A Scenario Editing Environment for Professional Online Training Systems

José Luis Aguirre-Cervantes and Jean-Philippe Pernin

Laboratoire d'Informatique de Grenoble,
961 rue de la Houille Blanche,
38402 Grenoble Cedex
{Jose-Luis.Aguirre,Jean-Philippe.Pernin}@imag.fr

Abstract. This paper presents a methodology and a graphical software environment dedicated to support the activity of various actors (instructional designers, training managers, pedagogical engineers, developers, learners, etc.), involved in the production of professional online training systems. After having presented the basis of our specific "wheel methodology" for building learning scenarios for those systems, we detail the models on which the methodology relies. The process behind the methodology follows a goal-oriented and a template-based approach. Finally, a prototype environment and the first experimentations carried out with this environment are described.

Keywords: learning scenario, professional training, goal-oriented process, learning templates.

1 Introduction

In previous works, we have focused our research on developing methods and tools aimed to model learning scenarios according to an *"intention-based"* approach [1]. Particularly, the ISiS (Intention, Strategies, interactional Situations) model assists a scenario designer (typically a teacher in secondary school) by providing him with a methodology based on the elicitation of actors' intentions (teachers, learners, domain specialists, etc.) for expressing a learning scenario. This one is considered as a hierarchical plan composed of strategies, interactional situations and learning components. After having experimented with this model in academic situations, we propose to adapt it to specific industrial situations where the needs and design contexts are different.

Our work was mainly developed inside the Learning Game Factory project (LGF, 2009-2011), granted by the European Regional Development Fund - Rhône-Alpes. LGF gathers four laboratories and five industrial partners around the question of how to use innovative objects in learning systems. Its goals are to integrate, control and

B. Murgante et al. (Eds.): ICCSA 2011, Part IV, LNCS 6785, pp. 502–517, 2011.

pilot, based on a pedagogical point of view, a large variety of components within professional online training systems. Concerned components can be learning games, simulations, interactive learning applications, etc. which can be developed in different languages or techniques. The aim of this paper is to present some of the main contributions resulting from this project.

Following this introduction, section 2 presents a specific methodology, the *"wheel iterative process"*, adapted for the building of professional training learning systems. We focus on providing involved actors with "agile methods" allowing an incremental definition and management of learning scenarios, by situating at the center of the process a goal-oriented and a template-based approach. We thus define a series of steps (device formulation, scenario design, scenario implementation, scenario execution, analysis and re-engineering), where the scenario is progressively enriched by different actors. For each step, the scenario can be characterized by a certain level of modeling. We propose then to consider those different models to constitute the basis upon which we have developed our environments.

In section 3 we present more precisely the core concepts of the two main models concerned with the scenario design step: the Context/Intention model and the Skeleton model. The elaboration of those models takes into account results of works about goal-oriented-design and intelligent agents' architectures.

Section 4 introduces Scenedit-II, the prototype of the scenario editor we have developed for the LGF Project. This environment provides the designers with three main *workspaces* from which they can progressively create the learning scenario. The first one, the *intentions workspace*, is dedicated to the definition of context and intentions, in terms of concerned knowledge, of associated constraints to the training context or of expected motivations for the learner and/or other actors (game motivations particularly). The second one, the *library workspace*, proposes a set of libraries where components (plans, strategies, interactional situations, game components, etc.) in the form of templates can be picked up or stored. The third one, the *scenarisation workspace*, allows to organize, by using different visual representations, the learning scenario activities. At every moment of design, the users can explore the *library workspace*, in order to look for components adapted to the intentions or contexts formulated in the *intentions workspace*.

In section 5, we present the first experimentations carried out with our tools, which partially implement the "wheel iterative process" for the production of a learning scenario. The first steps, device formulation and scenario design, are almost completely covered; scenario implementation and execution in a dedicated player are briefly considered for simple sequential scenarios. In this section, we also stress the fact that our architecture allows to handle different visual representations for the same learning scenario, where each representation is suited to different needs coming from different designers. We finish the paper with conclusions and perspectives on some future works.

2 A Methodology for Building Professional Training Learning Scenarios

2.1 Methodology as a "Wheel Iterative Process"

As stated in the previous section, we are concerned with the engineering process of learning scenarios for professional training. As figure 1 shows, several actors of different nature and accomplishing different tasks take part in this process. In the rest of the paper, we will refer to these scenarios as *training scenarios* or simply *TS*.

