On-the-Fly Adaptive Planning for Game-Based Learning

Ioana Hulpuş[1,2], Manuel Fradinho[1], and Conor Hayes[2]

[1] Cyntelix, Galway, Ireland
ihulpus@cyntelix.com, mfradinho@cyntelix.com
[2] Digital Enterprise Research Institute,
NUI Galway, Ireland
conor.hayes@deri.org, ioana.hulpus@deri.org

Abstract. In this paper, we present a model for competency development using serious games, which is underpinned by a hierarchical case-based planning strategy. In our model, a learner's objectives are addressed by retrieving a suitable learning plan in a two-stage retrieval process. First of all, a suitable abstract plan is retrieved and personalised to the learner's specific requirements. In the second stage, the plan is incrementally instantiated as the learner engages with the learning material. Each instantiated plan is composed of a series of stories - interactive narratives designed to improve the learner's competence within a particular learning domain. The sequence of stories in an instantiated plan is guided by the planner, which monitors the learner performance and suggests the next learning step. To create each story, the learner's competency proficiency and performance assessment history are considered. A new story is created to further progress the plan instantiation. The plan succeeds when the user consistently reaches a required level of proficiency. The successful instantiated plan trace is stored in an experience repository and forms a knowledge base on which introspective learning techniques are applied to justify and/or refine abstract plan composition.

1 Introduction

Business is becoming increasingly global, distributed, faster paced and fiercely competitive [12]. In this context, corporate training is a major concern and challenge for companies. Traditional training methods are often expensive as they involve getting experts and trainees together in a common place [21]. While online courses have become an alternative, they also have a problem that content may be unsynchronised with the dynamic needs of the business [18]. In this context, the interest in interactive games for learning has been growing [7,14,32]. Our paper focuses on how to retrieve and adapt learning-plans as the learner engages with the learning material. The basic unit of instruction in our system is what is known as a *serious game*, an interactive role-playing narrative designed to teach particular competencies. A learning-plan is a sequence of such narratives that is customised to the learners particular requirements and skill levels, for

I. Bichindaritz and S. Montani (Eds.): ICCBR 2010, LNAI 6176, pp. 375–389, 2010.

which there is generally insufficient data until the learner begins to play. Our goal is to research how the game-based learning process can be continuously adapted towards the efficient development of required competencies for each individual learner. This entails the use of a planner that collects feedback from the learner interaction with the suggested plans and uses this feedback to learn which parts of the plan are failing and how to recover. As described by Hammond [10], case-based planning (CBP) systems have the capability of learning from the interaction with the human users. They can also anticipate problems and learn how to avoid them. These features make CBP attractive for use in learning environments.

There has been a persistent interest in learning among case-based reasoning (CBR) practitioners since the early days of CBR. This has yielded good results particularly in intelligent tutoring systems (ITS) [28]. Drawing upon this previous work and recent learning theory, we take a step further and present a model for online-learning with serious games that is underpinned by a case-based planning strategy. We also research how alternative plans which target the same goals can be represented, and retrieved based on their outcomes for different learners. The retrieved plans are then incrementally instantiated during execution, taking into account the information derived from constant performance monitoring.

Our research is done within the context of the TARGET[1] European Project. TARGET's goal is to implement a new type of Technology Enhanced Learning(TEL) environment which shortens the time to competence for learners within enterprises. The targeted learning domains are innovation and project management. The TARGET platform is based on serious games.

The remaining of this paper is structured as follows. The next section summarises related work that leverages the use of CBR for the purpose of supporting human learning. Then, in Section 3 we present an overview of our research direction in the field of CBP and learning with serious games. In Section 4 we describe how our work relates to the family of CBP systems and provide a detailed description of our proposed solution. In Section 5 we present a discussion of the implications of our system, and then we conclude with some final remarks on future work.

2 Case-Based Reasoning and Human Learning

Using CBR to facilitate human learning has appealed to many researchers, partly due to its roots in cognitive science and its similarity to human problem-solving behavior [23,15,22]. In [23], Schank argues for a goal-based approach to education, in which case acquisition plays a central role. In [15], Kolodner suggests how CBR can enhance problem-based learning by recommending relevant problems to learners.

The research done by Jonassen and Hernandez-Serrano [13], also supports the use of CBR for instructional design based on problem solving. The authors argue for a story-based system supported by CBR that enables learning from

[1] http://www.reachyourtarget.org

other people's experiences. These experiences form a case library of narratives from employees that describe real-life work-related problems and their solutions. However, we are focused on how to create a suitable learning plan, rather than retrieve a similar narrative.

