information

Article

Ontology-Based Domain Analysis for Model Driven Pervasive Game Development

Hong Guo [1,*], Shang Gao [2], Hallvard Trætteberg [3], Alf Inge Wang [3] and Letizia Jaccheri [3]

1 School of Business, Anhui University, Hefei 230601, China
2 School of Business, Örebro University, 70182 Örebro, Sweden; shang.gao@oru.se
3 Department of Computer and Information Science, Norwegian University of Science and Technology, 7491 Trondheim, Norway; hal@ntnu.no (H.T.); alf.inge.wang@ntnu.no (A.I.W.); letizia.jaccheri@ntnu.no (L.J.)
* Correspondence: homekuo@gmail.com; Tel.: +86-155-511-85532

Received: 28 March 2018; Accepted: 27 April 2018; Published: 3 May 2018

Abstract: Domain Analysis (DA) plays an important role in Model Driven Development (MDD) and Domain-Specific Modeling (DSM). However, most formal DA methods are heavy weight and not practical sometimes. For instance, when computer games are developed, the problem domain (game design) is decided gradually within numerous iterations. It is not practical to fit a heavy-weight DA in such an agile process. In this research, we propose a light-weight DA which can be embedded in the original game development process. The DA process is based on a game ontology which serves for both game design and domain analysis. In this paper, we introduce the ontology and demonstrate how to use it in the domain analysis process. We discuss the quality and evaluate the ontology with a user acceptance survey. The test result shows that most potential users considered the ontology useful and easy to use.

Keywords: Model Driven (Software) Development (MDD); Domain-Specific Modeling (DSM); ontology; domain analysis; computer game; pervasive game; game development

1. Introduction

The Model Driven (Software) Development (MDD) field has given birth to Domain-Specific Languages (DSL) which formally represent domain solutions using high level concepts that are close to the problem domain. DSL development usually takes four stages (decision, analysis, design, and implementation) [1]. Among them, Domain Analysis (DA) provides crucial information for the DSL construction. However, this stage has not received as much attention as consequent stages. Few DSLs (only four out of 39 as evaluated in [1]) utilized more formal DA methods.

In addition, most DA methods are heavy weight and not practical for some domains. For instance, when games are developed, game design is gradually developed within numerous iterations where prototypes are constructed and playtest is performed. In such a case, it is not practical to fit a heavy weight DA process in such an agile process. To our best knowledge, few MDD trials in game domains used formal DA methods (as will be reviewed in Section 2). Insufficient and inefficient DA may partly count for the fact that most of reported MDD applications in game domains have not been followed up with more successful evidence.

In this research, we propose a light weight DA process which can be integrated into the original game development process. The DA process is based on a game ontology which serves for both DA tasks and game design tasks. To make our work more concrete, we focus on pervasive (computer) games instead of general computer games.

The rest of the article is organized as below: Section 2 introduces some background information regarding MDD, DA, and game domains. Then, Section 3 presents the proposed ontology (PerGO) and Section 4 introduces the domain analysis procedure based on PerGO. Section 5 demonstrates the usage of PerGO based domain analysis with an example. We discuss the related work with details in Section 6. Then in Section 7, we discuss the strengths and limitations of our work. A user acceptance test is also presented. Lastly, we summarize the article and talk about some possible future work.

2. Background

In this section, we introduce the background information on MDD, DA methods for MDD, and their application in game domains. We briefly introduce how we integrate DA and other MDD tasks in the game development process. The emerging pervasive game domain is also briefly introduced.

2.1. Model Driven Development and Domain Analysis Methods

Using models has been a tradition when designing complex software systems. By using models, it is easier to understand a complex problem as well as corresponding solutions. However, models have been thought as documentation artifacts primarily. This situation was changed when MDD is applied, where models instead of programs became the primary products of software development [2].

When MDD is applied, the working focus is changed from programming to solution modeling [3]. Two mechanisms are involved here: making abstractions which are closer to the problem domain and transforming models to programs [3]. By doing so, a number of potential benefits can be gained. Firstly, models/software are easier to understand and specify [2]. Secondly, models are easier to maintain than code [2]. Thirdly, the software development productivity can be significantly improved [4] and lastly, the quality of software can be improved.

MDD should be domain-specific to play its strength [5]. This requires Domain-Specific Languages (instead of General Purpose Languages) to raise the level of abstraction, and domain-specific platforms to reduce the complexity of the code generator. That is why Domain-Specific Modeling (DSM) is regarded as an alternative terminology to MDD in many scenarios. These two terminologies are used interchangeably in this article.

Despite the benefits of MDD, the development of MDD is not easy. There are some challenges about how to create abstractions close to the problem domain, and how to decide and formalize the DSL semantics [6]. To meet these challenges, Domain Analysis (DA) plays an important role. DA is one of the four tasks (decision, analysis, design, and implementation [1]) to perform DSM. The outputs from domain analysis support the decision making up front and provide a solid base to start design and implementation afterward. Thus, the quality of domain analysis can be crucial to the quality of DSL and the success of the overall MDD solution. Usually, outputs of DA includes a domain vocabulary, commonality space, and variability space [1].

Researchers proposed formal DA methods such as FAST (Family-Oriented Abstractions, Specification, and Translation) [7] and FODA (Feature Oriented Domain Analysis) [8]. Formal domain analysis showed good language design result, but most of the formal DA methods are heavy weight and not practical in some situations. Thus, the usage of these methods is limited [9,10]. Researchers also investigated Ontology Based Domain Analysis (OBDA) [9] since ontologies are closer than its original domains and provide an effective way to reuse the domain knowledge in traditional domain engineering [11].

2.2. MDD and DA Task in Game Development

As DSM has been applied in various domains and successfully achieved expected benefits, researchers tried to apply it elsewhere, in the computer games domain for instance [12–17]. The computer games domain looks desirable for this application. Due to complicated architecture and different domain knowledge, computer games can be extremely complex. Applying DSM would help lower the overall complexity by hiding the domain (knowledge) complexity in DSLs.

However, the application of DSM in computer game domains is not well-addressed. Few articles were found where MDD application was reported in game domains. This is partly caused by the conflict of heavy-weight MDD process and agile game development traditions. To address this issue, we proposed an agile workflow to integrate MDD tasks in the traditional game development process. This workflow will be briefly introduced in Section 2.3. Further, among these works, it is even more difficult to find detailed descriptions of how their DA task was carried out. In Section 7, we will review these articles with more details.

2.3. MDD/DSM Embedded Game Development

As stated above, the game development process is highly iterative. In addition, the problem domain (game design) is constructed in a gradual way. It is not practical to fit a heavy-weight DSM process in the original process in order to benefit from applying DSM. To solve this issue, we propose a light-weight DSM workflow which can be embedded in the traditionally highly iterative game development process. In this process, game design and DSM are performed in a woven way. As shown in Figure 1, original game development products (the left part and the right part in the figure) are reused as DSM inputs (the middle part in the figure) in an efficient way and which makes the DSM (including DA) process integrated with the game development process tightly with light weight. To synchronize both game design and DSM, and make the overall process more efficient, products of game design and DSM are reused in each other. In addition, an ontology is used as a base for both game design and DSM. We introduce the ontology as well as the ontology based DA process in Sections 3 and 4, while more details about the workflow, as well as how DSM and game development tasks are woven together, can be found in Reference [18].

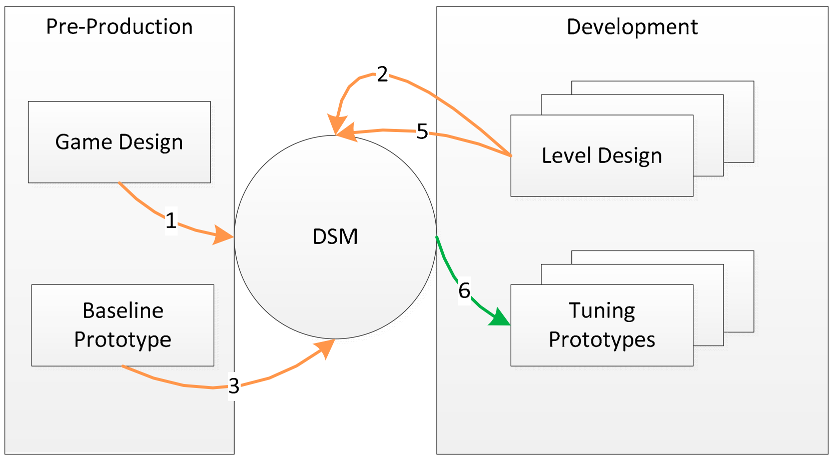

Figure 1. A Domain-Specific Modeling embedded workflow for game development [18].

2.4. Pervasive Game

To present our work in a more concrete way, we focus on pervasive (computer) games. This kind of computer games is emerging and attracting more and more interest recently due to the involvement of physical and social elements in the virtual game world as well as the development of sensor and mobile technology. Well known pervasive games include “Mobio Threat [19]” and “SupaFly [20]”. Typical pervasive games have features such as location awareness, using physical user interfaces, mobility and long lasting. In our previous research, more characteristics were summarized [21]. In Reference [22], the authors indicated that pervasive games aimed at combining the properties and advantages of three worlds: the physical worlds, social worlds, and virtual worlds. Furthermore, the authors proposed a meaning of pervasive games focused on user experience and pervasive games elements as follows: a pervasive game delivers to the player an enriched experience of game through an evolvement of the dynamics of the game, expanding the space of the game according to the context where it is played [23]. It allows the breaking of the boundaries of the game world, making reality part of it and that the elements of that reality have an influence during the game [23].