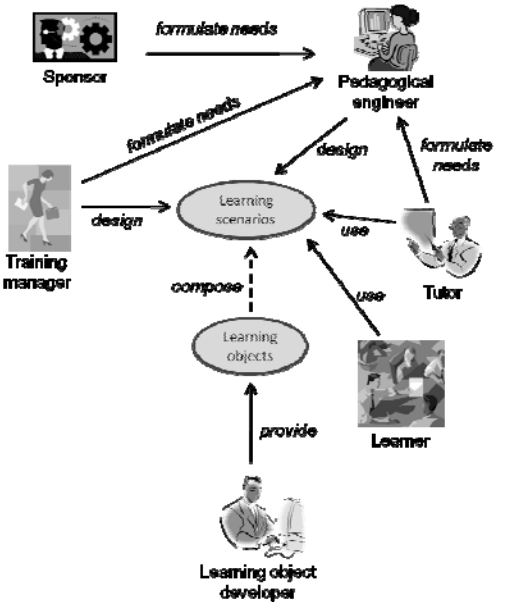

Fig. 1. Actors involved in the engineering of learning scenarios for professional training

Sponsors have the main responsibility of formulating the TS needs; TS needs, which can also be formulated by *training managers* and *tutors*, are captured and formalized by *pedagogical engineers*. Scenario design is the main responsibility of *pedagogical engineers*, *training managers* and *tutors*. *Learning object developers* are responsible of providing scenario components; finally, the learning scenario is mainly used by *learners* and *tutors*.

For this process, we propose a development methodology that follows an incremental iterative approach. This methodology, which is shown in figure 2, is inspired on results of previous research works on pedagogical scenario design [2] emphasizing the need of involving learning scenario designers in the elaboration of models and tools being proposed [3,1]. The methodology is strongly based on reuse concepts at several levels of granularity as such ones presented in [4].

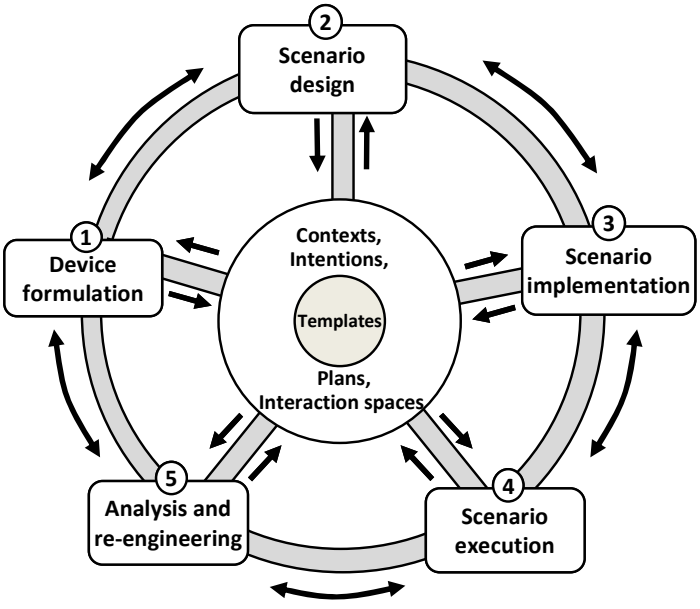

Fig. 2. Schema of an incremental iterative methodology for building professional TS

The process is directed by a collection of models situated in the center of the figure, as nodes surrounding the center represent the main phases constituting the methodology. Models can be originated from existing templates; they are elaborated by the actors appearing in figure 1, tackling different problems and being the main responsibles for the production of one or more specific models. For example, in the *Device formulation* phase, sponsors, training managers, tutors and pedagogical engineers must produce most of the *Context* and *Intentions* models. Moreover, production of models on later phases is guided by models defined in earlier phases. We will go deeper on that in section 2.2.

The figure shows that phases are not necessarily performed in linear order; on the contrary, the methodology could be seen as analogue to that one for developing software systems in a spiral and incremental way [5]. The arrows introduce a "wheel metaphor" reflecting the fact that, like in a wheel, sometimes we need to move forward (i.e. to get partial implementations of the scenario based on initial formulations), sometimes to move backward (i.e. to review the formulation based on partial implementations). In all cases, wheel movements are always dictated by the wheel hub, i.e. the models' contents in the center of the figure.

The proposal of this methodology is strongly influenced by the fact that the context of professional training learning scenarios development has some specificities which distinguish it from a more traditional academic context. In particular, the process is highly dynamic, i.e. the TS is constantly modified, and interactions between participants can last very short. Pedagogical engineers and sponsors can outline initial scenarios from the very first interview coming soon to initial model specifications. Some of these specifications can even directly concern advanced phases of the

methodology, and this is true all along the development process. This means that changes can be done to the models at any stage of the development process and following several directions.

One of the implications of such a methodology is that it can be considered as a workflow where different phases must be related by mediation of the models. This implies as well that we can *"short-circuit"* traditional linear flow, i.e. from *Device Formulation* to *Scenario Design* to *Scenario Implementation* and so on, permitting flows from one phase to any other phase. For example, scenario implementations could be modified without going first by changing initial formulations. As a result, the methodology is very *reactive,* so modifications to the implementation must be back propagated to modify the design of the scenario and what was set up in the initial formulation. To better clarify these ideas, suppose that the actors involved in the *Device Formulation* phase have established the intention to use a *competition simulation game*; if during scenario implementation, the actors access a template of a *role-playing board game* and decide to integrate it as one of the scenario's activities, the Context and Intention models defined at the beginning will reflect these changes.