There are already promising results of using CBR in ITS. For example, ILMDA (Intelligent Learning Material Delivery Agent) [28] that combines CBR with system meta-learning in the domain of computer science for undergraduates. An approach that comes closer to serious games is presented in [8,9]. They present a metaphorical simulation of the Java Virtual Machine to help students learn the Java language and reinforce their understanding of object-oriented programming. These two systems operate in well defined domains, where problems have a direct mapping to correct solutions. However, we are creating a system for use in two very complex domains: Project Management and Innovation Support. In these domains, the problems are open-ended and the competencies required are complex and difficult to model. Therefore, our approach is to create an open environment capable of reasoning with very complex, poorly structured domain knowledge. Furthermore, we focus on long term learning goals. For this, a single learning episode is not enough; the system must design consistent and coherent *learning plans*. As such, we use a CBP approach rather than classical CBR.

3 Approach Overview

The term competency carries many definitions in the related literature [11]. The definition that matches the best the way we use and assess competencies is that they are a set of personal characteristics, knowledge, skills and abilities that help successfully perform certain tasks, actions or functions and are relatively stable across different situations [31]. The initial TARGET competency framework will use the IPMA Competence Baseline[2] and SHL Universal Competency Framework[3].

The competencies of an individual are characterised by a state of attainment (degree of mastery) which the system estimates by analysing the user's *performances*. Therefore we define a competency profile as a set of competency-level pairs. A learner in our system has assigned both a *current competency profile*, and a *targeted competency profile*.

The core unit of instruction is represented by *stories* which are interactive narratives the learner engages with as he assumes a specific role, with a specific mission. A learning plan is a personalised sequence of stories. Throughout a story execution, towards the achievement of his mission, the learner is put in various *situations* meant to develop and evaluate his competencies. Each story has at its core a *story template* which can accommodate several situations. In order to create a story starting from a story template, a sequence of these potential situations is selected, based on the learner needs and requirements. The learning

[2] http://www.ipma.ch/certification/standards/Pages/ICBV3.aspx
[3] http://www.shl.com/OurScience/Documents/SHLUniversalCompetencyFramework.pdf

plan is created in a two stage process. First, an *abstract plan* is generated, as a sequence of story templates. Then, the concrete learning plan is instantiated incrementally, each story being created starting from the corresponding story template, when the plan execution thread reaches it. This process being the central focus of our paper, we describe it in more detail in the following sections.

3.1 Example

The TARGET game engine has the responsibility of instantiating stories with required competency-training situations, player roles and level, non-player characters and narrative thread. While the main focus on the paper is how the CBP mechanism can select stories, adapt and link them to create coherent learning plans, we give in this section a brief example of a story template and its "child" stories. We will come back to this example throughout the paper, to illustrate how the CBP engine decides the instantiation of the learning plan.

Figure 1 illustrates an example of a story template with its corresponding possible situations, and the competencies trained and evaluated in each situation. The situations are labeled with letters A-G. The arrows which lead from one situation to another show the possible flows of situations. For example, situation A "*Partner does not produce*", can lead to one or both situations B "*Conflict between partners*" and F "*Tasks not achieved or postponed*". The dashed lines in the figure illustrate the links between situations and the related competencies. For example, situation B trains and evaluates *conflict resolution.*

For each story template, an instantiated story consists of one path through its graph of situations. The game engine will instantiate the story according to the requirements stated by the CBP module. The story instantiations consists of: (i) selection of the story situations, (ii) instantiation of story parameters. Given the example in Figure 1, we can consider a user who wants to train in *conflict resolution, crisis management* and *resource planning.* Then, a candidate story is created by switching on the situations B, C and D. To train the required competencies, the learner chooses the role of project coordinator. During his