2.5. Ontology and Meta-Model

An ontology plays a role of a semantic domain in denotational semantics, while meta-modeling techniques play an important role in developing model description languages suitable for problem domains [24]. Meta-models define the structure (syntax and semantics) of the models to be built [25]. An ontology is defined as a specification of concepts and relationships between the concepts that can exist in a given setting [26]. Ontology and Meta-model are closely related. The meta-modeling technique can be used to develop an ontology. For instance, meta-modeling techniques were used to propose an awareness ontology that conceptualizes some of the most important aspects of awareness in a specific kind of system: collaborative systems for carrying out modeling activities [25]. Meta-modeling can be considered as an explicit description (constructs and rules) of how a domain-specific model is built. A valid meta-model is an ontology, but not all ontologies are modeled explicitly as meta-models [27]. In Reference [28], the author found that a language was ideal to represent phenomena in a given domain if the meta-model of this language was isomorphic to the ideal ontology of that domain. In this work, we use ontology to specify concepts for both game design and domain analysis. The Pervasive Game Ontology is organized according to the general structure of computer games. In addition, pervasive games specific concepts are included in PerGO. Accordingly, meta-models generated based on PerGO should be isomorphic to pervasive (computer) games domain, and in theory, the corresponding domain-specific language might be suitable for the designed pervasive game domain.

3. Pervasive Game Ontology (PerGO) Formalism

PerGO is designed to serve as a structured and efficient domain analysis. The motivation behind PerGO is to predefine a number of commonly used concepts (as well as the relationships between them) within the pervasive game domain and provide a structured way to produce domain analysis outputs with these concepts.

In total, there are more than 100 concepts in PerGO. These concepts are organized within 6 perspectives which focus on different aspects of game software individually. In addition, these concepts are of two levels. High-level concepts are common to all computer games and they constitute the core part of PerGO. Low-level concepts are specific to pervasive games (primarily used or often used in pervasive games). They are derived from high-level concepts and constitute the pervasive part of PerGO.

In this section, we introduce the design decisions in Section 3.1. Then we introduce the four perspectives as well as the core part concepts in Section 3.2, and the pervasive part concepts in Section 3.3.

3.1. Design Decisions

Concepts in PerGO are organized within six perspectives. The six perspectives were defined based on useful sources to identify DSL concepts [5] and on the current research focus of this article. In Reference [5], the authors suggested several useful sources to find language concepts, including the physical product structure, the look and feel of the system, variability space, domain (expert) concepts, and generation output. In this research, we predefine pervasive game concepts in PerGO according to domain concepts like the gameplay and AI, the look and feel of the system like the presentation, and the generation output like game elements. Then, we customized the concepts according to the variability space of a more specific domain like pervasive treasure hunting games. We refer to some well-known frameworks (see Section 7 for more details) to keep consistent with game design traditions. The sequence is decided among perspectives to handle potential design dependencies in an easier way. See Section 6 for more information about this.

PerGO concepts are of high-level or low-level. This way of splitting the overall ontology into two parts is common. Such two parts are usually called as "Upper-level Ontology" [29] or "Frame Ontology" and "Domain Ontology" [30]. In addition, concepts of the latter part are usually inherited from concepts of the former part [29,31].

Numerous concepts exist in one domain. PerGO focuses on concepts that could work as DSL concepts (ones that support code automation in addition to pure conceptual modelling). Three criteria are used when PerGO concepts are decided: (1) the concepts should be of proper complexity level. For instance, if one concept can be expressed by one variable of primitive type, it is thought to be of improper complexity and is excluded; (2) the concepts should be of proper abstraction level. We prefer to use concepts that usually appear in the discussion of game design or level design. Meanwhile, it is possible to implement such a concept as a class or some other encapsulated data structure. This is to balance between the expressiveness and the flexibility of the DSL; (3) the concepts should be constructive. In other words, the concepts should be named as nouns such as "Map", instead of adjectives, such as "interesting" or "intuitive".

A number of languages can be used to present ontologies [32]. In References [33,34], the possibility and feasibility of using UML as an ontology language were discussed. A UML class diagram [35] was chosen to formalize PerGO for several reasons. Firstly, PerGO will be further used to construct DSL meta-models, while such DSL meta-models are often presented in UML class diagrams as well. Using UML class diagram makes the transition between DA and DSL definition smooth. Secondly, for many software developers, UML is well known and the threshold of learning it is lower than other ontology languages. For instance, OWL [36] is not as widely known to software developers as UML class diagram. Thirdly, all standard hierarchical relations [37] (classification, aggregation, generalization, and association) that are needed to describe relationships among concepts have been included in a UML class diagram.

In a UML class diagram, classes and relationships among the classes are the main constituents. A class describes a concept and is represented as a rectangle. A relationship between two classes is drawn as a line. While a normal association relationship is drawn as a line with an arrow at the end, other kinds of relationships are represented as various kinds of lines. Association relationships are drawn as lines with a triangle at the end, and aggregation relationships are drawn as lines with a black diamond at the end. For dependency relationships, dotted lines are used instead of solid lines.

3.2. Perspectives and the Core Part of PerGO

In PerGO, six perspectives were defined from which useful concepts can be identified. These six perspectives, as well as most of the core part concepts included in them, are listed as follows with a brief introduction. Other concepts in the core part will be introduced further in Section 3.3 since they are used to derive more concepts in the pervasive part of PerGO.

- In the Gameplay perspective, all concepts that describe challenges and actions are included. While [Challenge] describes challenges that players need to overcome, [Action] describes actions that players could perform to overcome the challenges [38].
- In the Artificial Intelligence (AI) perspective, concepts that describe rules are included. Such rules are about how game elements react to players, or when no interaction happens, how game elements evolve. An abstract concept [AI] is defined in the core part of such rules.
- While Gameplay and AI depict dynamic characteristics of a game, GameWorldElement describes the static characteristics of the game. In this perspective, all concepts representing game elements like [Map], [Character], and [Element] are included.
- To make the game interactive, the Presentation perspective focuses on how a game utilizes various output devices to make the game status explicit and influence the physical world. A concept

[Presenter] and an interface [Presentable] are defined in the core part, standing for the presentation logic and objects that can be presented.

- Further, the Control perspective focuses on how game status can be controlled through various input devices. The concept [Controller] is defined referring to the control logic, and the interface [Controllable] is defined for objects that can be controlled (which implement the interface).
- Considering the complexity of control and presentation parts, the CtrlPresentation perspective is defined. While Control and Presentation are usually connected with one and only one I/O device, [CtrlPre] provides more flexible mechanics and can be connected to several different I/O devices. Such devices include not only traditional speakers and display, but also force feedback or other physical UI devices. CtrlPre can be connected to the Controllable and Presentable at the same time, but these two interfaces are implemented by one single game element. In this way, CtrlPre is differentiated from a combination of Controllers and Presenters.

In Figure 2, an overview of the core part is presented. In this figure, concepts, as well as relationships among these concepts (as introduced above), are presented.

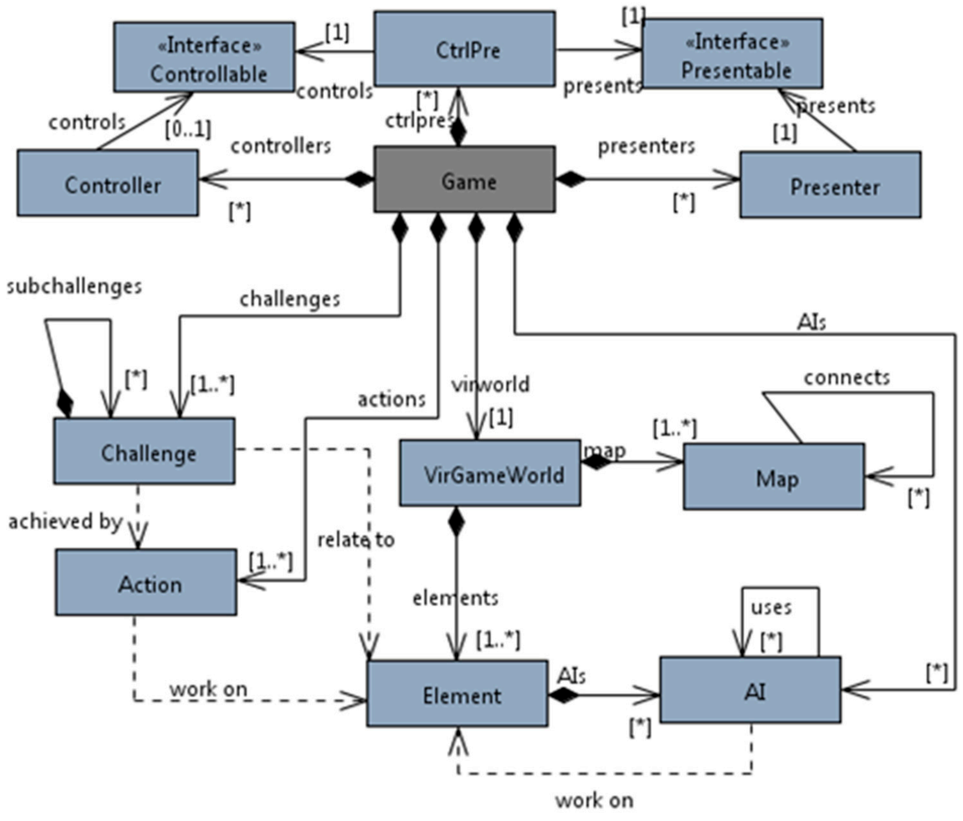

Figure 2. The core part of Pervasive Game Ontology

3.3. The Pervasive Part of PerGO

In the following sub-sections, the concepts in the pervasive part of PerGO will be briefly introduced. In each section, an overview figure is presented which shows all the concepts and relationships from the perspective. In these figures, colored concepts (core concepts) are within the core part (grey ones from the current perspective, and yellow ones from other perspectives), while the white concepts are in the pervasive part. All the concepts in the pervasive part are derived from core concepts. We focus on concepts that are primarily used in pervasive games. Concepts that are used by both pervasive games and general computer games will not be discussed in detail as they are fairly well known.

