Serious Games Scenario Modeling for Non-experts

Yohan Duval[1,3], David Panzoli[1,2], Axel Reymonet[3], Jean-Yves Plantec[1], Jérôme Thomas[3] and Jean-Pierre Jessel[1]

[1]*IRIT, University of Toulouse, Toulouse, France*

[2]*CUFR Jean-François Champollion, Albi, France*

[3]*ACTIA Automotive, Toulouse, France*

Keywords: Serious Games, Authoring, Scenario Modeling, Game-based Learning.

Abstract: The use of serious games and gamified softwares is a new and growing trend for training professionals in a wide variety of disciplines where procedures and decision-making are key (automotive diagnostic, surgery, etc). Serious Games are safer, less expensive and advocated to be more efficient. Unfortunately, there is a lack of methodology and tools adapted for non-computing experts to develop their own gamified learning scenarios. In this paper, we propose an approach allowing trainers to model professional activities in the form of serious games scenarios. Trainers are enabled to express their expertise using a domain specific graphical representation which will be implemented eventually in an easy-to-use authoring tool. The produced scenarios describe both the actions necessary for completing the professional procedure and the associated pedagogy to provide the learner with relevant educational feedback. The proposed approach specifies a model matching those requirements, and is illustrated by an application example in the automotive context. We intend to demonstrate that an appropriate model is likely to make scenario editing accessible to trainers who are not necessarily familiar with computer modeling in the first place.

1 INTRODUCTION

(Alvarez and Djaouti, 2011) defines a Serious Game as a *"computer application, for which the original intention is to combine with consistency, both serious aspects [...] with playful springs from the video game"*. Serious Games are becoming a new form of well-appreciated training tools, because of their safety, their low cost and their efficiency (Van Est et al., 2011). Although trainers would benefit introducing Serious Games in their training, developing such tools is too difficult for them, for obvious complexity reasons, but also, we believe, in lack of an adapted methodology.

Yet, involving the trainer in the process is essential for two reasons. Firstly, they have the knowledge to describe professional activities in which they are specialized, using their own natural language. Secondly, they are able to provide learners with relevant pedagogical feedback during the execution of these activities, such as the accomplishment of important educational objectives or the validation of acquired skills with Multiple Choice Questions (MCQ). However, trainers do not have the necessary computing expertise to transfer in a natural manner these two dimensions into a computer intelligible-formal format.

In this paper, we define a scenario as the description of the professional activities in terms of game interactions associated with the description of the pedagogical feedback. Our objective is to propose an approach that empowers the trainers with the ability to create serious game scenarios on their own by expressing their expertise as naturally as possible. Throughout the article, we illustrate our approach with a scenario from the Diag'Adventures project, set in the automotive context. Particularly, the game scenario we refer to describes the first steps of a car diagnostic operation, where activities are expected to be carried out on the car as well as on the diagnostic software. Figure 1 displays the virtual environment in which the game is set. In order to fulfill the steps of the procedure, the user can switch at any time between the garage view (a), where they can interact mainly with a virtual car, and the diagnostic software view (b). Whereas the garage is entirely simulated in a 3D virtual environment, it is interesting to note that the genuine diagnostic software is utilised. This entails a more immersive simulation of the process,

Duval Y., Panzoli D., Reymonet A., Plantec J., Thomas J. and Jessel J..
Serious Games Scenario Modeling for Non-experts.
DOI: 10.5220/0005489904740479
In *Proceedings of the 7th International Conference on Computer Supported Education* (CSEDU-2015), pages 474-479
ISBN: 978-989-758-107-6