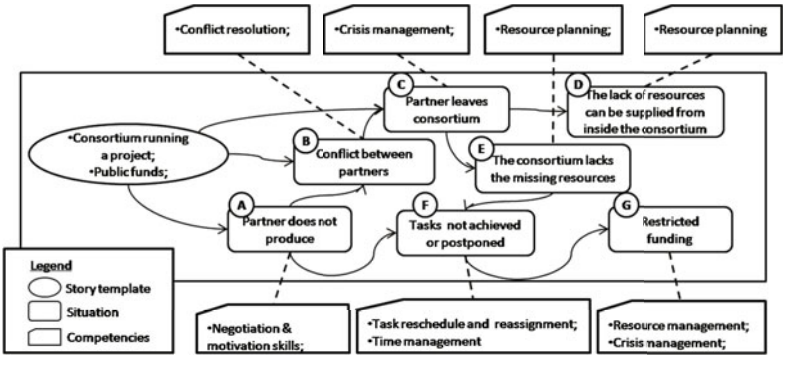

Fig. 1. Example of Story Template and Potential Situations

experience, the user is first evaluated on how he handles the conflict between the partners. Then he is evaluated on how he manages the situation where a partner leaves the consortium where other partners have sufficient resources to overcome the loss. Other candidate stories starting from the same template can be: $B \rightarrow C \rightarrow E$, or even $B \rightarrow C \rightarrow E \rightarrow F \rightarrow G$, which would suit a more experienced learner, or a learner who needs a more complex story.

Regarding the story parameters, for the given example such parameters would be the number of partners in the consortium, the number of partners involved in the conflict and the personality of the non-player-characters. All these contribute to an easier or more complicated story. Having the set of needed competencies, the case-based planner can choose the story templates to compose the abstract plan, but in order to know how to instantiate the stories (i.e, how to choose from the three stories we described above and instantiate the story parameters), it needs to know the performance of the user within the plan. Therefore, the story instantiated is delayed until the plan execution thread reaches its story template.

3.2 Principles of Linking Stories in a Learning Plan

A story represents a basic narrative unit for learning where closely related interactive situations focus on training the learner in a closely related competency set. However, a learning plan is a series of stories incrementally instantiated according to a learning strategy, but guided by a constant flow of feedback of the learner performance in each story. For example, a story may need to be repeated as a variation and situations added or removed and learning levels adjusted. The learner may be ready to move to a higher level in the next story or may not need particular situations to be instantiated. The learning plan must be created so that the flow of stories the user engages with lead him to the targeted competencies. For the creation of the learning plan we must consider the fact that the way learning episodes relate to each other is very important in order to keep the learner motivated and on the flow.

There are several aspects which we focus on in creating the learning plans. First of all, we have to consider if the company has a domain model where competencies have specific relations between them (e.g. decomposition, prerequisites. constraints). The learning model has to take these into account.

Secondly, it is important that the learning plan builds new competencies on top of existing ones. Following this principle, new competencies are developed in stories which also relate to possessed competencies. As well, within the learning plan the story complexity and the difficulty increases as the user performs.

The third principle is that learning needs practice, and often recursiveness and/or repetition. The variation theory of learning [19] and the cognitive flexibility theory [30] argue that practice of the same thing in different contexts, not pure repetition, leads to better learning outcomes. Following this principle, a learning plan should train the same competency in typical but varied situations until the learner reaches the desired level and also subject him to at least one atypical situation.

3.3 Reasoning with Abstraction to Build Learning Plans

In a game-based learning environment, a learning plan is an ordered list of stories meant to support the learner until he reaches the desired competency profile. The learning plan has to be adapted to the learner data like age, gender, cultural background. As well, it has to dynamically adapt based on the learner performances within the plan. This means that the planner does not have enough knowledge to create the whole plan in the initial stage of the planning. Therefore, at this stage several abstract plans are created, as sequences of story templates, and the learner can choose which one to execute. The story instances are created on-the-fly based on the story templates as the plan execution thread reaches them. At this stage the system has accumulated knowledge from the user's performances so far, and can personalise each story.

This methodology is inspired from the use of abstraction in case-based reasoning. By using abstraction, the less relevant features of a problem description are ignored in a first stage, leading to an abstract solution. Then, as the ignored features of the problem are being considered, the final concrete solution is derived from the abstract one [2]. In our case, the reasoner does not ignore features of the problem, but has to reason with an incomplete problem, which becomes complete as the solution is executed.

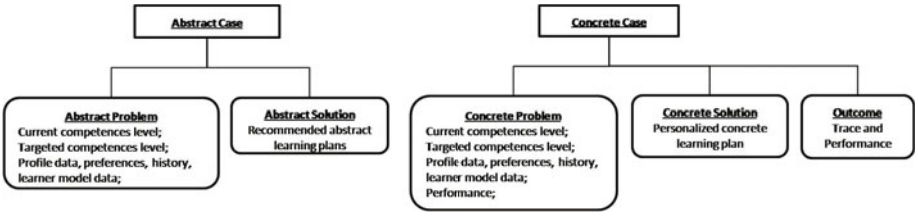

Fig. 2. Single Case Components

Following this hierarchical model, the abstract cases solve the problem requirements related to competency needs and learner profile data, by suggesting several abstract plans. The concrete cases have the problem enriched with the learner's performances, and therefore the solution is an iteratively created concrete learning plan. The two types of cases are represented in Figure 2.