In the next section we will briefly explain the different models coming out when the methodology is applied.

2.2 Methodology Models

In the *Device Formulation* phase, sponsors of the training device along with pedagogical engineers, and possibly training managers and tutors, set up the contexts where training will take place, and the intentions pursued by the training device with respect to those contexts. Typical examples of parts of the context are the targeted audience, the possibly roles played by that audience, the frames of knowledge and competences related to the training device, locations and resources used during training, etc. Intentions are next related with information included in the contexts. We will call the output of the first phase the Training Scenario Context/Intention Model (C/I Model); the form of the C/I Model is explained in section 3.1.

In the *Scenario Design* phase, pedagogical engineers in collaboration with training managers and tutors analyze the C/I Model for designing the skeleton of the TS. Taking into consideration the C/I Model, they specify what will be the main stages of the scenario and how those stages will be related, i.e. the logic of the scenario. This logic is defined in terms of operators indicating the way the stages will be chained: as a sequence, as a conditional alternative, as a single choice between several alternatives, as parallel branches, etc. General templates of common or specific activities can be used to define the skeleton, and also some specific activities can be attached to final stages. The output of this stage will be called the Training Scenario Skeleton Model (SkModel); the form of the SkModel is explained in section 3.2.

Next, the SkModel is refined by the pedagogical engineers and other actors like training managers, resulting in an execution model where more details are added about the use of the scenario. Specific training activities must be attached to all atomic stages of the SkModel. Attached activities can be created from scratch, or they can be recovered from libraries and repositories in order to be adapted to the specific needs of the training device. Based on what could happen in previous stages of the scenario, more subtle chaining conditions than those established in the SkModel are

defined for some of the atomic stages. For example, on the basis of resulting values of some parameters for precedent stages, one can specify what must be the initial state of another stage. Also, at this time training managers define the agenda for participants that will use the device during the training sessions: specific conditions about time and places, when and how the activities will be accessible, how the roles previously defined will be accorded to participants, evaluation rules for some of the stages, etc. The output of this stage is called the Training Scenario Execution Model (ExModel).

During the scenario execution phase, participants will work with the activities included in the TS. They will interact with players that can read and interpret the ExModels, and other actors like tutors or training managers, will follow the work of participants, analyze their performance, and possibly regulate at runtime some activities or adjust the scenario logic itself as a result of that analysis. As scenarios will be played by several actors, multiple outputs are the outcome of this phase. They will be called the Training Scenario Executed Models (ExedModel). Those models will include overall information about the main actions executed by the participants as well as some final results obtained by the evaluation means previously specified.

Finally, when the TS has been played by all the participants, a broader analysis can take place to evaluate the scenario itself. Validations can be done to verify, for example, if intended objectives were reached by the participants or if the inclusion of some activities has been valuable for enhancing participants' performance. Later on, the conclusions of those analysis can be taken into account for the adjustment of the same scenario to new training situations, or for defining completely new TS. The outcome of this phase is the Training Scenario Evaluated Model (EvModel).

3 Models and the Library Space

We present in this section models for the first two phases of the methodology: C/I Model and SkModel. The ExModel is by now partially developed as an extension and a transformation of the SkModel. The rest of the models are not considered for the moment. This section also includes the structure of a library of templates that can be reused for editing new TS.

3.1 Training Scenario Context/Intention Model

Following our approach, learning scenarios are first defined with respect to various contexts and intentions specific to the domain for which the TS is designed. In general, contexts are defined by all the objects and information that could play an important role for defining some part of the scenario, either a structural part or a behavioural part. For example, it is important, from the training perspective, to know where and how training will take place; what will be the places (physical or virtual) where learners will be trained; what kind of resources learners will access; what are the goal competences for which the training device is intended, etc. In our case, we identify three different types of context and intentions: organizational, training and knowledge. Figure 3 shows the UML class diagrams that represent the C/I Model.

