Agent Oriented Software Engineering for Multimedia Systems' Development
An Experimental Case Study

Alma M. Gómez-Rodríguez, Juan Carlos González-Moreno, David Ramos-Valcárcel and Francisco Javier Rodriguez-Martinez

Dept. de Informática, University of Vigo, Ed. Politécnico D-401, Campus As Lagoas, Ourense E-32004, Spain

Keywords: Multimedia Systems, Development Process, Agent Oriented Software Engineering(AOSE), INGENIAS Methodology, Development Tools.

Abstract: Multimedia systems, and in particular games, have special characteristics that make the process of obtaining system requirements very difficult. In most cases, an important work of codification is needed in order to obtain a prototype that can be analyzed by the customer and make requirements elicitation easier. This paper proposes a way of facilitating the process and obtaining a prototype in a short development time. Our proposal is based in the combined use of a well-known methodology for the construction of Multimedia Systems and Alice, and follows an Agent Oriented Software Engineering approach. The work introduces the process proposed as well as a tool that provides support for such process. Besides, a case study shows the application of the process and the tool to a simple example. From this example, the suitability of Agent Oriented Software Engineering approach and the proposed process for game prototype development is concluded.

1 INTRODUCTION

Nowadays, multimedia systems have a growing relevance among all the computer-based systems. Almost any device, from computers to smartphones or tablets, introduces multimedia components in the applications that they run (Songers et al., 2001). Besides, Multimedia developments are complex because they imply a multidisciplinary team, which must include: physicist because the system incorporates capabilities such as GPS, gyroscope ..., mathematicians, graphic designers, musicians, scriptwriters, and, obviously, software engineers (Ramos-Valcárcel et al., 2011).

Among the multimedia systems, games constitute the most complex system to construct. The reason is that they incorporate all the multimedia characteristics, as well as, other aspects, such as: defining the character's behavior and describing the interactions (among game characters or between them and the environment) (Masuch and Rueger, 2005; Capra et al., 2005; Moreno-Ger et al., 2009).

Despite the importance of multimedia developments, there is not a public methodology or process devoted to multimedia development. Although there are multimedia private companies with productive development processes, they are not public because of the competitive advantage provided.

The special characteristics of multimedia systems (in particular games, which have characters with a behavior and sometimes a certain degree of *intelligence*) make them suitable for applying the agent concept. Agent Oriented Software Engineering (AOSE) has proof its capability for definition of complex distributed systems, where different agents cooperate to achieve certain goals. Moreover, a previous work (Ramos-Valcárcel, 2011) has shown the correspondence between AOSE concepts and game concepts; for instance agent concept relates to game character, agent tasks to object functionalities, etc.

In AOSE, many methodologies have been proposed for systems development (Pavón and Gómez-Sanz, 2003; Cuesta et al., 2002; Cossentino and Sabatucci, 2004; Sturm et al., 2003). Among them, INGENIAS methodology has been selected attending two basic reasons. The first reason is that the different models and elements provided by the methodology were very suitable for multimedia modeling, as it has been introduced in previous works (Fuentes-Fernández et al., 2010; Gómez-Rodríguez et al., 2011). The second reason is that the methodology provides a tool that supports the development, called INGENIAS Development Kit (IDK) (Pavón et al.,

M. Gómez-Rodríguez A., Carlos González-Moreno J., Ramos-Varcárcel D. and Javier Rodriguez-Martinez F..
Agent Oriented Software Engineering for Multimedia Systems' Development - An Experimental Case Study.
DOI: 10.5220/0003998700850094
In *Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering* (ENASE-2012), pages 85-94
ISBN: 978-989-8565-13-6
Copyright © 2012 SCITEPRESS (Science and Technology Publications, Lda.)

2005). This tool generates automatically code from the models defined using it, so it can be used as an initial version of the system to be developed.

Nevertheless, games are very intensive in animation, so a tool that show how actors behave is needed. Moreover, the producer of a game expects to see a storyboard explaining the general idea of the game in order to take the decision of producing it. Nowadays there are works that automatically generate an animated storyboard from a system description that uses Natural Language (Ma, 2006; Kevin Glass, 2008; Bolaño-Rodríguez et al., 2011).

Our approach is different, since it generates the animated storyboard from software engineering models. From these models, a particular tool, the Interactive StoryBoard Generator (ISBGen), obtains the system description using XML. This XML documents are used as inputs for Alice (Group, 1995; Cooper et al., 2000), that visualizes these descriptions and provides them with 3D animation.

Finally, the utility of our approach is proved by using the development process proposed and ISBGen in a simple game development.

The remainder of the paper is organized as follows. Next section introduces the technologies used in this work, as well as, the relationships among them. Then, section 3 describes the ISBGen tool, which automatically generates the multimedia prototype. Section 4 broaches the development process for a system constructed following an AOSE approach and using the INGENIAS meta-models and the proposed tools. In the following section, this process is applied to a case study that shows the results obtained. The paper ends with the conclusions and future work.