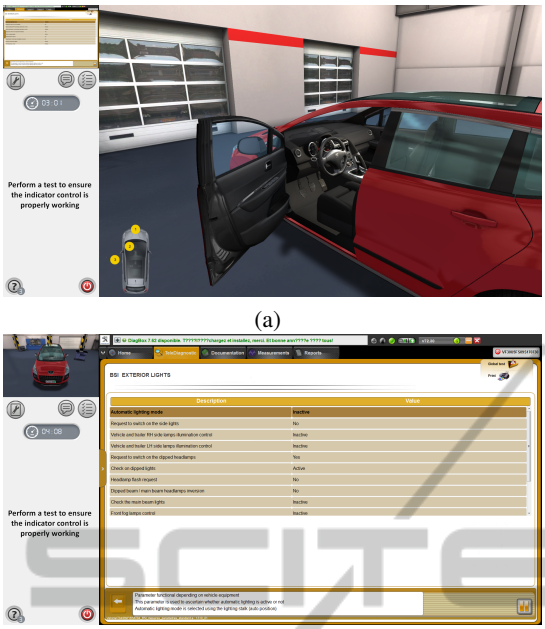

Figure 1: The game environment of Diag'Adventures features two views : (a) the interactive car in 3D and (b) the diagnostic software.

which should allow for a less steep learning curve and a more efficient transfer into the real world of the knowledge acquired virtually, since it eases the back and forth transitions between the game and the real-life activities. In the next section, we will briefly review the related work. Then, we will point out the coexistence of two separate dimensions within a scenario: the description of the activity and the pedagogy. This will lead us to propose a model, which we will detail and illustrate with the aforementioned scenario. We conclude that our approach empowers the trainers with the necessary tools to describe their professional and pedagogical expertise in an expressive yet formal fashion.

2 RELATED WORK

In the domain of virtual environment modeling, previous works have already been carried out to allow non-experts to perform this complex and multidisciplinary task. Existing textual modeling languages have been used to describe and orchestrate virtual environments. (Ishida, 2002) has extended the Scheme language to describe interaction scenarios between agents inside a 3D environment. The Q language thus created introduces in particular cues and actions, which refer respectively to observations and modifications in the environment. This allows an event triggering system which is expressive enough to be used by non-experts. In (Devillers and Donikian, 2003), authors have chosen to define their own grammar to design a textual language that allows the orchestration of agents in a 3D virtual world into multiple subscenarios. This language provides a syntax close to C++, and has the particularity of proposing instructions that can be represented as Finite State Machines. It is also possible to use Extensible Markup Language (XML) or JavaScript Object Notation (JSON) with specific schemas to describe different objects in a virtual environment (Tang et al., 2013) or to describe Web-Based Multimodal Interaction Scenarios (Katsurada et al., 2003). These languages have the advantage of being simpler to implement thanks to a large amount of dedicated tools. However, the direct use of these languages to describe activities in a virtual environment implies that the author of the scenario has computer-expert skills, and therefore this approach does not match our requirements.

Graphical notations can also be resorted to for the design of scenarios in a virtual environment. For instance, Unified Modeling Language (UML) class and state diagrams are used in (Tang and Hanneghan, 2008). (Buche et al., 2010) designs procedures in a virtual world using UML activity diagrams. (Panzoli et al., 2014) uses the Business Process Model and Notation (BPMN) language to model the activities and the interactions of several actors in a 3D virtual operating room. Game scenarios can also be described using Petri Nets (Araújo and Roque, 2009). They proved to be powerful mathematical analysis tools, but as such their usage is restricted to an expert population. These last three representations have the advantage of being easier to use by non-computing experts due to their graphical nature. However, they still lack expressiveness: BPMN does not support the definition of virtual world objects attributes (Tang and Hanneghan, 2008), Petri Nets have difficulties to model events in scenarios (Tang and Hanneghan, 2008), and UML is too complex since modeling every aspect of a scenario requires the use of many types of diagrams (Tang and Hanneghan, 2008)(Araújo and Roque, 2009). To overcome these problems, some researchers have used their own representation. One common way to do so is to use graphical blocks. (Van Est et al., 2011) uses high-level configurable building blocks to describe serious games scenarios. This representation allows non-experts to naturally express professional activities over time with actions and events; however, it still lacks support for the description of concurrent tasks and for a clear description of the associated pedagogy.

Puzzle blocks, used in Scratch (Resnick et al., 2009) and Alice (Kelleher and Pausch, 2006) for teaching programming to children, allow the description of various games scenario using a user-friendly representation. However, these blocks are too low-level for a non-computing expert to simply express high-level activities.