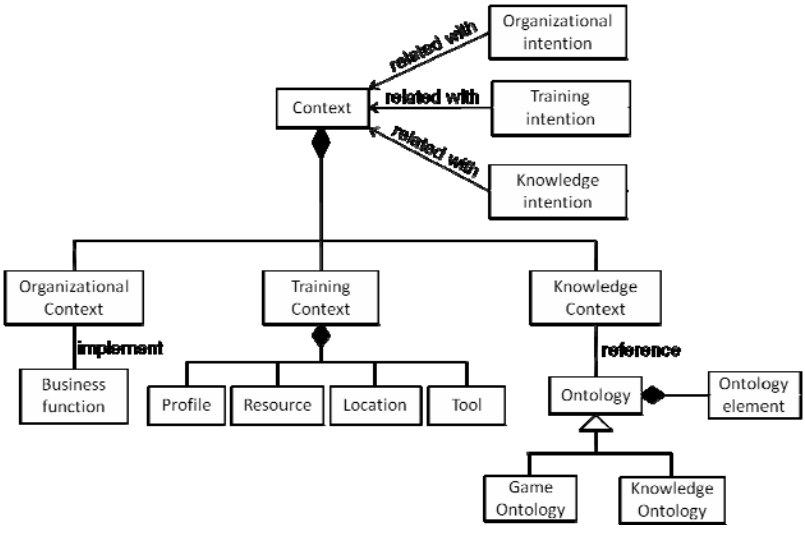

Fig. 3. C/I Model

The organizational context is restricted to some fields providing information about the enterprise, as its activity sector, a general description, and the business functions and services targeted by the training device. Organizational intentions are general statements related with the vision and strategies of the enterprise, but restricted to the business functions defined in the context. For example, *"to improve the quality of ... function"* or *"to reduce the waiting time of users for ... function providing ... service"*.

Training context enumerates and describes the targeted audience, the locations where the training device will be used, and resources and tools needed by the users of the training devices. Targeted audience is specified in the form of profiles referencing the business functions. Training intentions refer to general organisational and implementation aspects of the training device: organisation of sessions for the targeted audience, type of training device (distance learning, blended learning), tutor requirements, etc.

Knowledge context is provided by several ontologies, some of them of general nature, some others specific to the domains addressed by the training device. They include concepts and relationships between those concepts that are relevant to the training device. Knowledge intentions referred to knowledge and skills intended to be acquired by users of the training device. Those intentions can be expressed in structured way following the pattern:

`(<formulator>+,<target-public>+,<competence verb>+,<items of knowledge>+)` where

- `<formulator>+` refers to the persons who provide the intention (at least one); it could be any of the actors taking part in the process shown in figure 1.

- `<target-public>+` refers to the profiles specified in the training context (at least one).
- `<competence verb>+` denotes a competency list of verbs selected from competencies ontologies (at least one).
- `<items of knowledge>+` refers to one or more nodes of the ontologies constituting the knowledge context. A special kind of general knowledge and context intentions are for example those ones related with games. Game context furnishes general information about the use of games inside the enterprise and game intentions specify what are the intended forms of game planning to be used in the training device under the form of game motivations (an extended list of Caillois [6]) and game mechanisms [7].

3.2 Training Scenario Skeleton Model

The purpose of the training scenario skeleton model is to specify the scenario's logic. More specifically, TS logic is defined in terms of the chaining of the activities permitting the targeted audience to attain the intentions established in the previous C/I Model. The activities are supposed to be held in the contexts contained in this C/I Model.

Various educational modeling languages (EML) have been proposed for designing pedagogical scenarios. Certainly the most referenced in the literature is IMS Learning Design [8], which has been created as a "neutral language" for covering two complementary needs: (1) to be able to express a large variety of learning situations, and (2) to implement created pedagogical scenarios on a large range of EML (interoperability). Our objective is not to propose an alternative to replace such languages, but as underlined in [9], we think that it is necessary to develop more high level authoring or graphical environments adapted to the needs of specific designers. In our specific context of industrial development, we have identified a design step, where designers must manipulate easily the scenario's logic without taking into account implementation details. In further steps, it is possible to translate this logic towards others more "interoperable" notations, such as IMS-LD.

Inspired on the Belief-Desire-Intention agent architecture (BDI) [10], we propose to represent scenario skeletons as hierarchical plans, where the logic of the scenario is based on the use of plan operators. Beliefs, desires and intentions denote mental attitudes, representing respectively the information, motivational and deliberative states of an agent. These mental attitudes determine in fact the agent behavior. As beliefs represents in some sort what the agent knows about the world, desires represent objectives or situations that the agent would like to accomplish or bring about; goals represents consistent active desires. Finally, intentions represent what the agent has chosen to do, i.e. desires to which the agent has to some extent committed. With respect to our proposal, we can consider that the C/I Model encompasses some of the designer beliefs (what they know), and all of the desires and intentions (what they want).

In most of the BDI agents implementations, intentions are achieved by the execution of plans, maybe partially conceived at the beginning, and with details being filled in as the execution progress. A plan is defined as a sequence of actions achieving a goal [11]. We claim that the problem of building a learning scenario can

be considered as a partial-order planning problem. As a consequence, the solution of that problem, i.e. the learning scenario, can be represented as a graph of actions instead of a sequence. The graph is organized by means of operators whose arguments can be sub-plans or primitive actions.

To support our claim, we state that following an intention based approach, learning scenario designers must formulate the skeleton model selecting the activities that are considered to be the most compatible with the intentions, and arrange them on the base of chaining operators considering some partial order restrictions defined by the intentions. On the one hand, this means that some intentions must be attained before others, and logically, the activities addressing the first ones must be considered before the activities addressing the second ones. On the other hand, this means that in some cases, some activities can be placed in the scenario without specifying which one comes first.