3.3.1. Gameplay

Two kinds of concepts are included in this perspective. Challenge concepts depict goals that players need to fulfill and Action concepts describe behavior that players are allowed to perform to overcome such challenges. In some games, many challenges can be achieved by one action. For instance, Challenges like MoveToCha (moving to a place), GetCloseCha (getting close to a place or an area), EnterCha (entering an area), and AvoidCha (avoiding getting close to a place or an area) can be achieved by one action Move. On the other hand, it is also possible that one challenge can be fulfilled by performing different actions. For instance, GetObjCha can be achieved by Move (by co-locating only) or by Move and Get (by co-locating and an explicit action to get). Challenges and Actions must be designed carefully in order to achieve unique and interesting game experiences.

Challenge concepts are classified according to whether they can be fulfilled by an individual player by collaborating with other players, or by competing with other players. Accordingly, these concepts are derived from CollaborateChallenge and CompeteChallenge. In Figure 3, an overview of the concepts is shown.

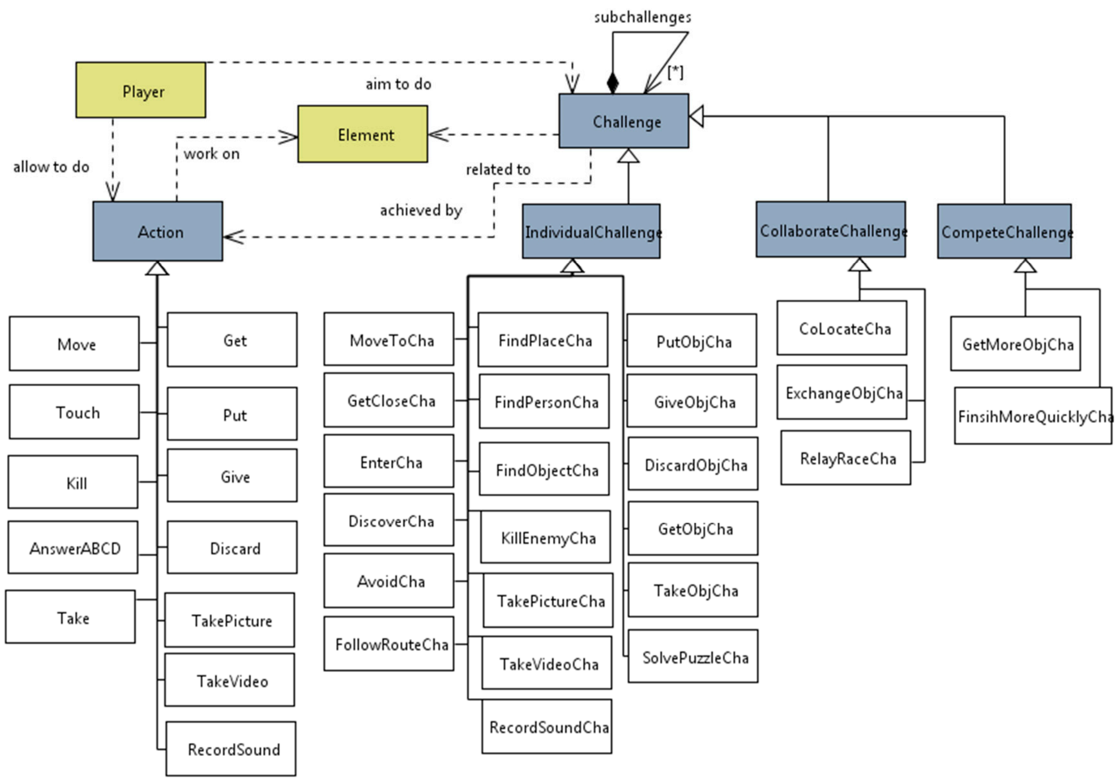

Figure 3. The pervasive game concepts in the gameplay perspective.

3.3.2. AI

In the AI perspective (as shown in Figure 4), two kinds of concepts are included. These concepts are derived from InteractionRule and EvolutionRule. Interaction rules are used to describe how game elements interact with each other while evolution rules are used to describe how game elements behave by themselves. Collide may be one of the most commonly used interaction rules and it is used to describe under which conditions a collision happens. For instance, when two elements get close enough, a collision happens. Additionally, when a collision happens, various events are often invoked. As for evolution rules, Duplicate describes how one element reproduces under specific conditions, MoveTo describes how one element finds the path from one place to another place, and so forth, are included in this perspective. In addition, for pervasive games, it is common that virtual elements status is updated according to the data captured from the physical world. UpdatePhysicalInfo is used to describe updating rules under such circumstances. On the other hand, it has become popular that when virtual elements status is updated, players' social network (Facebook [39], Twitter [40], and so forth) information is updated correspondingly. UpdateSocialInfo can be used then.

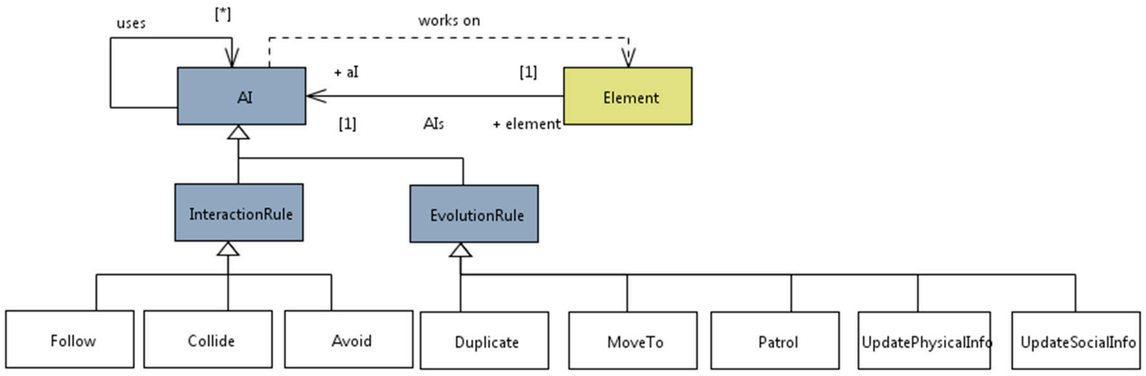

Figure 4. The pervasive game concepts in the Artificial Intelligence perspective.

3.3.3. Virtual Game World

In the virtual game world (see Figure 5), there is a Map and some map elements such as Point (indicating a position), Route (indicating a series of points), and Place (indicating an area). These MapElements are expressed in different ways according to the property of the map. For instance, in a map of a location-based game, each Point may be expressed with a pair of latitude and longitude values. However, in a map of a traditional 2D virtual game, Points are usually expressed with a pair of x and y value indicating horizontal and vertical coordinates in the 2-dimension virtual space. In addition to map and map elements, there are some WorldElements which are located or moved on the map. Commonly used elements are like Character (a figure of a human being), PlayerProxy (the character that is controlled by players), Group (a group of PlayerProxies which collaborate to play the game), and Community (a social network of PlayerProxies).

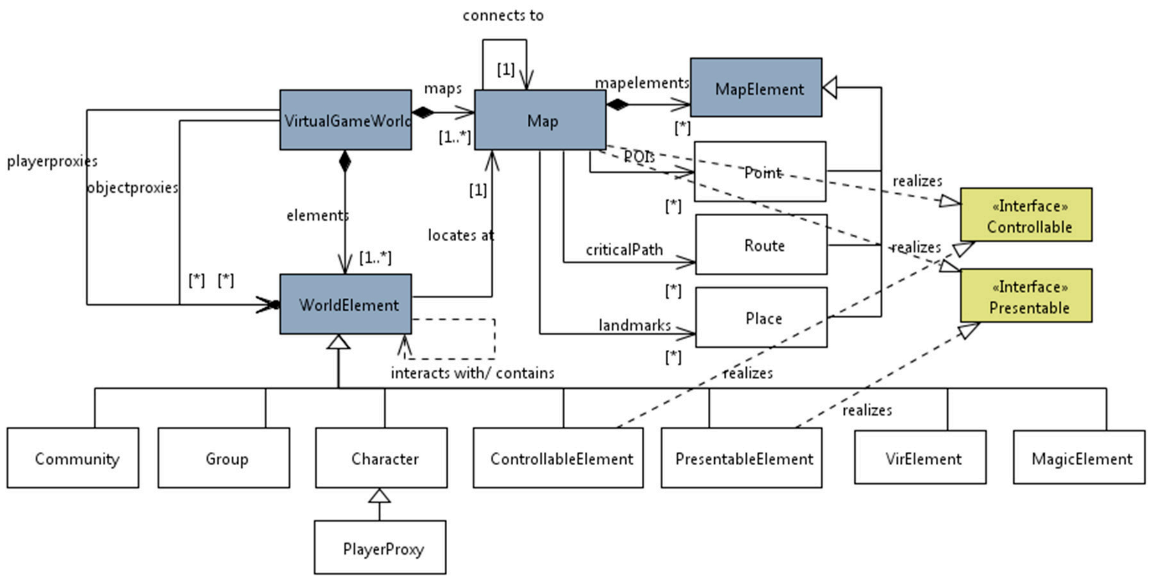

Figure 5. The pervasive game concepts in the virtual game world perspective.