The graphical approach seems to be the most relevant choice for defining a higher-level framework and methodology within the reach of non-programming experts. Indeed, textual languages do not provide them with enough expressiveness to describe in a formal and simple representation both the professional activities and the pedagogy. Existing traditional and custom graphical notations do not provide every feature we need either. However, the principle of cues and actions (Ishida, 2002), also used by (Van Est et al., 2011) with events and actions, is interesting since it allows non-experts to naturally express the common distinction between observing and modifying the virtual environment. We are also interested in using configurable building blocks (Van Est et al., 2011) as the scenario description basis, combined with BPMN (White, 2004) (Panzoli et al., 2014) and activities diagram notations (Buche et al., 2010), since it gives users the possibility to model high-level tasks with traditional representations.

3 PROPOSED APPROACH

As we pointed out in the introduction, trainers have the knowledge to describe both professional activities and associated pedagogical feedback in a natural language. On the other hand, they lack the computing expertise to transfer this knowledge into a computer-intelligible format, to finally create scenarios fitting their need and directly interpretable by a Serious Game Engine. Most of the methods previously discussed describe activities in an actor-centered point of view. Our approach will propose a method that describes these activities from the virtual environment point of view. As a matter of fact, it allows trainers to focus on the expected result instead of what the player should do in order to reach this result, which is more natural and straightforward to them. Besides, we did not notice any custom graphical modeling notation (blocks modeling) that uses existing and expressive representations (UML, BPMN) to model activities. Therefore, we will endeavor to specify a precise graphical syntax allowing non-computing experts to naturally express learning scenarios in a virtual environment. Furthermore, none of the previous methods takes into account the description of pedagogical feedback during the player's activities. Regarding our approach, we want to give the trainer the possibility to add educational information in the scenario, so the learner can get direct feedback during their learning sessions. Two dimensions in the process of describing serious game scenarios distinctly arise. On the one hand, the professional activities need to be described as sequences of tasks. On the other hand, pedagogical objectives and feedback (advice messages, MCQ, etc.) need to be modeled and associated to the execution of the activities. These two dimensions are orthogonal as they cannot be naturally described using the same representation. However, they share a common entity: the activity block, which simply represents an elementary action in the virtual environment. Those two dimensions, along with their respective properties, are detailed in the next sections.

3.1 Scenario Modeling: Activities

The first dimension consists in describing how the learners activities should be modeled. The goal we seek to achieve is to allow a non-computing expert trainer to make use of their expertise to describe these activities as naturally as possible using a graphical representation. Basically, the graphical notation is expected to mediate between i) the description in a natural language of one or several professional activities, and ii) the equivalent description in a formal language that can be used by the game engine to run the scenario. BPMN is a good candidate, since its primary goal is to provide a notation that is readily understandable by all business users, from the business analysts to the technical developers to the business people (White, 2004). However, this representation does not support the configuration of each individual task. This is why the configurable building blocks approach introduced in (Van Est et al., 2011) is interesting: it allows the division of the global activity into a set of individual configurable tasks. For the rest of this paper, we will call them activity blocks.

Some activity blocks can be generic enough to be reused in whatever scenarios, like logic blocks or variables. However, a large majority of blocks will be context-specific. Anyway, we classify them into 3 categories: *events*, *actions*, and *observations*. Each of these blocks represent a different notion, but they all aim at describing activities from the objective perspective of the virtual environment rather than the player's subjective point of view. It allows to describe the expected result without having to specify how to reach this result.

The first notion a trainer would want to describe is the need to wait for a specific change in the virtual

environment to resolve the next task. This is exactly why we define *event blocks*. In most cases, they will be used to wait for the player to do an action, like ' When the door is opened, then ... ". But it can also be used to wait for a particular state of the environment induced by a previous action from the player, for example "When the car speed is above 60 km/h, then ... ". We represent *Events* as circles.