2 TECHNOLOGIES IMPLIED

2.1 INGENIAS and IDK

INGENIAS (Pavón et al., 2005) is a methodology created for the development of *Multi-Agent Systems* (MAS). Originally the purpose of INGENIAS was the definition of a specific methodology for MAS development by integrating results from research in the area of agent technology, and from traditional Software Engineering Methodologies. The methodology covers analysis and design of MAS, and it is intended for general use, with no restrictions on application domain (Gómez-Sanz and Fuentes, 2002). INGENIAS is based on the definition of a set of meta-models which describe the elements necessary to specify and get a MAS from five viewpoints: *agent* (definition, control and management of agent mental state), *interactions*, *organization*, *environment*, and *goals/tasks*. The meta-models proposed cover the whole life cycle and capture different views of the system. In the latest years, these meta-models have demonstrated their capability and maturity as supporting specification for the development of MAS (Pavón and Gómez-Sanz, 2003).

The methodology can define either hardware or software agents, and has been applied in the creation of systems in several engineering fields, such as holonic manufacturing systems (Botti and Giret, 2008), multisensor surveillance systems (Pavón et al., 2007), knowledge management systems (Soto et al., 2006), business workflow definition (Hidalgo et al., 2007), simple games (Pavón and Gómez-Sanz, 2003) and Product-Line Engineering environments (Cuesta et al., 2007).

From the very beginning, INGENIAS methodology provides a supporting CASE tool called IDK. This tool facilitates the construction of the different meta-models defined in the methodology and generates code from those definitions, or at least, code templates.

Recently the INGENIAS Agent Framework (IAF) has been proposed taking into account the experience in application of the methodology during several years enabling a full model driven development. This means that, following the guidelines of the IAF, an experienced developer can focus most of his/her effort in specifying the system, and so, a great part of the implementation is converted in a matter of transforming automatically the specification into code.

2.2 Alice

Alice (Group, 1995; Conway et al., 2000; Cooper et al., 2000) is a programming environment which facilitates the creation of animations where a virtual world is populated with 3D objects (e.g., people, animals, vehicles ...). The tool is available freely and allows to create animated movies and simple video games. The tool has an interactive interface, where the user can drag and drop graphic tiles to create a program and allows to immediately see how the animation programs run.

Alice has a huge library of predefined 3D objects. It includes static and dynamic objects which have a graphical aspect and, for dynamic ones, a predefined behavior. This characteristic makes it very suitable for making a prototype, because, in some cases, the developer only has to choose what object to include in the animation.

Other important advantage of Alice is that it fol-

Table 1: Correspondence methodological and Alice concepts.

INGENIAS	ALICE
Agent	Object
Roles	Function activation/deactivation
Goals	Problems to solve
Task	Methods
Interaction	Functions
Use Cases	Scene Definition

lows an educative approach. This means that the tool is user-friendly, what makes easier introducing changes in the animation. This characteristic is very interesting in this case, because it facilitates changes in the prototype and the feedback to INGENIAS models.

The tool uses Java and C# in its implementation and stores the definitions using XML documents. As stated before, the data structures of both INGENIAS and Alice are based on XML, so that the integration of them is easier.

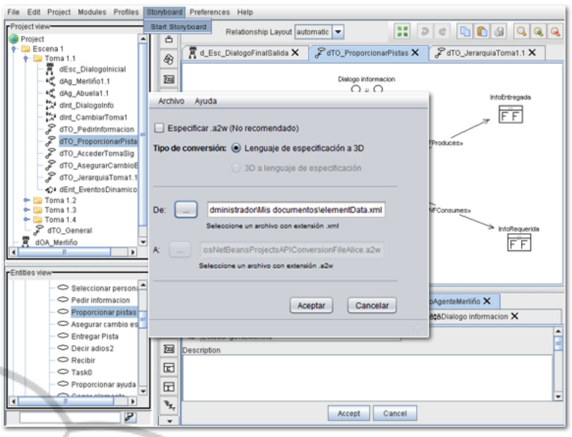
Moreover, a direct correspondence between the concepts used in the methodology and in Alice has been established in (Ramos-Valcárcel, 2011). These conceptual relationships constitute the formal basis for automatic generation of the storyboard. Table 1 reflects such correspondences.

3 ISBGen

This paper presents the tool ISBGen, which provides a 3D interactive storyboard automatic generation from the elements of the meta-models defined using IDK. ISBGen is one of the contributions of this work and constitutes an important tool to be used in the process of multimedia storyboard generation, as it will be shown in next section.

Figure 1 shows IDK with the ISBGen option. The central opened window shows the dialogue that appears when generating the storyboard from the meta-model definitions.

The tool facilitates the automatic production of the Alice file (with extension .a2w) that contains the storyboard in the suitable format. ISBGen translates the meta-models to an XML file that will be used as input in Alice. The use of XML provides a great versatility for adapting the tool, if new characteristics appear in the models or the tools (IDK and Alice).



