http://doi.org/10.3233/WOR-2012-0292-1124.

Shadbolt, N., Berners-Lee, T., & Hall, W. (2006). The Semantic Web Revisited. *IEEE Intelligent Systems*, *21*(3), 96–101. http://doi.org/10.1109/MIS.2006.62.

Sotamaa, O., & Karppi, T. (2010). *Games as Services - Final Report*. University of Tampere. Tampere, Finland.

Square-Enix. (2013). KINGDOM HEARTS HD 2.5 ReMIX in Development. Retrieved from http://na.square-enix.com/us/blog/kingdom-hearts-hd-25-remix-development.

Tang, T., Newton, G. D., & Wang, X. (2007). Does synergy work? An examination of cross-promotion effects. *The International Journal on Media Management*, *9*(4), 127–134.

Washburn, J. H., Till, B. D., & Priluck, R. (2000). Co-branding: brand equity and trial effects. *Journal of Consumer Marketing*, *17*(7), 591–604. http://doi.org/10.1108/07363760010357796.