4 Hierarchical CBP for Personalised Learning Plans

For planning on several layers of abstraction, many terms have been used in literature, the most common ones being hierarchical case-based reasoning [25] and stratified case-based reasoning [5]. The basic idea is that in the hierarchy of cases, only the "leaf" cases are concrete, and all the other nodes are "abstract cases". The studies made by Bergmann and Wilke [3], Branting and Aha [5] as well as

Smyth et al. [25], to name just a few, prove the advantages of this approach. Compared to classical case-based planning, it shows significant improvements in efficiency of retrieval and adaptation.

There are still differences in these approaches. In some of them, the abstract cases are created by abstraction and generalisation [1] of concrete cases. This is a bottom-up process which consists of merging concrete cases based on similar features. These are then discriminated based on their specific features, obtaining a hierarchical tree structure. In these systems the plans are retrieved entirely, and the new solutions are created by adapting them, e.g., in the PARIS system [3]. Other approaches create the abstract cases starting from task or goal decomposition. The concrete cases are the atomic actions which cannot be decomposed any more. The recent work of Lee, Chang and Liu in [17] uses such an approach. In these type of systems, each planning step is retrieved individually and then they are integrated to form the adapted solution. Smyth, Keane and Cunningham combine the two types of hierarchies in Déjà-Vu [25].

In our research, each planning step (a story instantiation) is not retrieved individually but is adapted by the user's previous interactions. Hence in our approach plan instantiation and the final steps of plan adaptation occur together. Generated abstract plans are presented to the user and he makes the choice of which one to follow. Every story generation is directly followed by user execution and system evaluation. The results are used to create new tasks for the subsequent steps.

In our solution, there are two levels of abstraction. In the systems which use abstraction it is common that the depth of the hierarchy is flexible, as the abstract cases are generated dynamically as soon as new cases share common features. The results of Branting and Aha in [5] show a significant improvement in efficiency when 3-4 levels of abstractions are used. If this proves to be valid in our system too, we will consider the option of using dynamic abstraction within each of the two current layers.

4.1 Abstract Plans

Abstract Case Representation. In order to represent the abstract cases we have to consider that there can exist multiple learning plans achieving the same learning goals. Consequently, all the plans which have similar initial states and goals are grouped under the same root. Then a description node is created for each abstract plan. This description node contains the users who executed the plan in the past, and a summary of their experiences (the plan outcome). This abstract-plan outcome includes information like the time the user needed to complete the plan, the average number of story repetitions, and the performances. It is important to note that this summary, although part of the abstract case representation, is extracted from the concrete layer. This way, we compensate for the loss of information which is inherent in reasoning with abstract cases [3]. Including this information in the description node gives us the possibility of combining CBR with collective filtering. In this scenario, collective performance information from similar learners will help in ranking candidate cases. The model

also supports the inclusion in the description of learners who were recommended the plan but did not choose to execute it. This information lends itself to providing explanations (e.g. 8 out of 10 learners selected this plan, 6 out of 8 learners completed this plan).

The description nodes have as children the abstract plans they describe. This hierarchy is illustrated in Figure 3. As mentioned in section 3.3, an abstract plan is a list of story templates. In Figure 3, the abstract plan 1 is composed of the story templates *ST1* → *ST2* → *ST3*, and the abstract plan 2 is *ST4* → *ST3*.

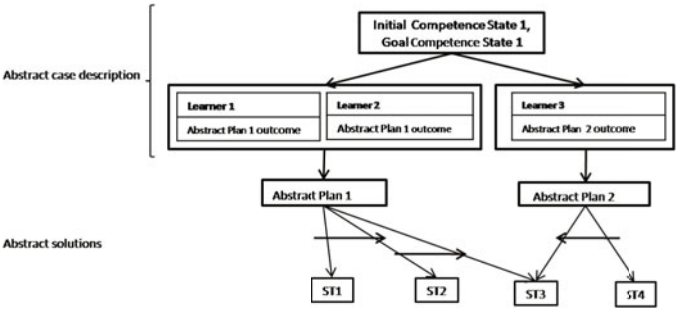

Fig. 3. Abstract Case Representation; ST - story template

Let us define the initial competency state as *(conflict resolution, beginner), (crisis management, beginner), (resource planning, upper average)* and the goal competency state as *(conflict resolution, average), (crisis management, average), (resource planning, expert)*. Then, the story template illustrated in Figure 1 is a good candidate for being part of the two abstract plans. Moreover, since it can bring together situations for all the goal competencies, it is a good candidate for being *ST3*.