[ ] Especificar .a2w (No recomendado)

Tipo de conversión: [✓ Lenguaje de especificación a 3D] 3D a lenguaje de especificación

De: dministrador\Mis documentos\elementData.xml
Seleccione un archivo con extensión .xml

A: osNetBeansProjectsAPIConversionFileAlice.a2w
Seleccione un archivo con extensión .a2w

Figure 1: ISBGen integrated in IDK.

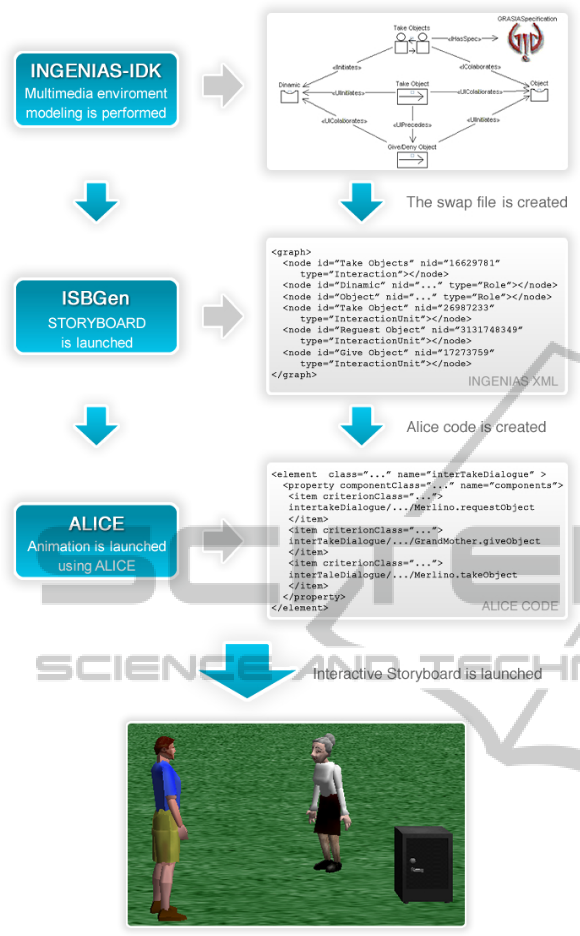
4 PROCESS OF DEVELOPMENT: FROM THE META-MODELS TO THE PROTOTYPE

Traditional game developments usually share two phases: pre-production and production. The first one includes the preliminary works oriented to define the basic characteristics of the game and to propose solutions to the most relevant functionalities. The phase of production will obtain the final product, developing in detail the sequences defined previously.

Habitually, in pre-production phase, the scripwriters explain the basic idea using a storyboard which is evaluated by the responsibles of the project. The specifications are given after studying the script or storyboard that graphic designers make. From both resources, script and storyboard, a first prototype of the product can be modeled and an animated sequence is generated . This provides a good simulation of the final sequences to be developed and constitutes a natural way for the customer (producer, director, etc.) of verifying the model built.

Our development process covers the pre-production phase and has as main aim the automatic generation of the storyboard. This process is based on the use of AOSE techniques adapted to multimedia systems, and on the utilization of ISBGen that produces fast simulation of the multimedia project at animation level (Storyboard Interactive). At a high level of abstraction, the process consists of the cyclic application of the following steps:

1. The elements (object, tasks, goals, etc.) which constitute the system are defined using INGENIAS meta-models.
2. Using Alice, the graphical part of the elements



```
<graph>
  <node id="Take Objects" nid="16629781"
     type="Interaction"></node>
  <node id="Dinamic" nid="..." type="Role"></node>
  <node id="Object" nid="..." type="Role"></node>
  <node id="Take Object" nid="26987233"
     type="InteractionUnit"></node>
  <node id="Request Object" nid="3131748349"
     type="InteractionUnit"></node>
  <node id="Give Object" nid="17273759"
     type="InteractionUnit"></node>
</graph>
```

```
<element  class="..." name="interTakeDialogue" >
  <property componentClass="..." name="components">
   <item criterionClass="...">
   intertakeDialogue/.../Merlino.requestObject
   </item>
   <item criterionClass="...">
   interTakeDialogue/.../GrandMother.giveObject
   </item>
   <item criterionClass="...">
   interTaleDialogue/.../Merlino.takeObject
   </item>
  </property>
</element>
```

Figure 2: Prototype generation process.

(characters and objects in the scenery) is included. This object can be taken directly from Alice library or adapted from the available ones.

3. The ISBGen tool is used to generate in an automatic way the prototype.

4. The prototype is revised with the customer, the improvements are noted down and the process starts again from the first step. The changes only imply to redefine the meta-models.

Figure 2 shows a single iteration of the process previously described.

Starting from a model constructed using IDK, an animated sequence, which constitutes a rapid interactive prototype that can be evaluated by the customer, is obtained. Also, the pre-production team can analyze the results, corroborate the implementation work that has been carried out and repair the deficiencies or optimize the features and animations.

Using this process, it is possible to make available for the customer the idea that the scriptwriter has in mind, but starring with supporting actors (whose cost is significantly lower than protagonists). The prototype developed is a first test, that facilitates the study of all possible technical and graphics solutions to be applied in the final sequences. Moreover, the prototype permits to identify not obvious features and contributes to detect errors and problems that may appear in the future implementation.