Examples of plan operators for pedagogical purposes are the sequence itself, the alternative, the free choice, etc. Figure 4 shows the graphical representation of a scenario skeleton where terminal nodes correspond to specific activities and all of the internal nodes correspond to sub-plans of the main plan represented by the root of the structure shown on the figure. In the figure, sequence operators are represented by circles, alternative operators are represented by diamonds, and free choice operators are represented by triangles. As it will be presented in section 4, other visual representations are possible.

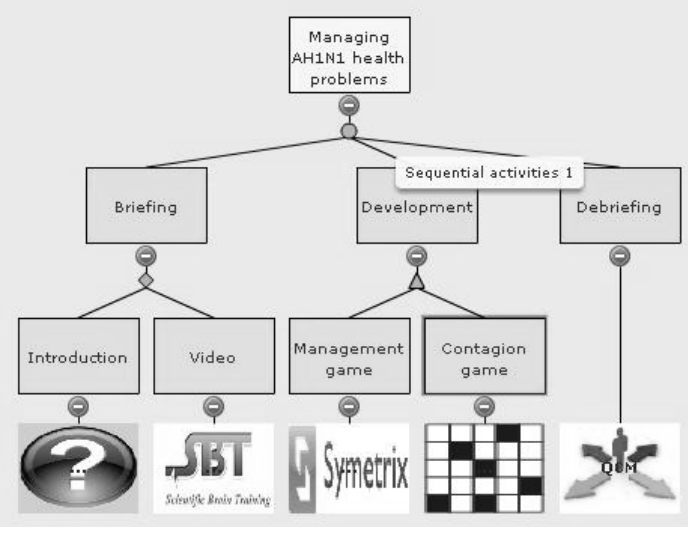

Fig. 4. Example of Skeleton Model

Skeleton terminal nodes correspond to conceptual objects denoted as *interaction spaces (IS)*. They reflect the fact that inside them, learners are confronted with the execution of the specific activities intended to work for achieving the intentions established in the scenario. In IS, learners can interact with dedicated interfaces, for

example to participate in a blog, to play a game, to work on the elaboration of a report, etc.

Figure 5 shows the UML class diagrams for the skeleton model, which is in fact an instance of the Composite design pattern [12].

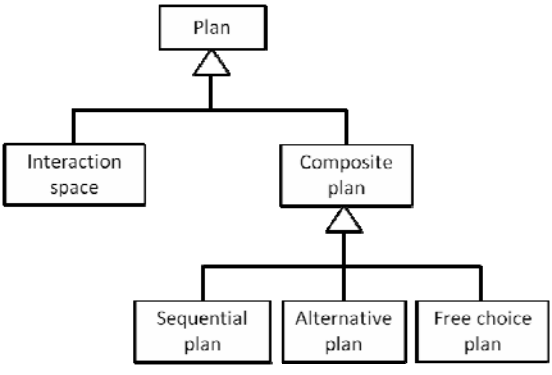

Fig. 5. UML Skeleton Model

Even if currently models are built by human agents, we believe that the fact of dealing with scenarios as plans can bring in many benefits in order to envision the development of artificial agents for assisting the tasks done when applying the wheel methodology. We will discuss about this in section 6 when talking about conclusions and future works.

3.3 Library Space and Interoperability

One of our concerns is the reuse of learning components at several levels of granularity, from basic non interactive components (videos, texts, etc.) to complex interactive objects (multiple choice questions, learning games, simulations, complete scenarios or part of scenarios, etc.). For this doing, we make use of learning object templates.

We must satisfy several requirements to incorporate the use of templates in all phases of the methodology presented in section 2. Specifically, during design and implementation phases, templates must be accessible to designers for being integrated through reuse and adaptation into the scenario's specification.

For design purposes, a template is a black box whose interface is contained in an information package expressed in the form of metadata. Metadata describes templates' properties for making possible their interoperability. By the moment, we propose an initial protocol allowing patterns to interoperate in terms of four capacities: *harvestability*, *configurability*, *observability* and *adaptability* (see figure 6).

Harvestability is the template capacity of being harvested, i.e. searched on according to filtering criteria; *configurability* is its capacity to be given an initial setting; *observability* is its capacity to be traced at runtime; *adaptability* is its capacity to be adjusted at runtime, i.e. to change its actual setting. The four capacities can be used at design time. For *harvestability*, templates are annotated using LOM format

[13]; for the other capacities we use a complementary XML file specification. For the LGF project, game classifications has been added to the <classification> section of the LOM specification.