The second notion the trainer would need to describe is the fact that at some point during the execution of a given activity, they want to modify the virtual environment to place the learner in a specific situation. We use *Actions blocks* to express this idea. For example, it could be "Move the car from A to B", or "Change the Camera View to the inside of the car". We choose to represent *Actions* as rectangles.

Finally, the trainer may want to check one or several attributes of the virtual environment, to perform different actions depending on the observed result. We define *observation blocks* to represent this notion. It helps to represent cases like "If the lights are on, then ... else ... ". We represent *Observations* as diamond shapes.

We define each of these activity blocks to be able to possess a set number of arguments, so they can be reused in different contexts. For example, for the event "When the car speed is above 60 km/h, then ... ", the speed limit could be a variable argument in the block. Additionally, we want to allow the scenario author to reuse sequences of activity blocks in future scenarios or in different sections of a same scenario, without them having to describe the sequence twice. In this perspective, we let the trainer define *composite blocks*, which are basically the merging of a sequence of elementary activity blocks into one higher-level block. They also have a set number of arguments, so it is possible to quickly modify some of the parameters of the internal elementary blocks. These composite blocks are stored in a library accessible from any scenario.

Finally, the trainer may want to describe concurrent activities, to express the fact that two different tasks can be performed in parallel. To represent this idea, we use the BPMN notion of *pools*. A scenario has at least one pool, which contains the main activities description. It starts with a special event block named "Start" and ends with a another named "End". From this single pool, we can add as much as necessary, each of them containing a series of activities, starting with a global event to specify when the associated concurrent activities is being started.

3.2 Scenario Modeling: Pedagogy

The second dimension consists in defining how should be modeled the pedagogical feedback the learner will receive in the virtual environment while they play a training session. We introduce *pedagogy blocks* to describe these feedback.

A learning scenario has pedagogical objectives (Alvarez and Djaouti, 2011) (Garris et al., 2002) that need to be specified so the learner knows their progress along their activity and what they have learned so far. To let the non-expert express this notion, we define *objective blocks*. The writer can instantiate as many of these blocks as necessary.

We also introduce a tree view representation so the trainer can hierarchically order these objectives. In practice, completing one (high-level) objective could depend on completing multiple sub-objectives, and so on in a recursive fashion. Starting from the root which represents the global objective of the scenario, the author will be able to specify a hierarchy between objectives and sub-objectives.

3.3 Associating the Activity and the Pedagogy

Having described both the activities and the pedagogy of a scenario, by means of their respective graphical notation, interconnecting those two dimensions happens at the level of the activity block. The association is performed by the author. It consists, for each elementary objective (a leaf on the objective tree), in defining two notions: which task starts the objective? and which task completes the objective? This is achieved by connecting each objective to a composite block (i.e. a series of block as detailed in section 3.1), the first block within which being related to starting the objective and the last block being related to completing the objective.

In the objectives description, the author is given the possibility to define two sequences of *feedback blocks* per objective: one for when the the objective is started, and another for when it is completed. They can then express logic like "When the sub-objective 1 is started, display the advice..." or "When the objective 2 is completed, launch a MCQ to check acquired knowledge".

3.4 Application Example

To illustrate the proposed approach, we have come up with the Figure 2 presenting a scenario description example applied to the context of car diagnosis. In this scenario, the trainee is merely expected to establish