The big advantage of this process is that after this first model done in the analysis phase, a sketch of the system and the possible animated sequences that the script creates can be presented to the customer. Moreover, the feedback provided by him can be incorporated directly to the meta-models to improve the scene.

As it has been said before, the process is quick, because, the graphical part of most objects is obtained from Alice library. If it is the case that there is not a suitable object in Alice for a particular element, the graphical part must be modeled in that moment. Nevertheless, the Alice objects can be adapted, if needed.

The process obtains as output a static model, which locates the elements inside the scene and defines what parameters should be modeled. This means establishing a scenery (a location) and natural components of the environment by default. These choices should reflect aspects such as the natural elements that interact with the environment: trees, plants, rivers,wind, rain . . . , or the lighting and its different types (radial, environment, parallel, etc.).

These elements are considered part of the environment and, as happens in virtual worlds like Second Life (Kumar et al., 2008; Rymaszewski, 2007) and OpenSim (Childers, 2009; Fishwick, 2009), should have their own predefined attributes. Sometimes, the user should be able to act on such elements, but, in other moments, it is much more interesting that the elements have a random behavior, so that, the user can understand how they interact with the new world (like in some sites of virtual worlds).

The main aim of this part of the process is to fix the meta-model elements that are necessary to represent a multimedia scene and the other elements that are present on the scene but have not a relevant behavior. The agent paradigm provides an enormous potential to model these elements with undefined behavior, that perform the role of extras. The use of intelligent agents allows defining their behavior independently. Once this behavior has been defined, new not anticipated situations may appear from the interaction among agents, and these situations can be identified before interfering with the relevant ones.

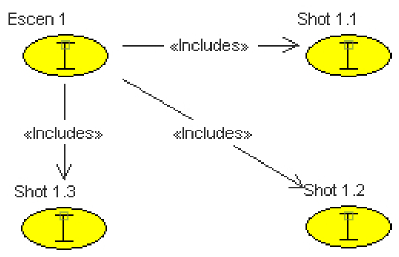


Figure 3: Use case model.

5 CASE STUDY

The system to be modeled as a basic example is a dialogue between two characters (named *Merlin* and *Grandmother*) who discuss how to give each other a particular object. The following section explains in detail and step by step the results that can be achieved using the process previously described and the introduced tools (Alice, IDK and ISBGen).

5.1 Structure and Use Cases Models

The initial step of this process will be to define the **project structure**, that is, the number of scenes/shots that compose the whole system. For the case study, the dialog is composed of a single scene that, in turn, is divided in several shots (in the example of Figure 3 three shots are shown).

Next, a **use case diagram** must be modeled. Although this kind of diagram is not used in INGENIAS, we consider that it is of great utility in this moment of development. The diagram introduces the scenes which form part of the project and verifies the integrity of the structure previously defined. Figure 3 presents the use case diagram for the case study. The diagram shows a system with a scene consisting on three shots, but the example we are dealing with will be focused only in the first shot.

From this diagram, the ISBGen tool creates in Alice the objects defined in the diagram. Each use case corresponds to a shot and the tool introduces a number when generating the Alice object. These numbers indicate the order in which the use cases are going to be played. For instance, the model contained in Figure 3 will generate two objects in Alice: *shot1* and *shot11*. *Shot1* refers to ”Scene 1”, while *shot11* corresponds to the ”shot1.1” of INGENIAS model that is part of ”Scene1”' (see the image in the central bottom part of Figure 4).

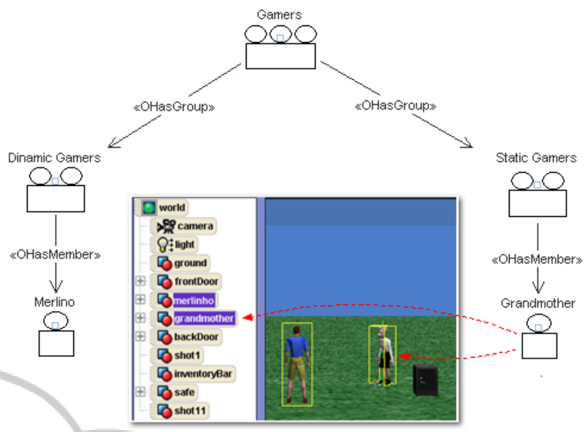


Figure 4: General organization model.

5.2 Organization Model

Following with the definition of the prototype, next step is defining the **Organization Model**, which describes the groups of agents, the functionality and the restrictions imposed on the agents. Before starting this diagram it is important to know the actors who interact with the system, their roles and, if the actors play different roles, how this roles will be grouped.

In Figure 4, the Organization Model of the case study is presented. The system consists of two groups of agents: *static* and *dynamic* agent. The agents belonging to the dynamic group are characterized by their capability to attend user interactions, while those of the static group do not require user intervention, so that they are completely autonomous, and act following their established goals.