Each story template in an abstract plan, has assigned the competencies it has to train and evaluate within that plan, forming an initial set of tasks. For example, let us consider the story template in Figure 1 is labeled *ST3* in Figure 3. Then, within the two abstract plans, the template is assigned the tasks to select situations which match *conflict resolution, crisis management* and *resource planning* since these are the competencies it was chosen for, even if it can support situations addressing other competencies as well (i.e., *negotiation*). This information is used when the story is created. It can be seen as an explanation why the template is part of the abstract plan. Still, this data is not enough for a personalised story. To personalise the story, more tasks to fulfill are assigned to the template as described later in section 4.2.

Abstract Plans Retrieval and Reuse. The retrieval of the abstract learning plan is a top-bottom traversal of the tree presented in Figure 3. This consists of two main steps: during the first step the system matches the current problem's initial state and goal to existing cases in the case base. Considering Figure 3,

this stage retrieves a set of nodes from the first level. During the second step the system retrieves the child nodes of the nodes returned after the first step. Then, for each such child it computes a *suitability* value, rather than a similarity value. The suitability value takes into consideration the learner similarity, the plan outcome for him and as well adaptation complexity [26].

After the most suitable abstract plans are retrieved, they are adapted so that they fulfill all the problem's requests: they fit the current competency of the learner, as well as his targeted competencies. The adaptation consists of adding/removing/replacing story templates from the original abstract plan.

4.2 Concrete Cases

Concrete Case Representation. Concrete case representation inherits from hierarchical representation used by Smyth et al. in Déjà-Vu [25]. The similarity comes from the fact that stories are generated step by step, therefore the final concrete solution is obtained by integrating the individual stories. Still, our suggested planner executes each step before instantiating the next. Both approaches permit multiple case reuse, which means that each planning step can be retrieved and reused from multiple cases.

As described in Figure 2, a component of the concrete problem is the set of previous user performances. Therefore, a learning plan that has been even partially executed by the learner is stored along with its performance score as a plan trace. The performances are analysed and depending on the result, the system selects and tailors the next story to play. Figure 4 shows the concrete plans layer, standing between the abstract plans layer and performance layer.

In the example in the figure, there are two abstract plans: $ST1 \rightarrow ST2 \rightarrow ST3$, and $ST4 \rightarrow ST3$. The first abstract plan has two instantiations, i.e., two concrete learning plans: $S1 \rightarrow S2 \rightarrow S3$ and $S1a \rightarrow S2 \rightarrow S3a$. The second abstract plan

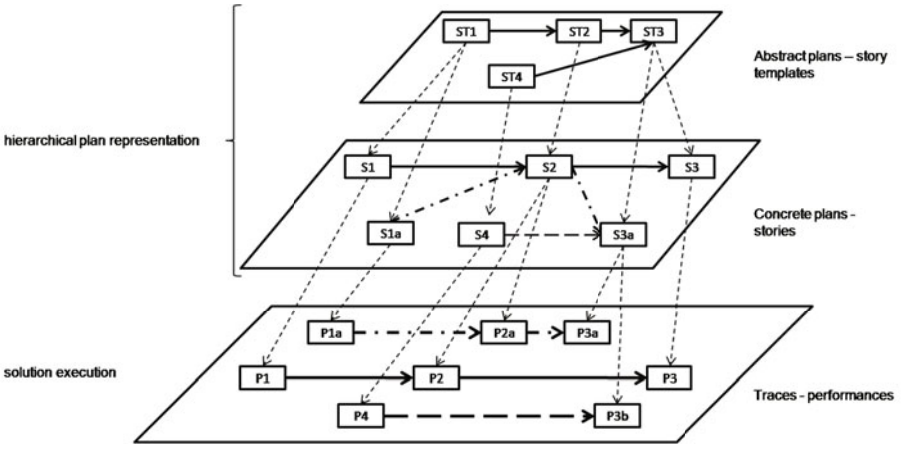

Fig. 4. Learning plans hierarchy; ST - story template; S - story; P - performance

has only one instantiation in the case base: *S4* → *S3a*. The arrows from the abstract plan to the concrete plan show how the story templates have been instantiated. The third layer shows how each concrete plan forms a trace as it is being executed. For example, the concrete plan *S1* → *S2* → *S3*, was executed once, leading to the trace: *P1* → *P2* → *P3*.