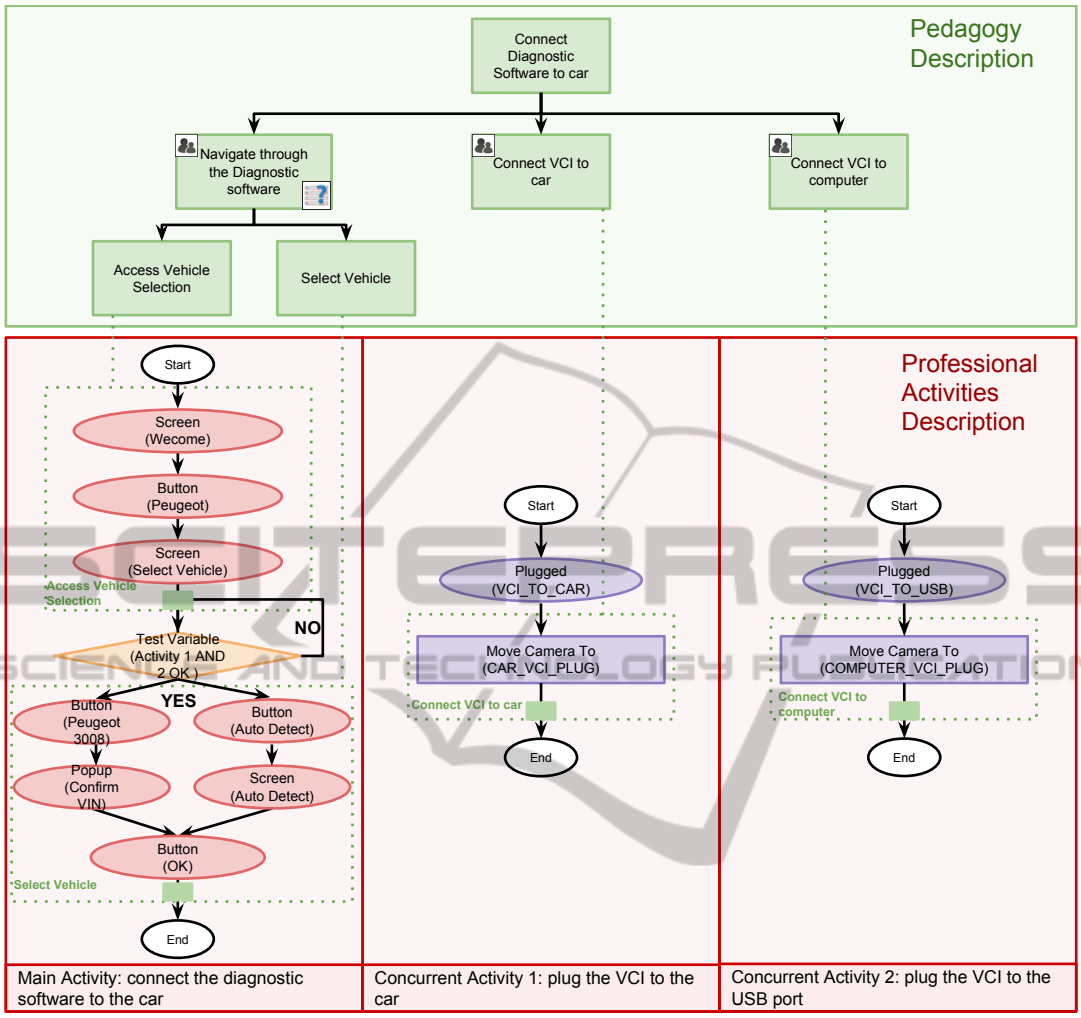

Figure 2: A toy-example scenario in Diag'Adventures. At the top (green), the pedagogy tree. At the bottom (red), the activities. Learning objectives and activities are connected by the dotted green lines.

the connection between the diagnostic software and the (virtual) car. The process is described in the professional activities description view. The software is mainly a succession of screens, in which the user can interact with a number of GUI elements, to progress in their diagnostic activity. We could just have described the succession of screens that would complete the main objective, if we did not care how the learner accessed them. However, in this particular scenario, we want the user to follow an optimal path; this is why we also describe the buttons that lead to this optimal use of the software. The two concurrent activities describe interactions in the 3D virtual environment that can be done at any time during the scenario.

In the pedagogy description, we defined the hierarchy between the scenario objectives and sub-objectives. Each leaf of the tree is then associated with a sequence of activity blocks in the activities description view. Pedagogical feedback are represented by the icons on the objective nodes/leaves. As an example, when the "Navigate through the diagnotic software" objective is started, we change the supporting message to indicate to the learner what they have to do. Then, when this objective is completed, we present a MCQ to the user to check if they have understood the process.