This model has a direct correspondence with the 3D world modeled in INGENIAS and automatically generated by the tool, which can be seen in bottom part of Figure 4. ISBGen will generate, both the *Merlin* and the *Grandmother* agents from the diagram to Alice code. So, adding another agent to the virtual world would be as simple as including him in the organizational model and link him to a group of agents.

Following the process previously defined, the next step is to define another **Organization Diagram**. In this case, the diagram details each of the shots associated to a particular scene. For instance, in the diagram of Figure 5, the interaction between agents that play the roles *dynamic* and *help* that fulfil the tasks *Request Information* and *Provide Clues*.

From this model, the communication between the two actors that play the roles *dynamic* and *help* can be extracted. The dynamic role selects the character to which it makes the request for information and, then the other role, through the establishment of a dialogue, provides the requested information. This dia-

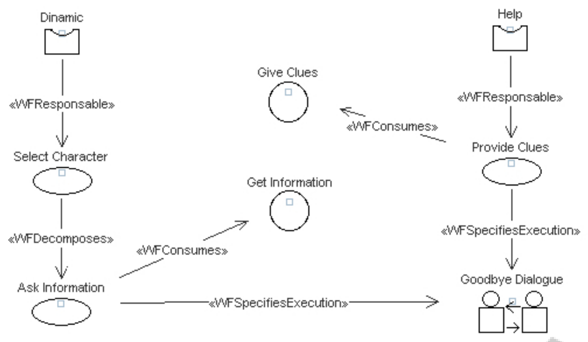


Figure 5: Organization model for Shot 1.1.

logue makes that both roles achieve their established goals. The elements of this diagram are used to determine which tasks are involved in the interactions that occur during the execution of the system.

ISBGen does not use this diagram for code generation, sice since the tasks definition is performed later, and the information provided by this diagram is just illustrative for the developer. For this reason, this step can be skipped and it is only recommended to define these diagrams in large systems where this extra information can be useful.

5.3 Tasks and Goals Model

Next phase consists in specifying the order in which the goals must be completed by agents. It is mandatory to take such decision before starting the execution of the system, and it is made through the definition of **Tasks and Goals Model**. The Tasks and Goals Model represents the relationships between goals, so that the order in which they must be achieved can be deduced. Moreover, the model may be used to break down the goals in the particular tasks that must be done to fulfil them. In the case study, the Tasks and Goals model has a sole goal: *to dialogue*, so that, the diagram is not showed here.

Task and Goals Model is very relevant in this process because, the hierarchical structure of the system is completed using the information that it contains. The methods and functions responsible for managing the lists of basic and secondary goals are automatically built by ISBGen.

From all the properties available in Alice, we have choose the most suitable ones for defining agent characteristics. In particular, figure 6 reflects in Alice the list of goals, methods, functions and properties associated with the case study. The properties panel (Properties) of the object *world* (the root of the project) -top left image- is an ordered list of the main goals of the system called *GoalHierarchy*. It also contains the objects that reference the current, the next

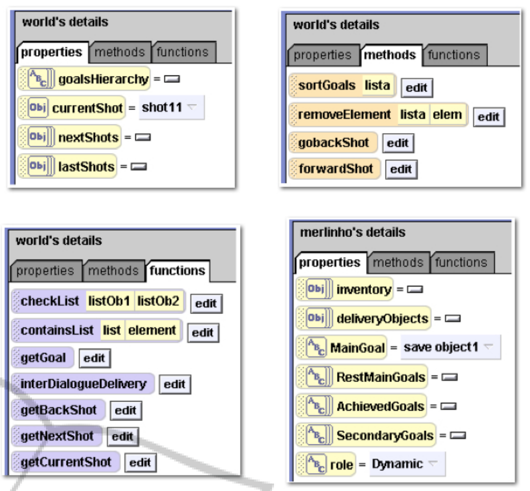


Figure 6: Correspondences Alice - Task and goal model.

and the previous shot. In the methods panel (Methods) -upper right image- the methods for operating on the goals list and for modifying the current shot are shown. The bottom left panel is functions and contains the methods needed to perform operations on agents, objects, shots and goals. Finally, the bottom right panel displays the properties of the object *principal*, that is,the main attributes of the main character, the list of objects and the list of goals, among others.

5.4 Agent Model

Next diagrams to be constructed are based on the **Agent Model**. An Agent Model must be done for each agent in the system. The model will show the capabilities and goals of each of the actors. The capabilities are represented by tasks, and each task must have an associated goal that must be achieved after execution. Summarizing, the agent model allows a detailed description of existing agents in the multimedia system.