Let us consider the example in section 3.1, and take two learners, *L1* and *L2*. Because they have similar current and targeted competencies, they are recommended the same abstract plan: *ST1* → *ST2* → *ST3*. Let us further consider that the story template in Figure 1 is labeled *ST3* in Figure 4. Before the instantiation of *ST3*, the learner *L1* has executed two stories, *S1* and *S2*, with performances *P1* and *P2*. At the same stage, the learner *L2* has executed the stories *S1a* and *S2* with performances *P1a* and *P2a*. *L1* makes a good progress and successfully execute the tasks is short time. The planner can then decide to instantiate the *ST3* template to a complex story, therefore creating story *S3* as the flow of situations $B \rightarrow C \rightarrow E \rightarrow F \rightarrow G$. To make the story challenging, the planner also chooses to enforce a large consortium, with a spread conflict which determines a key partner to leave and cause a big resource gap. At the same time, *L2* has a slow progress, with blockages and long idle times, so the system decides to instantiate *ST3* into *S3a* as the flow of situations $B \rightarrow C \rightarrow D$. To make the story accessible, it defines a consortium of 4-5 partners with only two conflicting partners. A parter with a low contribution has to leave, and the lost resources can be covered from within the remaining consortium.

Planning on First Principles. The learning plan is created step by step, by instantiating the story templates of the abstract plan, at the moment they are needed or when the learner requests it.

Algorithm 1. NextStory(Learner, AbstractPlan, CurrentPlan, step)

```
   Input: Learner L, AbstractPlan AP, CurrentPlan CP, planning step n
 1 S_n = CP[n] ;                          /* S_n is the last executed story in CP */
 2 ST_n = AP[n];
 3 P_n = Performance(L, S_n);
 4 if P_n < ST_n.goal then
 5 |   T = GenerateSelfTasks(L,S_n,P_n);
 6 |   ST_n.tasks = ST_n.tasks ∪ T;
 7 |   S_n^a = CreateStory(ST_n);
 8 |   return S_n^a;
 9 else
10 |   if n + 1 < AP.length then
11 |   |   T = GenerateTasks(S_n, P_n, L);
12 |   |   DistributeTasks(AP, n + 1, T);
13 |   |   S_{n+1} = CreateStory(ST_{n+1});
14 |   |   return S_{n+1};
```

Algorithm 2. DistributeTasks(AbstractPlan, step, Tasks)

```
   Input: AbstractPlan AP, step n, Tasks T
 1 ST_n = AP[n];
 2 foreach task t ∈ T do
 3 |   if ST_n can satisfy t then
 4 |   |   ST_n.tasks = ST_n.tasks ∪ {t};
 5 |   |   T = T \ {t};
 6 if T ≠ ∅ and n + 1 < AP.length then
 7 |   DistributeTasks(AP, n + 1, T);
```

In order to create the next story, the system needs to interpret the previous performances and modify the remainder of the plan accordingly. Algorithm 1 illustrates how this is done. It uses a task creation and distribution mechanism shown in Algorithm 2: after a performance has been analysed a list of tasks is created. If the learner failed to reach the level planned for the current story, the planner can recommend him to replay a variation of the same story with a different difficulty level. Otherwise, the planner sends the package of tasks to the first subsequent story template. The story template keeps for itself the tasks which it can achieve and sends the rest further to the subsequent template in the plan, and so on. In case a story template cannot satisfy any new task, it is considered that it needs no further personalisation, and it is instantiated based on its initial set of tasks, set by the abstract plan.

An example of the concrete-plan creation process based on the task distribution is presented in Figure 5. The dashed arrows in the figure show how each performance triggers the delivery of tasks to the subsequent story template, which keeps for itself the tasks it can achieve and sends the rest forward.