3.3.4. Control

In traditional computer games, general input devices such as mice and keyboards are used by players to control the game. Touch screens are often used nowadays as smartphones are getting popular. In pervasive games, various kinds of input devices are utilized to introduce the status of physical elements in the game world. Two kinds of Controllers are defined in this perspective: HumanPhyController sensing players' physical information and EnvironmentController sensing physical information in the real world.

When HumanPhyController is involved, physical sensors including cameras, microphones, or gyroscopes are utilized to measure the player's physical information to change game status or invoke actions. Such information may be about physical location or physiological signals (for example, heart rate, brainwaves) of human beings. HumanPhyController also transforms data to meaningful or more structured formats from its original format, generated in control devices. For example, when location information is sensed in a GPS device, RouteCtrl (which is connected to the GPS device) not only records location continually but also calculates routes according to and reacts according to the routes. On the other hand, in pervasive games, sensors are also used to capture information in the physical environment (such as temperature and wind direction) to control the game status. EnvironmentControllers are used at that time. An overview of Controllers is shown in Figure 6.

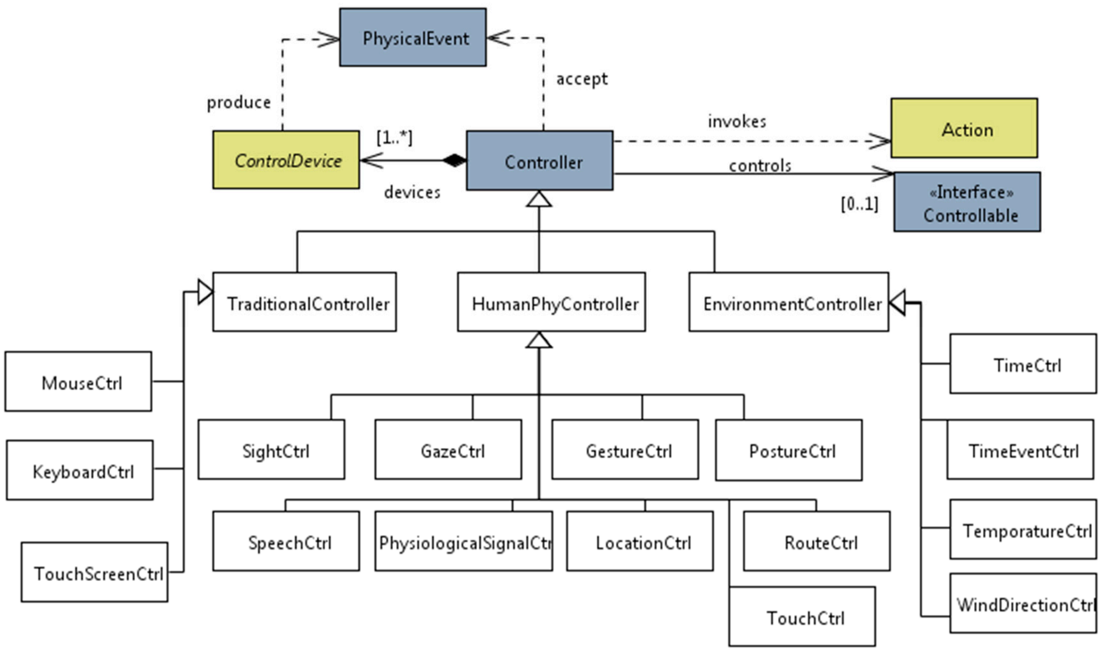

Figure 6. The pervasive game concepts in the control perspective.

3.3.5. Presentation

In Figure 7, concepts in the Presentation perspective are presented. In pervasive games, in addition to general video (VisualPre) and audio (AudioPre) presentation, the physical (PhyPre) and electronic (ElePre) approaches are also used to present game status. Typical physical approaches include producing force feedback (ForceFeedback), moving physical objects (MoveObject) and typical electronic approaches such as capturing a picture (CapturePicture)/video(CaptureVideo)/sound(CaptureSound), making a call (MakeCall), sending a short message (SendSMS), sending an email (SendEmail), or pushing data to update the social network status (ShareOnSocialApp).

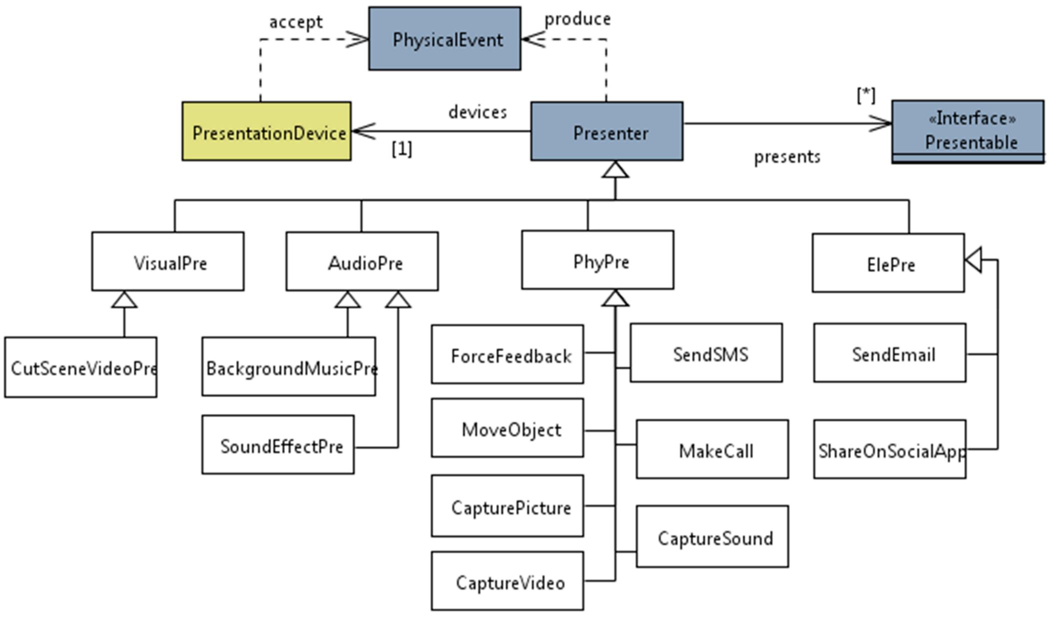

Figure 7. The pervasive game concepts in the presentation perspective.

3.3.6. CtrlPresentation

In this perspective (as shown in Figure 8), concepts regarding logic for control and presentation at the same time are included. Generally used concepts include GUIElement (Graphical User Interface elements), MapView (the main view showing the overall map), and AirView (smaller view overlapping on the main view and presenting a simplified bird's eye view). In addition, PhysicalUIManager (to interact with physical UI devices) and MixedView (mixing the real sight of the player and virtual elements) are often used in pervasive games.

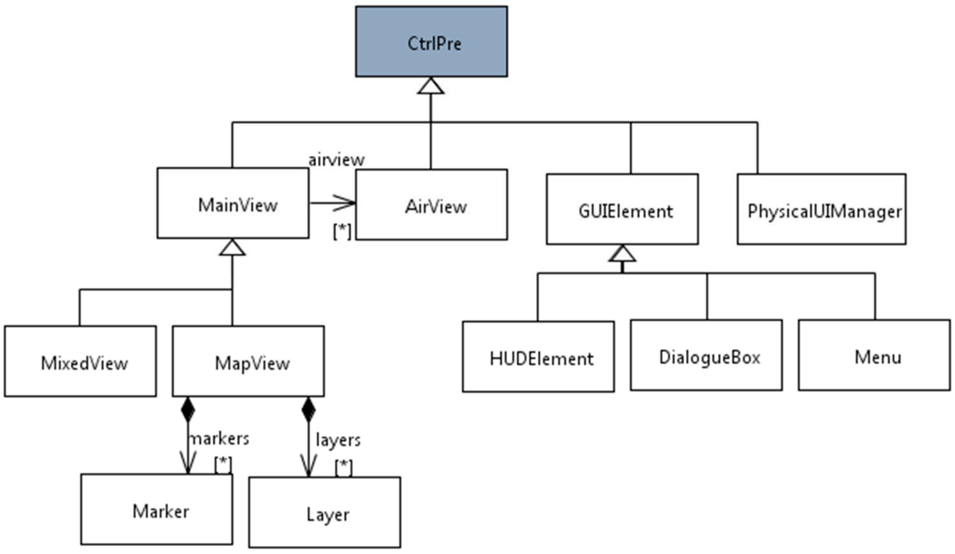

Figure 8. The pervasive game concepts in the CtrlPresentation perspective.