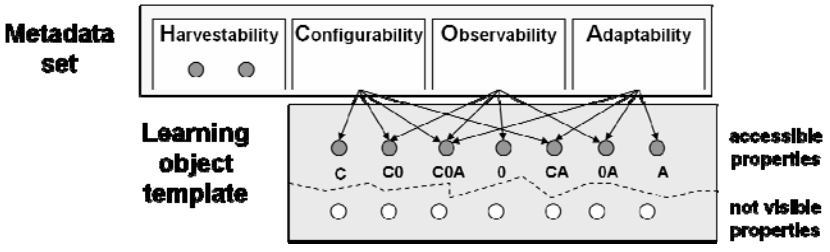

Fig. 6. Picture of a learning object template

4 ScenEdit-II: A Working Prototype

One goal within the LGF project was to offer a tool where different actors could work on the first phases of the methodology: from *Device Formulation* to *Scenario Implementation*. The ideas exposed in sections 2 and 3 were implemented in a prototype called *ScenEdit-II*. The tool's front-end is shown in figure 7.

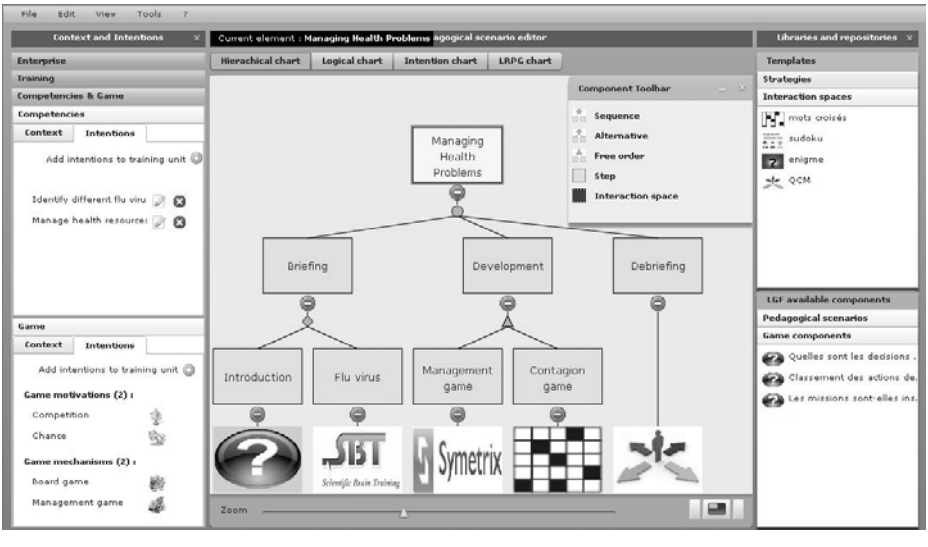

Fig. 7. An example of a training scenario developed with our editing tool

The first two phases of the methodology are completely included in the prototype and the Scenario Implementation phase is partially implemented. This will bring the

possibility to build the specification of TS that could be later executed. The interface is organized in three vertical workspaces from which one can progressively create the TS. Each space deals with the construction of one or more models produced by the methodology. The left part of the screen is dedicated to the construction of the C/I Model. Scenario designers can enter all the characteristics describing the 3 different types of contexts: Enterprise, Training and Knowledge (Competencies). Knowledge intentions can be entered in the structured format of section 2.1 by the use of graphical interfaces that reference ontologies' nodes that constitute the knowledge context (see figure 8).

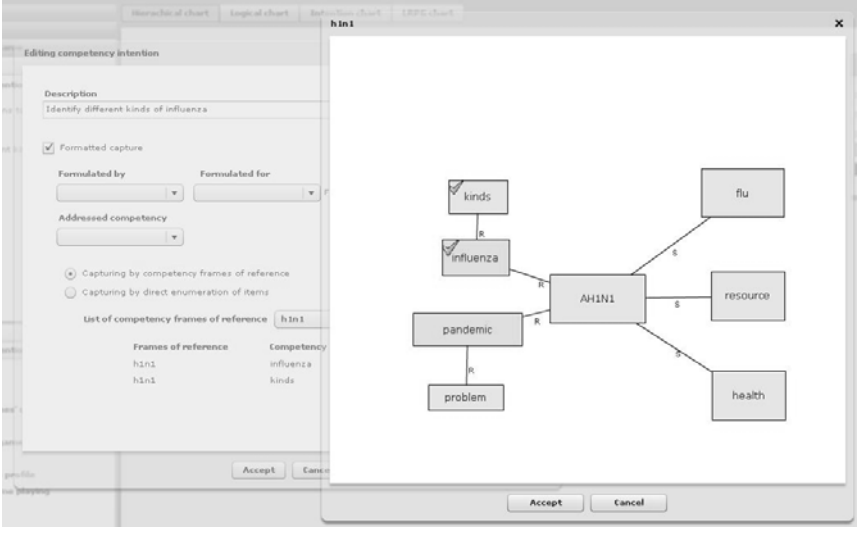

Fig. 8. Graphical interface for edition of knowledge intentions showing the graphical representation of an ontology