Through this example, we see the two advantages of the graphical representation and the methodology we propose. Firstly, it makes a clear division between the descriptions of the professional activities and the pedagogy. This allows trainers to simply express their dual-expertise into two orthogonal dimensions, and to establish the link between them only afterwards. Secondly, the common entity shared by these two dimensions, the activity block, uses a specific syntax which gives enough expressiveness for the user to de-

scribe a large amount of professional activities. That being said, it must be noted that the expressiveness is restricted to a specific context and therefore the approach is more suited for describing procedural activities as compared to activities entailing an absolute freedom of the learner inside the environment. As a matter of fact, it is easier to monitor the player's actions in a procedural scenario where we describe the optimal path, than in a free scenario where we could merely describe checkpoints. Besides, the pedagogy feedback could not be described as accurately as in a procedure. In the future, we plan to work on this aspect in order to cover a larger number of scenarios.

4 CONCLUSION AND PERSPECTIVES

In this paper we have described a methodology and the associated graphical modeling approach allowing the description of serious games scenarios. The main idea is to split a scenario into two dimensions different by nature yet connected: the pedagogy and the procedure. It is advocated that a non-computing expert can then express his expertise in the most natural fashion.

Future work will address the design and implementation of an authoring tool to validate this approach. We will define and model a number of various representative scenarios using our methodology. We intend to measure the expressiveness of our representation, and assess the proportion of the situations likely to be imagined by a trainer that can be modeled. Among these scenarios, we will have a particular interest in the description of collaborative tasks like in (Panzoli et al., 2014) or (Buche et al., 2010).

ACKNOWLEDGEMENTS

Diag'Adventures is supported by the Midi-Pyrénées region and collaboratively developed by ACTIA Automotive, OPERANTIS, and the University JF Champollion.

REFERENCES

Alvarez, J. and Djaouti, D. (2011). An introduction to serious game definitions and concepts. *Serious Games & Simulation for Risks Management*, page 11.

Araújo, M. and Roque, L. (2009). Modeling games with petri nets. *Breaking New Ground: Innovation in Games, Play, Practice and Theory. DIGRA2009. Londres, Royaume Uni*.

Buche, C., Bossard, C., Querrec, R., and Chevaillier, P. (2010). Pegase: A generic and adaptable intelligent system for virtual reality learning environments. *International Journal of Virtual Reality*, 9(2):73–85.

Devillers, F. and Donikian, S. (2003). A scenario language to orchestrate virtual world evolution. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 265–275. Eurographics Association.

Garris, R., Ahlers, R., and Driskell, J. E. (2002). Games, motivation, and learning: A research and practice model. *Simulation & gaming*, 33(4):441–467.

Ishida, T. (2002). Q: A scenario description language for interactive agents. *Computer*, 35(11):42–47.

Katsurada, K., Nakamura, Y., Yamada, H., and Nitta, T. (2003). Xisl: a language for describing multimodal interaction scenarios. In *Proceedings of the 5th international conference on Multimodal interfaces*, pages 281–284. ACM.

Kelleher, C. and Pausch, R. (2006). Lessons learned from designing a programming system to support middle school girls creating animated stories. In *Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006. IEEE Symposium on*, pages 165–172. IEEE.

Panzoli, D., Sanselone, M., Sanchez, S., Sanza, C., Lelardeux, C., Duthen, Y., and Lagarrigue, P. (2014). Introducing a design methodology for multi-character collaboration in immersive learning games. *Sixth International Conference on Virtual Worlds and Games for Serious Applications: VS-Games 2014*.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., et al. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11):60–67.

Tang, S. and Hanneghan, M. (2008). Towards a domain specific modelling language for serious game design. In *6th International Game Design and Technology Workshop, Liverpool, UK*.

Tang, S., Hanneghan, M., and Carter, C. (2013). A platform independent game technology model for model driven serious games development. *Electronic Journal of e-Learning*, 11(1):61–79.

Van Est, C., Poelman, R., and Bidarra, R. (2011). High-level scenario editing for serious games. In *GRAPP*, pages 339–346.

White, S. A. (2004). Introduction to bpmn. *IBM Corporation*.