As an example, let us consider that the story template *ST3* represents the story template illustrated in section 3.1, Figure 1. The template receives the tasks *T1* and *T3* due to the performance *P1*. *T1* states that the complexity of the story should be high, and *T3* requires that the *team management* competency should be approached so that it suits a beginner. *ST3* receives the two tasks but, because *T3* refers to a competency the story template cannot address, it can only keep *T1*. *T3* is sent further to the next story template. Due to previous performance *P2a*, *ST3* also receives tasks *T5* and *T6*. *T5* states that the competency *crisis management* should be approached so that it suits an average level learner. *T6* states that the set of competencies needed to successfully achieve the story mission must include the learner's targeted competencies, but not exclusively.

When *ST3* needs to be instantiated, it has to consider the tasks *T1*, *T5*and *T6*. Because of *T1* and *T6*, the story is created so that it brings a large number of situations. Because of *T5*, the situations have to be chosen and adapted so that *crisis management* is required in many situations, but not very demanding. This requirements lead to the instantiation of story *S3* as the flow of situations $B \rightarrow C \rightarrow E \rightarrow F \rightarrow G$ with parameters instantiated so that situations C and G cannot be handled unless the learner has an average level of proficiency in crisis management (due to *T5*).

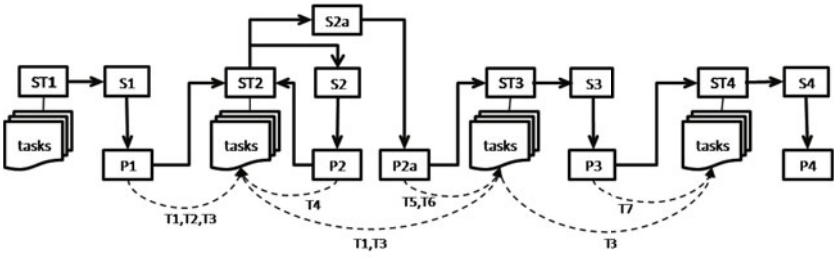

Fig. 5. Plan Instantiation Example

Using CBP Adaptation Techniques to Create Concrete Plans. The way the stories are created at this step from the story templates, can be either based on first-principles, or using one of the case-based-planning adaptation techniques like *derivational* or *transformational analogy* [20,29,6]. If the stories are generated on first-principles planning, then the system does not need to retrieve concrete plans from the case-base. They are created starting from the abstract plan, using only domain knowledge. Knowledge about how the tasks are generated from performances is needed. As well, how to instantiate a story starting from a story template and a set of tasks.

The transformational approach relies on the fact that the system saves entire plans and the new solution is created by reusing the old solution. If we use such an approach, then the system does not care to learn about tasks. If the old story's performance was partially similar to the current story's performance, then the system adapts the next story in the old plan to achieve the new story. In this approach domain knowledge is needed to be able to adapt a story to new previous performance.

On the other hand, using the derivational analogy, the cases are adapted based on the way the old solution has been built. Here, the system does not save the stories, but the tasks which lead to them. If the old story's performance is partially similar to the current story's performance, then the system adapts the old set of tasks and creates the new tasks. Using these tasks, it generates the story. Here, the system needs domain knowledge on how to integrate the tasks in the story creation.

Still, research and evaluation of the possible approaches has to be done before the best fitted solution can be selected.

5 Discussion and Future Work

By now, we have presented how learning plans are created for the learners and adapted to match their needs and performances. Another crucial part of case-based systems is the retain phase, during which the system adds the new cases to the case-base. The case base should avoid redundancy and be kept at a size which does not negatively influence the retrieval and adaptation efficiency. For this, we propose to keep all the traces and experiences in a separate storage, and then periodically carry out maintenance analysis [24] to make sure that only the cases which bring value to the case-base are retained.

Keeping the traces of successful and failed plans allows us to analyse the features and feature weighting that are leading to unsuccessful retrievals. Introspective learning techniques for feature weighting are designed to increase or decrease the weights of selected case features on the basis of problem solving performance [4]. Such techniques have also been used to facilitate easier adaptation of cases [16]. Analysing the repository of plan traces using introspective learning should allow us to improve the retrieval of abstract cases and their adaptation to the learner context.

Throughout this paper we mention the user data like age, gender and demographical details to be used for finding the suitable plan. Although it has been proven that cultural background, age and gender might influence a person's way of learning, we have to analyse if this data is relevant in our system. Therefore, we will use this data only for analysis during the early stages of the case base. If the analysis of cases proves any relation between learning and these parameters, we will consider them for plan retrieval.

Another aspect we have to consider when plans and stories are generated is diversity [27]. We need diversity both for the learner and the system. For the learner, it is important that recommended plans are varied and do not overlap with the user's already executed plans. For the system, it is important that it explores the efficacy of new plans as well, not only relying on old highly evaluated ones.