4. PerGO Based Domain Analysis Procedure

Based on PerGO, a four-step procedure is proposed to produce DA outputs in a structured way. As previously introduced, such outputs include a domain vocabulary, commonality space, and variability space. Table 1 was used when we analyze, collect, and produce corresponding information. The four steps are:

1. Quickly identify perspectives corresponding to the current domain. Record these perspectives in the first column;
2. Go through the perspectives in the first column. For each perspective, consider the common game design and select useful concepts from PerGO or derive more specific concepts based on PerGO to represent the design. Record these concepts in the second column. If more detailed information should be specified, especially attributes of the concepts, record the details in the third column;
3. Go through the perspectives in the first column again. However, at this time, consider the variable game design that will be used in different game samples. Decide whether more concepts are needed in addition to those currently in the second column. Select the concepts in PerGO or make new abstractions, and add them in the second column;
4. Go through all the concepts in the second column, and decide how to utilize the concept attributes or concept relationships to support variable game design. Then write them in the last column.

Table 1. The domain analysis table.

Perspective	Concept	Commonality Details	Variability Details

The four steps help designers analyze in a reasonable order, from general aspects (perspectives) to concrete aspects (concepts, attributes, and relationships), and from common aspects to variable aspects (variability often depends on commonality). Further, whenever the perspectives are gone through, they should be gone through according to the sequence presented here: from Gameplay to AI, VGW, Control, Presentation, and Ctrl-Presentation. The rationale behind this order will be explained in Section 6.

By using the ontology to structure and accelerate the production of the main domain analysis outputs (concepts, commonality, and variability), it is expected that more qualified DSLs can be produced in an efficient way. Two benefits are expected: (1) to accelerate the domain analysis process by proposing a number of premade domain-specific artefacts (concepts and relations) which can be used directly or with minor modification to generate domain analysis outputs; (2) to regulate and structure the domain analysis process by proposing ordered steps based on PerGO.

5. A Pervasive Game Example

In this section, a pervasive game "PervasiveTreasureHunting" is analyzed as an example of how to use our proposed DA procedure. The use of exemplars is widely recognized as a technique for the early evaluation of modeling approaches [41]. Most of the features of this game example have been previously implemented in several prototypes, and thus, the description can work as a simplified game design document. How to carry out domain analysis with PerGO by following the steps introduced in Section 4 is illustrated. The game description is as follows:

PervasiveTreasureHunting is one location-aware variation of traditional treasure hunting games. The game allows several groups of players to physically move within the Trondheim downtown area and play the game. They try to get notes from predefined places, finish the tasks described in these notes, and compete with each other according to which group or which player has the most game points collected by getting notes or finishing tasks. A predefined route can be made covering the places, and in that case, players will have to follow the route to get the notes. Different tasks can include moving to someplace, taking a picture of something, taking some objects from somewhere, and exchanging objects with other group members. Players may not get the game points associated with a post or a task if they use more time than the predefined time limit to find the post or solve the task.

Mobile devices are needed to play the browser game. The game should present a map that is annotated with places as well as the positions of players, and other game status information (like the amount of time past, the amount of time left, and who is the winner). The client-side application should allow players to login with a unique group ID and player ID, and present information in different ways, accordingly. For instance, the icons of players within the same group of the current player should be displayed with a different transparency than those icons of other group players. For the server side of the application, it is required to use a map with a different style (a dark one ideally) and all the player icons are highlighted with full transparency. The players play the game mainly by physically moving, which will cause the player proxies to move in the game world. Players use one hotkey to read notes or solve tasks.

The results of the domain analysis are presented in Table 2. Due to the limited space, only part of the results is presented here. The upper part of Table 2 shows the concepts in the Virtual Game World perspective. These concepts are usually easy to find if we pay more attention to the nouns in the game description (see the words highlighted in yellow above). Some of the concepts need to be renamed in order to align with PerGO and to maintain consistency with other DSLs that are developed based on PerGO. For example, PlayerProxy is used (from PerGO) instead of Player (which is used in the description) as the concept name. It is a little tricky to decide whether an attribute should be specified in the commonality (Step2) or variability (Step4) section. The key is, if the attribute needs to be assigned when constructing the codes, then the attribute should be specified in the variability section. If the attribute needs to be assigned at run time, it should be specified in the commonality section (meaning all the designs need this attribute). For instance, Game Points of a Note should be defined before the codes are constructed (in variability). While Game Points of a PlayerProxy should be calculated at run time (in commonality). These should be considered carefully because the content in commonality may not contribute to the meta-model of the DSL.

Table 2. The domain analysis table for the PervasiveTreasureHunting game.

Perspective	Concept	Commonality Details (Attributes)	Variability Details (Attributes & Relationships)
Virtual Game World	Game	Name, Introduction, TimePast, TimeLeft, Winner	MaxNumberOfGroup, MaxNumberOfPlayer, GroupWinnerOr-IndividualWinner
	Group	ID, GamePoint	Relates to Player
	PlayerProxy	ID, GamePoint	Relates to Group
	Note		ID, Position, GamePoints, TimeLimit, relates to Task
	Task	Name, TaskDescription	ID, GamePoints, TimeLimit, relates to Note, relates to Challenge
	...		
Gameplay	TakePictureCha		PictureID (to judge whether the challenge is fulfilled)
	TakePicture		Relates to KeyboardCtrl (to invoke the action)
	...		

The lower part of Table 2 shows a Challenge concept (TakePictureCha) and an Action concept (TakePicture). Candidates for these concepts can be found by observing the verbs that are used to describe scenarios about how players play the game (see the underlined words in the game description). Besides textual description for the specific game, sometimes additional details for the commonality and variability need to be complemented according to the semantics and common knowledge. As the variability shows, a KeyboardCtrl should be related to the action. Consequently, when a key is pressed, the action will be invoked. On the other hand, a picture ID should be assigned to the challenge, so that when a picture is taken, the challenge can judge whether it fulfills the predefined requirements (sharing some common objects of the known picture with the ID, for instance).

After the table is filled in, it is time to construct an initial meta-model for the DSL. All the nodes were created corresponding to the concepts found in the table. If some concepts did not have any variability details, the corresponding nodes were not created since they were not going to be used to decide how to generate the code. Abstraction nodes were then added from PerGO, which can work as a container to contain the previous nodes, or as a parent to derive the previous nodes. All the concepts were organized as an aggregation tree, as it was required by the MDD code generator. Attributes and other associations were added lastly, indicated in the variability section details. Figure 9 shows the result of the meta-model corresponding to the previous domain analysis (attributes are not included due to the limited space). From the figure, we can see that most of the concepts are directly from PerGO (the colored ones).

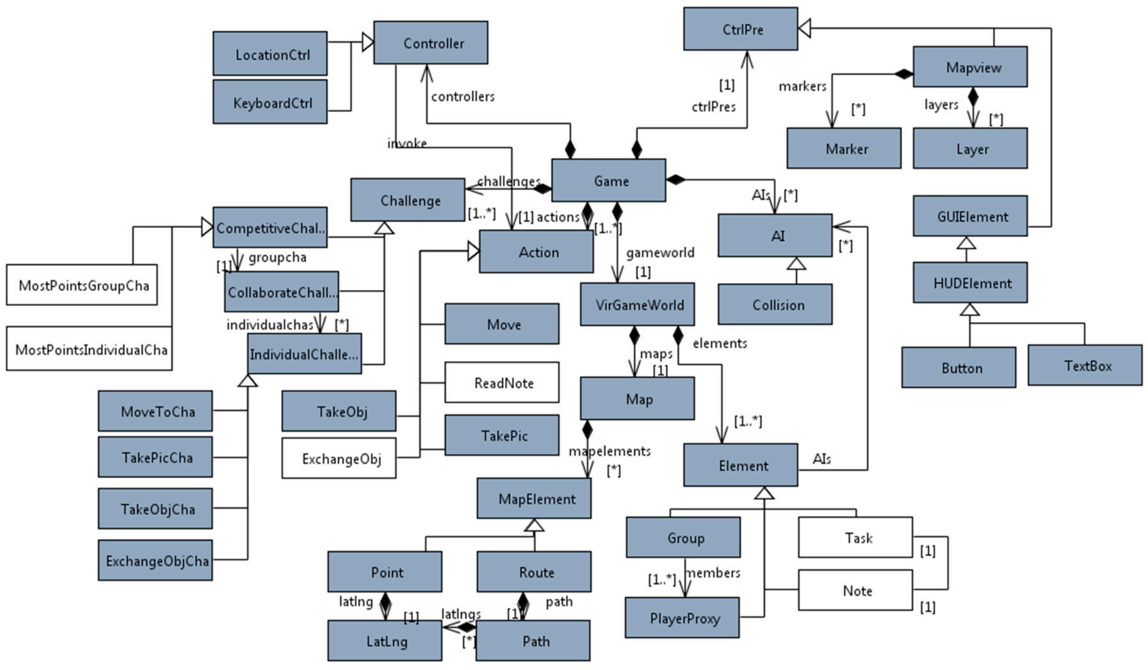

Figure 9. The initial meta-model for the case study.