The central part of the screen is dedicated to the edition of the Skeleton and the partial Execution Models. Figure 7 illustrates the skeleton of a training scenario where the terminal nodes correspond to specific components based on different templates, while the internal nodes correspond to groups of activities organized by operators. In the figure, circles represent sequence operators, diamonds represent alternative operators, and triangles represent free choice operators. Thus, the scenario is composed of a sequence of three stages: the first stage is an alternative between two activities, and the second stage includes two activities that can be "played" in any order. This visualization mode is well suited for scenario designers as it shows in a very structured fashion the composition of the TS. Other visualization modes are possible like the one shown in fig. 9 representing the same scenario of fig. 7; this visualization correspond to a temporal view of the activities included in the TS, and is best suited for other actors as tutors or learners. In any case, edition of the TS is accomplished by simple drag-and-drop operations by accessing a component toolbar, or by context menus attached to the graphical representations of the scenario components.

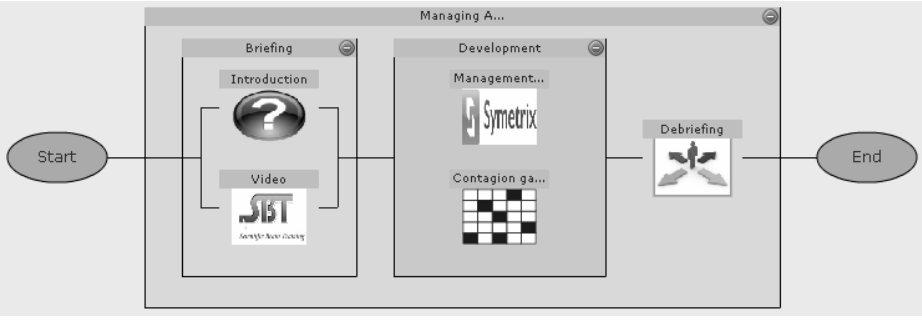

Fig. 9. Temporal graphical representation of a training scenario

Intentions of any kind can be attached to operator nodes. A propagation schema of intentions is established either in a top-down or a bottom-up direction. The former one means that we can attach to a node intentions coming from any of its ancestors; the later one means that if a new intention is attached to a node, that intention is considered automatically as an intention of all of its ancestors.

The right part of the editor shows the library of learning object templates. Templates can be gathered from local or external repositories by the application of special algorithms matching scenario intentions with harvesting metadata. Gathered components are ranked according to the result obtained by application of those algorithms; they can be easily integrated into the scenario skeleton by simple drag-and-drop manipulations.

For gathering purposes we consider the set I of scenario intentions as the union of three disjoint sets, $I = KO \cup GO \cup CV$ where

- KO is the set of nodes from knowledge context ontologies being referenced in the knowledge intentions,
- GO is the set of nodes from the special game context ontology being referenced in the game intentions,
- CV is the set of competency verbs appearing textually in the knowledge intentions

The actual matching algorithm implements a weighted function

$f(T) = w_1 m(KO) + w_2 m(GO) + w_3 m(CV)$ where

- T is a learning object template,
- KO, GO and CV are as explained before,
- w_1, w_2 and w_3 are constants such that $0<w_i<1$ and $w_1+w_2+w_3 = 1$. This constants are the weights of the matching criteria taking into account by the function. Weights denote the relative importance assigned to each criterion.
- m is a function which calculates in what extent the template matches a specific criteria. This is done by looking at particular template LOM tags, such as the <keyword> tag in the <general> LOM section or an <educational-objectif> tag included by us in the <classification> LOM section.

For each criteria the function `m` returns a result between 0 and 10, so the matching function `f` will return a result between 0, meaning that the template does not fill any of the TS expected intentions, and 10, meaning that the template fills all of the TS expected intentions.

The editor also allows the users to save partial and complete scenarios, to create new templates to be added to the library, and to transform the scenario's skeleton model into a partial execution model for which specific platform interpreters can be developed.

5 Current Experimentations

We have technically experimented with success the different phases of our methodology in the context of the Learning Game Factory project (LGF, 2009-2011). The protocol of our experimentation was the following:

- development by each of the five "component providers" of generic "learning game components" able to be harvested, configured, adapted and observed. For example, a component may be a generic "memory game", whose objects' properties, rules and interface can be adapted according to designers' needs;
- integration and indexation of the developed learning components in different repositories managed by each partner;
- in the scenario editor: harvesting of available components according to the design context;
- in the scenario editor: importation and adaptation of selected harvested components by accessing the metadata package that implements the interoperability protocol mentioned in section 2.3;
- at runtime, in a specific player developed for the experimentation by a partner: execution of sequential scenarios with dynamic settings and observation of harvested components.

We are currently working on the specialization of the tool in the context of training devices based on the use of role games [14]. The basic models of section 2.2 have been extended for this purpose.