An example of this kind of model is shown in Figure 7. From the definition of Agent Models, ISBGen generate the data related to the agent, its tasks and goals, the methods, and the list of goals to be met, which will be used in Alice. The figure reflects all the previous attributes for the main character of the case study. Each new feature of the 3D character of the system must be specified in the corresponding Agent Model as a task and, also, the goal accomplished by the task must be specified. For instance, to make an actor capable to move, in the Agent Model an agent with the tasks *move* and the goal *walk* must be defined. Once the information is used by ISBGen for generating the character, this will be reflected in the 3D world as a new method of the character of Alice,

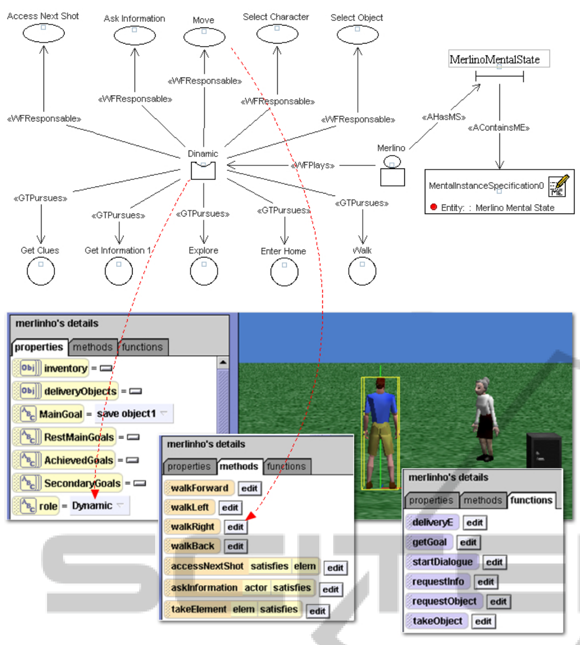


Figure 7: Agent model.

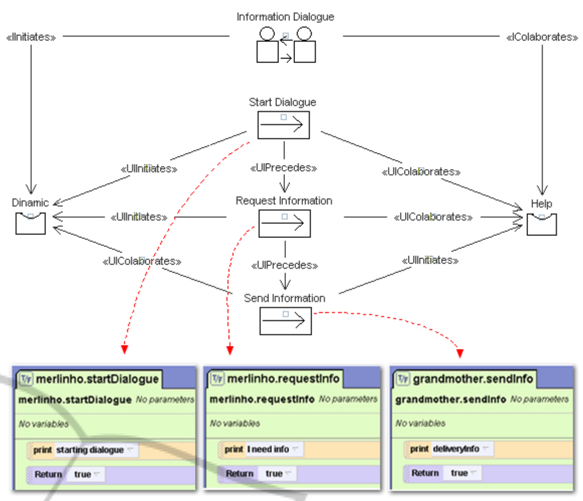


Figure 8: Interaction model.

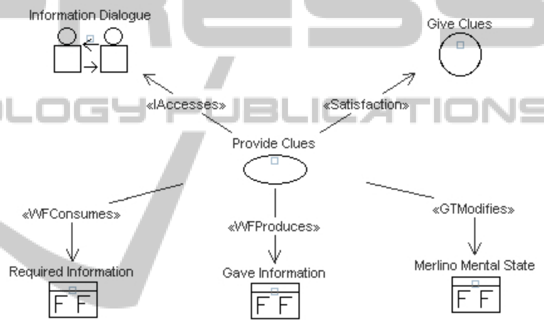


Figure 9: Task and goal model.

symbolizing the actor.

The bottom part of Figure 7 contain the properties, the methods and the functions for the main character of the case study, in particular the one called *Merlinho*. The properties panel represented contains the list of objects in the system, the ordered lists of goals and the role played by the agent selected. The second image -lower left- panel shows the object's methods, that is, the basic methods of the dynamic character, the movements and all the functionalities specified in the Tasks and Goal Model for the agents and tasks. The last image, bottom right, shows the functions panel, where the exchanges between the elements and the environment are shown. These functions represent the units of interaction described in the Interaction Model.

5.5 Interaction Model

To complete the example of case study, it is necessary to use the Interaction Model. All the information exchanges that occur between two or more actors in the system should be reflected as interactions. This diagram represents the information exchanges, breaks down the actions of each interaction and specifies what is sent, and who is the sender and the receiver.

Figure 8 shows the model that defines the dialogue between the two agents, that is the core of the system. The bottom part of the figure introduces the ALICE definitions associated with the interaction units of the model. Each interaction unit appears in ALICE as a function associated with the agent that initiates the interaction.

5.6 Environment Model

The inclusion of events in the 3D world is done through the Environment Model, by identifying different internal applications that can be used and shared by the actors.

Finally, it will be necessary to describe in detail each of the tasks in the system using again Tasks and Goals Models. In these models the inputs and outputs that occur during the execution of a task and what are the initiators of such events should be described. The example in Figure 9 contains the description of the task *Provide clues* initiated by receiving certain information, called *RequiredInformation*. This task produces other task *GaveInformation*, which is used in the interaction, to complete the objective *Provide clues*. In addition, the task changes the mental status of the agent responsible for the task.

After modeling the environment, as stated before, an initial review with the customer is carried out. This approach, though simple, allows to discuss with the customer this first prototype of the final product.