By utilizing the proposed ontology, the domain analysis was carried out in a structured (four steps to produce outputs) and efficient way (36 out of 42 concepts in the meta-model are from the ontology in the case).

6. Related Work

As mentioned previously, the application of DSM in computer game domains is not quite widespread. Few articles have been found where MDD application was reported in game development. Among them, it is even more difficult to find comprehensive descriptions about how their domain analysis was carried out, or where the abstractions in their DSLs came from.

Some scientific articles were reviewed in the MDGD domain to gain an understanding of how domain analysis process was performed. The main aspects reviewed were as below:

1. Was any formal DA methodology (like OBDA [9]) adopted?
2. Was the DA process structured (with clarified framework or steps)?
3. Was there any predefined vocabulary that can be used directly in the DA process?
4. Was the predefined vocabulary full-spectrum instead of focusing on one or two specific aspects of the game?

It is actually difficult to find many papers presenting DSL or DSM practices in the computer game domain. Therefore, the review included some unfinished work and some work using MDA and UML as well (such as C and B, since they are comparatively more related). Most papers introduced DSLs or workflow without illustrating much of the DA part in particular. Table 3 summarizes the main findings. In the table, each literature column is used for a work which is introduced in one or several scientific papers. The "*" symbol is used to indicate that the work meets the criteria specified in the first column. Additionally, "s" is used to indicate that the work meets the criteria in a semi-finished way: using a semi-structured process or semi-formal vocabulary.

Table 3. The domain analysis in the computer game domain.

	Literature									
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
1 Formal DA Methods										*
2 Structured DA/DD	*	s	s	s						
3 Predefined Vocabulary	*	s								
4 Full Spectrum Vocabulary	*	*								
5 DSL (Not GPL)	*			*	*	*	*	*	*	*

A: [42,43] ([44,45]); B: [46] ([47,48]); C: [49,50]; D: [17]; E: [51]; F: [13]; G: [15]; H: [52]; I: [53]; J: [54]

The review revealed that no paper claimed to have used any formal DA methodology. Many of the papers proposed DSL concepts directly such as E, F, G, H, and I [13,15,51–53], while some other papers identified several dimensions which contained the DSL concepts. For instance, in D [17], two dimensions (visible structure and gameplay related entities) were identified. In C [50], the structure, behavior, and control mapping were used to structure the domain designs. Furthermore, in C [49], although domain analysis was not mentioned explicitly, the same authors identified six areas to address game design concerns including gameplay, graphics interface, audio design, artificial intelligence, game storytelling, and level design. Among them, only gameplay was addressed in the DSL. These practices are thought of as semi-structured domain analysis/design.

References [42,43] may be the work that is closest to our approach. In Reference [42], the authors proposed ten dimensions to define a product line including User interface, Game flow, AI, Sound/Music, and so forth. A more detailed ontology introduced in Reference [43] was used to define the SharpLudus product line (as the main part of the domain analysis task) and create the corresponding DSLs. However, the root concepts in the ontology did not match the ten dimensions of the product line definition or the top level DSL concepts in an explicit way. Exactly how the product line definition (domain analysis) contributed to the construction of DSL concepts based on the ontology remains a question. In B [46], a conceptual framework was introduced which consisted of a three tier design architecture (flow, scenarios, and objects) and components (screen components, GUI components, In-game components, and so forth) for serious games. This conceptual framework structures the abstractions and relationships, but might not be able to work as a domain analysis structure in a common way. Also, the framework encompassed serious games instead of common computer games. Some concepts were enumerated in the framework, which, for our purpose, can be thought of as a semi-formal vocabulary. J [54] is the only work in which traditional formal domain analysis (FODA) was used. As indicated in the paper, different types of existing games and their internal features have been analyzed in order for users to develop any type of 2D videogame of touch ability, puzzle, platform, trivia style, and turn based strategy video games. In the paper, a meta-model was presented but without an introduction about the theoretical basis. The feature model was not presented as well.

As a result, although using the formal methodology and predefined domain knowledge (for example, as vocabulary) were thought useful, these were generally not applied in MDGD. In the approach in this research, the attempted remedy was to propose a structured DA process based on a full scale pre-defined domain vocabulary.

7. Discussion and Evaluation

In this section, we discuss the quality of our proposed DA approach, both in a qualitative way and with a user survey. In addition, we talk about usage scenario of this approach, as well as cost and limitations.

7.1. Towards an Effective and Efficient Domain Analysis Method

As introduced previously, a formal domain analysis method would be based on domain vocabulary in order to reuse the complex domain knowledge. To ensure the method's effectiveness and efficiency, the following criteria are proposed to evaluate such a method (and the domain vocabulary that the method is based on):

1. (Effectiveness) The vocabulary should support the modeling problem domain features. Domain-specific characteristics can be captured by using vocabulary concepts;
2. (Effectiveness) The vocabulary should support generating solution domain artifacts. Concepts within the vocabulary should be constructive, be of proper abstraction level and complexity level;
3. (Efficiency) The overall process should be reasonably ordered to solve the design dependencies/constraints. For instance, variabilities are usually decided based on commonalities.

7.1.1. The Effectiveness of Vocabulary

Three criteria were used when selecting such concepts to ensure that the concepts in PerGO were constructive and with a proper level of abstraction and complexity. As a domain vocabulary, it is also very important that such concepts should be able to capture the characteristics of the target domain effectively.

As suggested in Reference [5], we proposed the six main perspectives of PerGO according to channels where DSL concepts can be found. We evaluated if this structure of PerGO is in line with important conceptual frameworks in the domain of computer games. In this way, the future extension and evolvement of PerGO could be made easier. In Reference [55], the authors proposed to automate game design as a pipelined process: "begin with an abstract game, represent it, add a thematic content "skin", and set up control mappings". Perspectives in PerGO have been identified and organized in a similar way. In PerGO, Gameplay, AI, and the Virtual Game World are about the core rules of a game, and they constitute an "abstract game". Further, Presentation, Control, and CtrlPresentation enable players to interact with the game (represent and control). Narration can be thought of as the "thematic content skin" which may be extended later in PerGO. In Reference [56], the author proposed four orthogonal dimensions to separate high level game design concerns: Simulation, Ludology, Narratology, and Gambling. PerGO supports game design in these dimensions. For example, simulation can be achieved by the cooperation of the virtual game world with AI and gameplay. Ludology is achieved by control, gameplay, and AI. While Gambling can be implemented with logic encapsulated in AI. Again, Narratology is expected to be supported in the future. Additionally, the basic idea about Narration is that, in addition to current concepts, more concepts such as scenes, plots, characters regarding a story would be included in PerGO. In addition, there might be relationships between concepts in the narration perspective and concepts in existed perspectives. For instance, one map may be needed for one scene.

In addition to general computer games, PerGO should also support special features of pervasive games. In one earlier research of ours, more than 30 pervasive games were reviewed and the main characteristics of pervasive games were summarized in one conceptual framework named TeMPS [21]. While TeMPS was designed to understand and analyze pervasive games, PerGO is designed to construct pervasive games. We examine whether the PerGO concepts can be used to realize TeMPS features. There are four perspectives in TeMPS: Temporality, Mobility, Perceptibility, and Sociality. Temporality can be realized with an attribute of a Game concept in PerGO. Mobility can be realized with Map, MapElement, and MapView concepts. Additionally, to realize features in Perceptibility, a combination of Action concepts, Controller concepts (HumanPhyController or EnvironmentController for instance), and Presenter concepts (such as PhyPre or ElePre) can be used. Sociality is one perspective that has not been covered well enough in the current version of PerGO. We have planned this for future work.

From the experience of analyzing the case in Section 5, we feel that it is natural to follow the structure of PerGO to carry out DA tasks. We think the overall structure (perspectives and the core part) of PerGO is clear and easy to use. In addition, most of the concepts needed were already included in PerGO so that they could be used directly (38 concepts out of the overall 44 concepts).

7.1.2. The Efficiency as a Process

Game domain sometimes is really complex and considerations pertaining to the various aspects of the domain weave with each other often. To keep the overall domain analysis efficient, the sequence to frame analysis should solve the underlying dependencies well. The sequence is discussed in both the procedure steps and the perspectives.

As previously introduced, in the DA process, step 1 is to go through perspectives, while step 2 is to go through concepts for common game design, and step 3 and step 4 are to go through concepts for variable game design within each perspective. This indicates two sequences from wide to narrow (from perspectives to concepts within them), and from common to special (commonality to variability). Both are decided based on the dependency relationship.

Further, among the perspectives, an ordered sequence was suggested from Gameplay to AI, Game Element (Virtual Game World), Control, and Presentation (also CtrlPresentation). This sequence is defined based on the general game design traditions [38,55], such as (1) gameplay always takes the priority (whether a game can succeed depends to a large extent on the gameplay); (2) after thinking about how the player interacts with game elements (Gameplay), it is natural to think about how game elements react to the player (AI); (3) when gameplay and AI are decided, it is easier to enumerate all the world elements that could support the gameplay and AI; (4) decide how to control it and present it.

When going through concepts within a perspective, in the case where there are more than ten concepts, it is suggested to go through them according to which core concepts they are derived from. For example, Gameplay concepts should be gone through in the order of IndividualChallenge derived concepts to CollaborateChallenge derived concepts, then CompeteChallenge derived concepts and Action derived concepts. In this way, indexing and finding concepts can be more efficient.