6 Conclusions and Future Works

We have presented in this paper an iterative “wheel” methodology and a tool implementing this methodology to support the activity of the actors involved in the production of professional online training systems. The methodology is sustained in the enrichment of various models in order to progressively produce more detailed definitions of the training scenarios. Models for the two initial phases, formulation and design of the training scenario, have been introduced and explained. The first one captures the contexts and intentions of the device, as the second one uses a partial-order planning approach to define a structure that satisfies the intentions in the defined contexts.

The methodology is strongly based on the reuse of previously designed components expressed in the form of templates. A protocol has been defined allowing the recovering, filling and adaptation of those templates to new learning situations.

The first results on the use of the developed tool, ScenEdit-II, have shown that our approach was relevant and allowed to design a training scenario from the recovering and adaptation of objects coming from different sources based on the specification of an initial Context/Intention Model. A scenario design platform independent model is generated by the tool; this model is later interpreted by a specific player which executes the scenario.

As one of our future works, we want to refine and complete the models of our methodology following a Model-Driven Engineering (MDE) approach, in particular the execution model. MDE is proposed to overcome the difficulty inherent to the development of complex systems in order to obtain domain specific tools and applications through the clear expression of domain specific concepts [15]. The idea is to build a system through the successive construction of models moving from higher to lower levels of abstraction through well defined transformations. We consider that the task of creating learning scenarios is a complex situation justifying the utilization of this approach, as it is already proposed in other works like [16] and [17].

We want also to experiment with intelligent agents' technologies for assisting the users in the elaboration of entire or partial scenario structures, in particular with common and current planning algorithms. Artificial planners could propose solutions to designers based on various sources of knowledge. For example, a partial-order graph of the intentions could be used by partial-order planners to obtain initial solutions, which could be later refined by the use of knowledge about actors' preferences (tutors, learners, etc.), knowledge about general pedagogical rules, etc.

References

1. Emin, V., Pernin, J.-P., Guéraud, V.: Model and tool to clarify intentions and strategies in learning scenarios design. In: Cress, U., Dimitrova, V., Specht, M. (eds.) EC-TEL 2009. LNCS, vol. 5794, pp. 462–476. Springer, Heidelberg (2009)
2. Hotte, R., Godinet, H., Pernin, J.-P.: « Scénariser l'apprentissage, une activité de modélisation. » Numéro Spécial, Revue Internationale des Technologies en Pédagogique Universitaire, Montréal (2008)
3. Martel, C., Vignollet, L., Ferraris, C., Villiot-Leclercq, E.: A design rational of an editor for pedagogical procedures. In: 9th International Conference on Computer Supported Collaborative Learning, vol. 2 International Society of the Learning Sciences (2007)
4. Paquette, G.: Apprentissage sur Internet: des plateformes aux portails d'objets à base de connaissance. In: Pierre, S.(ed.) Innovations et tendances en technologies de formation et d'apprentissage, pp. 1–30. Presses de l'école polytechnique de Montréal (2005)
5. Boehm, B.: A Spiral Model of Software Development and Enhancement. ACM SIGSOFT Software Engineering Notes 11(4), 14–24 (1986)
6. Caillois, R.: Man, Play and Games. Free Press of Glencoe (1961)
7. Mariais, C., Michau, F., Pernin, J.P.: The Use of Game Principles in the Design of Learning Role-Playing Game Scenarios. In: 4th European Conference on Games Based Learning, pp. 181–184. Academic Publishing Limited, Copenhagen (2010)

8. IMS Learning Design Specification, IMS Learning Global Consortium (consulted on April 2011), http://www.imsglobal.org/learningdesign/
9. Koper, R., Tattersall, C.: Learning Design: A Handbook on Modeling and Delivering Networked Education and Training. Springer, Verlag (2005)
10. Wooldridge, M.: Reasoning About Rational Agents. MIT Press, Cambridge (2000)
11. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Prentice Hall, Englewood Cliffs (2009)
12. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading (1995)
13. Learning Technology Standards Committee of the IEEE. Draft Standard for Learning Object Metadata. IEEE, New York (2002), http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf (last consulted on April 2011)
14. Mariais, C., Michau, F., Pernin, J.P., Mandrau, N.: Learning Role-Playing Games: méthodologie et formalisme de description pour l'assistance à la conception. In: Environnements Informatiques pour l'Apprentissage Humain, Mons (2011), (to be published)
15. Schmidt, D.C.: Guest Editor's Introduction: Model-Driven Engineering. Computer 39(2), 25–31 (2006)
16. El-Kechai, H., Choquet, C.: Reusing Pedagogical Scenarios at a Knowledge Level: a Model Driven Approach. In: Seventh IEEE International Conference on Advanced Learning Technologies (ICALT 2007), IEEE Press, NY (2007)
17. Laforcade, P., Choquet, C.: Next Step for Educational Modeling Languages: The Model Driven Engineering and Reengineering Approach. In: Sixth IEEE International Conference on Advanced Learning Technologies (ICALT 2006), pp. 745–747. IEEE Press, NY (2006)