Any change to this prototype, represents a very small cost because the system is still in the very early stages of development. So that, this does not signify a cost beyond the predictions and is part of the process of product development. The customer can immediately see this first draft and confirm or change the specification of the generated prototype . Thus, the interaction developer/customer becomes really efficient, resulting in rapid prototypes that conform to the pattern expected. Attending the customer changes, a process redesign, involving product improvement through optimization of the results of the prototype, is followed. With all this, a rapid definition of the expected product, which may be brought to the stage of implementation without risks, is obtained.

6 CONCLUSIONS

In the first steps of a multimedia project a high-level prototype makes easier the task of obtaining and validating the basic system's functionalities. The prototype makes available to the user an overview of the multimedia product, and facilitates the feedback. This increases the efficiency in the pre-production process, because all the changes needed are made at initial stages of development: unnecessary sequences can be eliminated or others are discovered. Moreover, the obtained models have more meaningful elements whose implementation is partially automated thanks to the tools offered. The use of ISBGen to obtain in an automatic way the storyboard from those definitions substantially reduces the time and costs of pre-production.

The use of a well-defined process supported by several tools introduces a certain degree of formalization in multimedia development, which usually lacks a formal process of software specification. In addition, the use of agent paradigm is very valuable, as the concept of role, associated to a particular agent behavior, enhances the creation of intelligent autonomous behaviors that can discover new situations, not defined a priori, that may occur in the scenes. The use of ALICE, the modeling elements provided by the tool, as well as their by default behavior, makes it much easier to find objects that adapt to a particular situation to model.

Using the case study presented, it can be concluded that modeling a multimedia system is feasible using Agent Oriented Software Engineering. The use of INGENIAS methodology is not specially relevant, and in the future, other methodologies will be used. We consider this is feasible, because the concepts and elements between different methodologies coincide, and the process proposed can be applied in the same way.

The case study has identified some limitations of the methodology for being used in this kind of developments. For instance, INGENIAS introduces many modeling elements that are not needed for multimedia definitions (like the sequence Model or the BDI mental state). Due to this, in the future, we plan to pose a new methodological proposal that overcomes such limitations. In addition, a new IDK profile for multimedia systems will be defined, which incorporates the proposed changes .

Despite the satisfactory results of ISBGen tool, several improvements are intended to be made in the future, in order to extend its functionalities. In particular, an interesting issue will be to incorporate the capability of selecting the graphical aspect of the agents. At the moment, the tool automatically assigns an Alice character for each of the agents in the system. In the future, the user will be able to choose for each agent its character from the ones available in Alice library.

The process proposed in this paper follows a top-down approach. Nevertheless, once the storyboard is obtained and presented to customer, it suffers changes. Offering a mechanism to automatically incorporate those changes to meta-models definitions, providing some kind of reengineering process, is, at the moment, under study.

Finally, we have the intention of integrate the NPL4INGENIAS tool (Gonzalez-Moreno and Vazquez-Lopez, 2008; Gómez-Rodríguez et al., 2011) in the process of development. NPL4INGENIAS is a tool that obtains the system requirements form its description in Natural Language. The requirements obtained are documented using the INGENIAS meta-models, this implies that the tool can be easily integrated in the process defined here and can simplify the obtention of a first model of the system to construct. A first experiment, which suggest this is possible has been done in (Bolaño-Rodríguez et al., 2011).

ACKNOWLEDGEMENTS

This work has been supported by the project *Plataforma Tecnoloxica Intelixente de Xestin Avanzada para Produtores de Planta Ornamental de Galicia* with grant 10MRU007E of Xunta de Galicia.

REFERENCES

Bolaño-Rodríguez, E., Moreno, J. C. G., Ramos-Valcárcel, D., and López, L. V. (2011). Using multi-agent systems to visualize text descriptions. In *PAAMS*, pages 39–45.

Botti, V. and Giret, A. (2008). *ANEMONA: A Multi-agent Methodology for Holonic Manufacturing Systems*. Springer Series in Advanced Manufacturing.

Capra, M., Radenkovic, M., Benford, S., Oppermann, L., Drozd, A., and Flintham, M. (2005). The multimedia challenges raised by pervasive games. In *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA '05, pages 89–95, New York, NY, USA. ACM.

Childers, B. (2009). Run your own virtual reality with opensim. *Linux J.*, 2009.

Conway, M., Audia, S., Burnette, T., Cosgrove, D., Christiansen, K., Deline, R., Durbin, J., Gossweiler, R., Koga, S., Long, C., Mallory, B., Miale, S., Monkaitis, K., Patten, J., Pierce, J., Shochet, J., Staack, D., Stearns, B., Stoakley, R., Sturgill, C., Viega, J., White, J., Williams, G., and Pausch, R. (2000). Alice: Lessons learned from building a 3d system for novices.

Cooper, S., Dann, W., and Pausch, R. (2000). Alice: a 3-d tool for introductory programming concepts. *Journal of Computing Sciences in Colleges*, 15(5):107–116.

Cossentino, M. and Sabatucci, L. (2004). Agent System Implementation. *Agent-Based Manufacturing and Control Systems: New Agile Manufacturing Solutions for Achieving Peak Performance. CRC Press*.