By addressing the potential dependency among the analysis, several benefits are expected: (1) analyzers would feel natural and comfortable enough to follow the process; (2) concepts can be found in PerGO quickly; (3) it is less necessary to go back and forth to accomplish the overall analysis. As was experienced in the case study (see Section 5), almost no situations were met where it was necessary to go back and force the domain analysis process to finish. In the overall process, the steps and perspective sequence were followed comfortably, and the concepts were always found quickly.

7.2. User Acceptance Tests

In addition to internal theoretical evaluation, we also tried to apply our DA approach in a complete MDD approach (as introduced in Section 2.3) and evaluated potential user acceptance degree towards both PerGO and the overall MDD approach [57]. We investigated constructs defined in the Technology Acceptance Model (TAM) [58] which is a widely used research model to test attitude and the intention of users to adopt new technologies. TAM defined three main constructs: Perceived Usefulness (PU) as the extent to which potential users believe that using a technology will improve the job performance, Perceived Ease of Use (PEOU) as the extent to which potential users believe using the technology will be free of effort, and Intention to Use (IU) as a predictor of usage behavior. TAM claims that

IU can be explained by PU and PEOU, and PU is determined by PEOU. More practical lessons we have learned can be found in Reference [59]. We performed the survey among around 50 persons in Norway. As most of the recruitment happened within universities, the education of participants is comparatively high (bachelors or above). As a result, 47 persons were recruited to perform the survey, and 46 of them answered all questions (one person left answers to some questions blank). According to the result, most of the external potential users considered the ontology useful (the mean score of PU is 3.9 out of 5) and easy to use (with the mean PEOU score of 3.37). Most people intended to use PerGO in the future (3.75 as the mean IU value). More details about the user evaluation can be found in Reference [57].

7.3. Usage Scenario, Costs and Limitations

PerGO is named as Pervasive Game Ontology, indicating that it is served for pervasive games domain primarily. However, as introduced previously, the perspectives and core part of PerGO is designed to be general for all computer games. Therefore, other ontologies can be developed based on the core part for other subdomains of computer games for MDD usage. Further, as the domain analysis procedure only depends on the perspective concept structure of domain vocabularies, it can be used for other domains as long as the domain vocabulary is organized with the same structure.

The motivation of utilizing the proposed approach (both the DA procedure and the PerGO ontology) is to save time by reusing domain knowledge in an efficient way. This approach may not be worthy for all domains since building an ontology and evolving it also incurs costs. It was used for the pervasive game domain because it is new, and since there is no commonly agreed definition for such games. A predefined knowledge base helps people, especially non domain experts, to be able to design and develop such games better and more quickly. Compared to building PerGO, maintaining PerGO demanded fewer resources. In the earlier stages of the research, there were more times when it was felt that it was necessary to enhance the ontology structure or concepts and more resources were used to maintain PerGO. However, the ontology gradually became more mature, and by the time the case study in Section 5 was performed, most of the maintenance work was just adding a few new concepts or changing concept names to distinguish them from each other. Such maintenance work costs very little.

However, regarding the limitations, the most prominent limitation at present is that PerGO is not formal and mature enough from a technical perspective. We have put our focus on the infrastructure construction and the feasibility evaluation of PerGO previously. We would pay more attention to the formalization and technical quality in the future. We will define the semantics in a more and sufficient way. We will perform more case studies to refine the structure and concepts. Furthermore, there are some perspectives that were not included in the current version of the ontology, such as sociality and narration (as mentioned previously). Lastly, the proposed ontology and the procedure to carry out the domain analysis based on the ontology need to be put into a more complete MDD process to evaluate its gains and costs. The plan for how to compensate for these limitations will be discussed later.

8. Conclusions and Future Work

In this article, we proposed the Pervasive Game Ontology, which contains more than 100 pre-defined concepts and their relationships as a base to regulate and accelerate domain analysis for pervasive game DSL development. We demonstrated the usage of it through a case study. By utilizing our proposed ontology, the domain analysis was carried out in a structured (four steps to produce outputs) and efficient way (36 out of 42 concepts in the meta-model are from the ontology in our case). We reviewed the related work in the model driven game development field. The review shows that although domain analysis plays an important role in model driven development, most of these works have not employed formal DA methodology, or even depicted detailed information about the DA tasks. Few of them used pre-defined vocabulary or structured the process similar to PerGO. We evaluated our approach in a qualitative way and with a user survey. The qualitative evaluation shows that PerGO helps overcome the effectiveness and efficiency challenges that are common to many model-driven development approaches. The user survey shows that most of the potential users considered PerGO useful and easy to use.

We discussed the limitations of the current research result. We understood that the current ontology has not been refined and evaluated sufficiently. Thus, we identified three directions for possible future work: first, we will refine and extend PerGO. It is not possible that one DSL/DSM can be designed once and used forever, and "a large part of its power comes from its ability to evolve" [5]. We need to refine and evolve the ontology by carrying out more case studies. Additionally, we also plan to extend on some other perspectives like narration and sociality. For instance, the authors included storytelling in the design of education video games to motivate students' participation [60]. Second, we may enhance an overall MDD approach for game creation to support the domain analysis and other following MDGD activities. PerGO is designed to make domain analysis more structured and efficient. However, the ultimate goal is to improve the DSM definition and the overall game development. Research questions of interests can include how to utilize domain analysis result to produce DSM artifacts, how to integrate DSM tasks in the traditional game creation process, how traditional game creation participants adapt themselves to new tasks, and how traditional documents used in game creation process can be reused in the new process. Last, after setting up an overall workflow, we need to carry out more a formal and accurate evaluation (especially on the improvement on the productivity) on our ontology (and to improve them in the next iterations).

Author Contributions: All the authors have contributed to the work presented in the paper. H.G., Hallvard Trætteberg, A.I.W. and L.J. contributed to the design work of PerGO. H.G., S.G., H.T. contributed to the evaluation work of PerGO. H.G. and S.G. analyzed the data collected in the evaluation work. All authors worked collaboratively on writing the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mernik, M.; Heering, J.; Sloane, A.M. When and how to develop domain-specific languages. *ACM Comput. Surv.* **2005**, *37*, 316–344. [CrossRef]
2. Selic, B. The pragmatics of model-driven development. *IEEE Softw.* **2003**, *20*, 19–25. [CrossRef]
3. Stahl, T.; Voelter, M.; Czarnecki, K. *Model-Driven Software Development: Technology, Engineering, Management*; John Wiley & Sons: Hoboken, NJ, USA, 2006.
4. Atkinson, C.; Kuhne, T. Model-driven development: A metamodeling foundation. *IEEE Softw.* **2003**, *20*, 36–41. [CrossRef]
5. Kelly, S.; Tolvanen, J.-P. *Domain-Specific Modeling Enabling Full Code Generation*; John Wiley & Sons: Hoboken, NJ, USA, 2008.
6. France, R.; Rumpe, B. Model-driven development of complex software: A research roadmap. In Proceedings of the 2007 Future of Software Engineering, Minneapolis, MN, USA, 23–25 May 2007.
7. Pohl, K.; Böckle, G.; van der Linden, F. *Software Product Line Engineering*; Springer: Berlin, Germany, 2005; Volume 10, pp. 3–540.
8. Kang, K.C.; Cohen, S.G.; Hess, J.A.; Novak, W.E.; Peterson, A.S. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*; Carnegie-Mellon University, Pittsburgh Software Engineering Institute: Pittsburgh, PA, USA, November 1990.
9. Tairas, R.; Mernik, M.; Gray, J. *Using Ontologies in the Domain Analysis of Domain-Specific Languages*; Springer: Berlin, Germany, 2009.
10. Ceh, I.; Crepinšek, M.; Kosar, T.; Mernik, M. Ontology driven development of domain-specific languages. *Comput. Sci. Inf. Syst.* **2011**, *8*, 317–342. [CrossRef]
11. Falbo, R.d.A.; Guizzardi, G.; Duarte, K.C. An ontological approach to domain engineering. In Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, Ischia, Italy, 15–19 July 2002.