Cuesta, P., Gómez, A., González, J., and Rodríguez, F. J. (2002). The MESMA methodology for agent-oriented software engineering. In *Proceedings of First International Workshop on Practical Applications of Agents and Multiagent Systems (IWPAAMS'2002)*, pages 87–98.

Cuesta, P., Gómez-Rodríguez, A., and González-Moreno, J. C. (2007). Agent oriented software engineering. In *Whitestein Series in Software Agent Technologies and Autonomic Computing*, pages 1–31. Springer.

Fishwick, P. (2009). An introduction to opensimulator and virtual environment agent-based applications. In *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pages 177 –183.

Fuentes-Fernández, R., no, I. G.-M., Gómez-Rodríguez, A. M., and González-Moreno, J. C. (2010). A technique fordefining agent-oriented engineering processes with tool support. *Engineering Applications of Artificial Intelligence*, '23(3):432– 444.

Gómez-Rodríguez, A., González-Moreno, J. C., Ramos-Valcárcel, D., and Vázquez-López, L. (2011). Modeling serious games using aose methodologies. In *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pages 53 –58.

Gómez-Sanz, J. J. and Fuentes, R. (2002). Agent oriented software engineering with ingenias. In *Fourth Iberoamerican Workshop on Multi-Agent Systems (Iberagents'2002), a workshop of IBERAMIA'2002, the VIII Iberoamerican Conference on Artificial Intelligence*.

Gonzalez-Moreno, J. C. and Vazquez-Lopez, L. (2008). Design of multiagent system architecture. In Ieee, editor, *IEEE International Workshop on Engineering Semantic Agent Systems*, pages 565–568, Turku (Finlandia). IEEE.

Group, U. U. I. (1995). Alice: Rapid prototyping for virtual reality. *IEEE Computer Graphics and Applications*, 15:8–11.

Hidalgo, A., Gomez-Sanz, J., and Mestras, J. (2007). Workflow Modelling with INGENIAS methodology. In *5th IEEE International Conference on Industrial Informatics (INDIN'07)*, volume 2, pages 1103–1108. Vienna, Austria.

Kevin Glass, S. B. (2008). Automating the creation of 3d animation from annotated fiction text. In *IADIS 2008 : Proceedings of the International Conference on Computer Graphics and Visualization 2008*, pages 3–10.

Kumar, S., Chhugani, J., Kim, C., Kim, D., Nguyen, A., Dubey, P., Bienia, C., and Kim, Y. (2008). Second life and the new generation of virtual worlds. *Computer*, 41(9):46 –53.

Ma, M. (2006). *Automatic conversion of natural language to 3D animation*. PhD thesis, University of Ulster, Derry, Ireland.

Masuch, M. and Rueger, M. (2005). Challenges in collaborative game design developing learning environments for creating games. In *Proceedings of the Third International Conference on Creating, Connecting and Collaborating through Computing*, pages 67–74, Washington, DC, USA. IEEE Computer Society.

Moreno-Ger, P., Fuentes-Fernández, R., Sierra-Rodríguez, J.-L., and Fernández-Manjón, B. (2009). Model-checking for adventure videogames. *Information and Software Technology*, 51(3):564 – 580.

Pavón, J. and Gómez-Sanz, J. (2003). Agent Oriented Software Engineering with INGENIAS. *Multi-Agent Systems and Applications III*, 2691:394–403.

Pavón, J., Gómez-Sanz, J. J., Fernández-Caballero, A., and Valencia-Jiménez, J. J. (2007). Development of intelligent multisensor surveillance systems with agents. *Robotics and Autonomous Systems*, 55(12):892–903.

Pavón, J., Gómez-Sanz, J. J., and Fuentes-Fernández, R. (2005). *The INGENIAS Methodology and Tools*, article IX, pages 236–276. Idea Group Publishing.

Ramos-Valcárcel, D. (2011). *Ingeniería de software orientada a agentes en el modelado de sistemas multimedia (in spanish)*. PhD thesis, Universidad de Vigo. Departamento de Informática. Escola Superior de Enxeñería Informática.

Ramos-Valcárcel, D., Fajardo-Toro, C., and de la Pena Ojea, F. (2011). Multimedia smart process (msp). In *Information Systems and Technologies (CISTI), 2011 6th Iberian Conference on*, volume 1, pages 320–325.

Rymaszewski, M. (2007). *Second life : the official guide*. John Wileyy.

Songers, A., Diekmann, J., and Karet, D. (2001). *Animation-based construction schedule review*. Construction Innovation: Information, Process, Management, Colorado.

Soto, J. P., Vizcaino, A., Portillo, J., and Piattini, M. (2006). Modelling a Knowledge Management System Architecture with INGENIAS Methodology. In *15th International Conference on Computing (CIC'06)*, pages 167–173. Mexico City, Mexico.

Sturm, A., Dori, D., and Shehory, O. (2003). Single-model method for specifying multi-agent systems. In *AAMAS*, pages 121–128. ACM.