12. Furtado, A.W.B.; Santos, A.L.M.; Ramalho, G.L. SharpLudus revisited: From ad hoc and monolithic digital game DSLs to effectively customized DSM approaches. In Proceedings of the Compilation of the Co-Located Workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11, Portland, OR, USA, 23–24 October 2011; pp. 57–62.
13. Moreno-Ger, P.; Sierra, J.L.; Martínez-Ortiz, I.; Fernández-Manjón, B. A documental approach to adventure game development. *Sci. Comput. Program.* **2007**, *67*, 3–31. [CrossRef]
14. Albright, R.; Demers, A.; Gehrke, J.; Gupta, N.; Lee, H.; Keilty, R.; Sadowski, G.; Sowell, B.; White, W. SGL: A scalable language for data-driven games. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Vancouver, BC, Canada, 9–12 June 2008.
15. Hernandez, F.E.; Ortega, F.R. Eberos GML2D: A graphical domain-specific language for modeling 2D video games. In Proceedings of the 10th Workshop on Domain-Specific Modeling, Reno, NV, USA, 17–21 October 2010; p. 1.
16. Behrens, H. Mdsd for the iphone: Developing a domain-specific language and ide tooling to produce real world applications for mobile devices. In Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion, Reno/Tahoe, NV, USA, 17–21 October 2010.
17. Walter, R.; Masuch, M. How to integrate domain-specific languages into the game development process. In Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology, Lisbon, Portugal, 8–11 November 2011; pp. 1–8.
18. Guo, H.; Wang, A.I.; Gao, S. A Workflow for Model Driven Game Development. In Proceedings of the IEEE International Enterprise Distributed Object Computing Conference, Adelaide, SA, Australia, 21–25 September 2015.
19. Segatto, W.; Herzer, E.; Mazzotti, C.L.; Bittencourt, J.R.; Barbosa, J. Mobio threat: A mobile game based on the integration of wireless technologies. *Comput. Entertain.* **2008**, *6*. [CrossRef]
20. Jegers, K.; Wiberg, M. Pervasive gaming in the everyday world. *IEEE Perv. Comput.* **2006**, *5*, 78–85. [CrossRef]
21. Guo, H.; Trætteberg, H.; Wang, A.I.; Zhu, M. TeMPS: A Conceptual Framework for Pervasive and Social Games. In Proceedings of the IEEE 3rd International Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL 2010), Kaohsiung, Taiwan, 12–16 April 2010.
22. Hinske, S.; Lampe, M.; Magerkurth, C.; Rcker, C. Classifying pervasive games: On pervasive computing and mixed reality. In *Concepts and Technologies for Pervasive Games—A Reader for Pervasive Gaming Research*; Shaker Verlag: Herzogenrath, Germany, 2007; Volume 1.
23. Arango-López, J.; Gallardo, J.; Gutiérrez, F.L.; Cerezo, E.; Amengual, E.; Valera, R. Pervasive games: Giving a meaning based on the player experience. In Proceedings of the XVIII International Conference on Human Computer Interaction, Cancun, Mexico, 25–27 September 2017.
24. Saeki, M.; Kaiya, H. On relationships among models, meta models and ontologies. In Proceedings of the 6th OOPSLA Workshop on Domain-Specific Modeling (DSM 2006), Jyväskylä, Finland, 22 October 2006.
25. Gallardo, J.; Molina, A.I.; Bravo, C.; Redondo, M.A.; Collazos, C.A. An ontological conceptualization approach for awareness in domain-independent collaborative modeling systems: Application to a model-driven development method. *Expert Syst. Appl.* **2011**, *38*, 1099–1118. [CrossRef]
26. Guarino, N. Formal ontology in information systems. In Proceedings of the First International Conference (FOIS'98), Trento, Italy, 6–8 June 1998; Volume 46.
27. Pidcock, W. What are the Differences between a Vocabulary, a Taxonomy, a Thesaurus, an Ontology, and a Meta-Model? 2003. Available online: http://www.metamodel.com (accessed on 1 May 2015).
28. Guizzardi, G. On ontology, ontologies, conceptualizations, modeling languages, and (meta) models. *Front. Artif. Intell. Appl.* **2007**, *155*, 18.
29. Niles, I.; Pease, A. Towards a standard upper ontology. In Proceedings of the International Conference on Formal Ontology in Information Systems-Volume 2001, Ogunquit, ME, USA, 17–19 October 2001.
30. Sowa, J.F. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*; Brooks/Cole: Pacific Grove, CA, USA, 1999.
31. Aßmann, U.; Zschaler, S.; Wagner, G. Ontologies, meta-models, and the model-driven paradigm. In *Ontologies for Software Engineering and Software Technology*; Springer: Berlin, Germany, 2006; pp. 249–273.
32. Denny, M. Ontology Building: A Survey of Editing Tools. 2002. Available online: www.XML.com (accessed on 1 May 2015).

33. Cranefield, S.; Purvis, M. *UML as an Ontology Modelling Language*; University of Otago: Dunedin, New Zealand, 1999.
34. Kogut, P.; Cranefield, S.; Hart, L.; Dutra, M. UML for ontology development. *Knowl. Eng. Rev.* **2002**, *17*, 61–64. [CrossRef]
35. Booch, G. *The Unified Modeling Language User Guide, 2/E. 2005: Pearson Education India*; Addison Wesley: Boston, MA, USA, 2005.
36. McGuinness, D.L.; Van Harmelen, F. OWL web ontology language overview. *W3C Recomm.* **2004**, *10*, 2004.
37. Krogstie, J. *Model-Based Development and Evolution of Information Systems: A Quality Approach*; Springer: Berlin, Germany, 2012.
38. Rouse, R., III. *Game Design: Theory and Practice*; Jones & Bartlett Learning: Burlington, MA, USA, 2010.
39. Facebook Inc. Facebook. 2013. Available online: http://www.facebook.com/ (accessed on 1 May 2015).
40. Twitter Inc. Twitter. 2013. Available online: https://twitter.com/ (accessed on 1 May 2015).
41. Feather, M.S.; Fickas, S.; Finkelstein, A.; van Lamsweerde, A. Requirements and specification exemplars. *Autom. Softw. Eng.* **1997**, *4*, 419–438. [CrossRef]
42. Furtado, A.; Santos, A.; Ramalho, G. *Streamlining Domain Analysis for Digital Games Product Lines Software Product Lines: Going Beyond*; Bosch, J., Lee, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 316–330.
43. Furtado, A.; Santos, A. Defining and Using Ontologies as Input for Game Software Factories. In Proceedings of the 3rd Brazilian Symposium on Computer Games and Digital Entertainment, Perth, Australia, 4–6 December 2006.
44. Furtado, A.W.B.; Santos, A.L.; Ramalho, G.L.; de Almeida, E.S. Improving Digital Game Development with Software Product Lines. *IEEE Softw.* **2011**, *28*, 30–37. [CrossRef]
45. Furtado, A.W.; Santos, A.L. Using domain-specific modeling towards computer games development industrialization. In Proceedings of the 6th OOPSLA Workshop on Domain-Specific Modeling (DSM06), Jyväskylä, Finland, 22 October 2006.
46. Tang, S.; et al. Towards a Domain Specific Modelling Language for Serious Game Design. In Proceedings of the 6th International Game Design and Technology Workshop (GDTW'08), Liverpool, UK, 12–13 November 2008.
47. Tang, S.; Hanneghan, M. A Model-Driven Framework to Support Development of Serious Games for Game-based Learning. In Proceedings of the Developments in E-systems Engineering (DESE), London, UK, 6–8 September 2010.
48. Tang, S.; Hanneghan, M. Fusing games technology and pedagogy for games-based learning through a model driven approach. In Proceedings of the Colloquium on the Humanities, Science and Engineering (CHUSER), Penang, Malaysia, 5–6 December 2011.
49. Reyno, E.M.; Cubel, J.Á.C. A Platform-Independent Model for Videogame Gameplay Specification. In Proceedings of the Digital Games Research Association Conference on Breaking New Ground: Innovation in Games, Play, Practice and Theory, London, UK, 1–4 September 2009.
50. Reyno, E.M.; Carsí Cubel, J.Á. Automatic prototyping in model-driven game development. *Comput. Entertain.* **2009**, *7*, 1–9. [CrossRef]
51. Maier, S.; Volk, D. Facilitating language-oriented game development by the help of language workbenches. In Proceedings of the 2008 Conference on Future Play: Research, Play, Share, Toronto, ON, Canada, 3–5 November 2008; pp. 224–227.
52. Funk, M.; Rauterberg, M. PULP scription: A DSL for mobile HTML5 game applications. In *Entertainment Computing-ICEC 2012*; Springer: Berlin, Germany, 2012; pp. 504–510.
53. Zhu, M.; Wang, A.I. Engine-Cooperative Game Modeling (ECGM): Bridge Model-Driven Game Development and Game Engine Tool-chains. In Proceedings of the International Conference on Advances in Computer Entertainment Technology, Osaka, Japan, 9–12 November 2016.
54. Núñez-Valdez, E.R.; García-Díaz, V.; Lovelle, J.M.C.; Achaerandio, Y.S.; González-Crespo, R. A model-driven approach to generate and deploy videogames on multiple platforms. *J. Ambient Intell. Hum. Comput.* **2017**, *8*, 435–447. [CrossRef]
55. Nelson, M.; Mateas, M. Towards automated game design. *Lect. Notes Comput. Sci.* **2007**, *4733*, 626.
56. Lindley, C. Game Taxonomies: A High Level Framework for Game Analysis and Design. 2003. Available online: www.gamasutra.com/features/20031003/lindley_01.Shtml (accessed on 1 May 2015).

57. Guo, H.; Gao, S.; Krogstie, J.; Trætteberg, H.; Wang, A.I. An evaluation of ontology based domain analysis for model driven development. *Int. J. Semant. Web Inf. Syst.* **2015**, *11*, 41–63. [CrossRef]
58. Davis, F.D. Perceived usefulness, perceived ease of use and user acceptance of information technology. *MIS Q.* **1989**, *13*, 319–340. [CrossRef]
59. Guo, H.; Trætteberg, H.; Wang, A.I.; Gao, S.; Jaccheri, M.L. RealCoins: A Case Study of Enhanced Model Driven Development for Pervasive Game. *Int. J. Multimed. Ubiquit. Eng.* **2015**, *10*, 395–411.
60. Padilla-Zea, N.; Gutiérrez, F.L.; López-Arcos, J.R.; Abad-Arranz, A.; Paderewski, P. Modeling storytelling to be used in educational video games. *Comput. Hum. Behav.* **2014**, *31*, 461–474. [CrossRef]