While the goal of this paper was to present a model of CBP and online learning using serious games, we should discuss our plans for implementation and evaluation. This work is being developed as part of the large European project TARGET, which contains academic and industrial partners. Although the case-based planning engine will be evaluated iteratively in small user trials, the full integration with the game engine and the evaluation of the overall system by TARGET industrial partners will take place in early 2011.

6 Conclusion

We have presented a methodological framework for creating personalised learning plans based on serious games - interactive narratives designed to teach particular competencies. We justified our reasons for proposing a novel a case-based planning approach and described in detail our hierarchical case structure and our iterative retrieval and adaptation process. We proposed that the learning process can be continuously adapted for each individual learner. We showed how alternative plans which target the same goals can be represented, and retrieved based on their outcomes for different learners. The retrieved plans are then adapted on-the-fly, based on an evaluation of the learner's performance. We proposed a hierarchical planning methodology which enables the planner to retrieve and personalise the learning plan for each user. We also examined how plan traces from all learners can be exploited to improve the case base of learning plans. This work is being developed as part of the European project TARGET and will be evaluated iteratively in small user trials. The full evaluation by the TARGET industrial partners is planned for early 2011.

Acknowledgments

This work was partly supported by the TARGET project under contract number FP7-231717 within the Seventh Framework Program, and by the Lion II. project funded by SFI under Grant SFI/08/CE/I1380. The authors also wish to thank Marcel Karnstedt for the many fruitful discussions.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations and system approaches. AI Communications 7(1), 39–59 (1994)
2. Bergmann, R., Wilke, W.: Building and refining abstract planning cases by change of representation language. Journal of Artificial Intelligence Research 3, 53–118 (1995)
3. Bergmann, R., Wilke, W.: On the role of abstraction in case-based reasoning. In: Advances in Case-Based Reasoning. LNCS (LNAI), pp. 28–43. Springer, Heidelberg (1996)
4. Bonzano, A., Cunningham, P., Smyth, B., et al.: Using introspective learning to improve retrieval in CBR: A case study in air traffic control. In: Leake, D.B., Plaza, E. (eds.) ICCBR 1997. LNCS, vol. 1266, pp. 291–302. Springer, Heidelberg (1997)
5. Branting, K., Aha, D.W.: Stratified case-based reasoning: Reusing hierarchical problem solving episodes. In: IJCAI, pp. 384–390 (1995)
6. Cox, M.T., Munoz-Avila, H., Bergmann, R.: Case-based planning. The Knowledge Engineering Review 20(3), 283–287 (2006)
7. de Freitas, S.: Emerging technologies for learning. Tech. rep., Becta (2008)
8. Gómez-Martín, M., Gómez-Martín, P., González-Calero, P.: Game-driven intelligent tutoring systems, pp. 108–113. Springer, Heidelberg (2004)
9. Gómez-Martın, P., Gómez-Martın, M., Dıaz-Agudo, B., González-Calero, P.: Opportunities for CBR in learning by doing. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 267–281. Springer, Heidelberg (2005)
10. Hammond, K.: Case-based planning: A framework for planning from experience. Cognitive Science 14, 385–443 (1990)
11. Harzallah, M., Berio, G., Vernadat, F.: Analysis and modeling of individual competencies: Toward better management of human resources. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans 36(1), 187–207 (2006)
12. IBM, Seriosity: Virtual worlds, real leaders: Online games put the future of business leadership on display. Tech. rep., IBM and Seriosity (2007)
13. Jonassen, D.H., Hernandez-Serrano, J.: Case-based reasoning and instructional design: Using stories to support problem solving. Educational Technology Research and Development 50(2), 65–77 (2002)
14. Kebritchi, M., Hirumi, A.: Examining the pedagogical foundations of modern educational computer games. Comput. Educ. 51(4), 1729–1743 (2008)
15. Kolodner, J.L., Hmelo, C.E., Narayanan, N.H.: Problem-based learning meets case-based reasoning. In: Second International Conference of the Learning Sciences (1996)
16. Leake, D., Kinley, A., Wilson, D.: Learning to improve case adaptation by introspective reasoning and CBR. LNCS, p. 229. Springer, Heidelberg (1995)
17. Lee, C.H.L., Cheng, K.Y.R.: A case-based planning approach for agent-based service-oriented systems. IEEE International Conference Systems, Man and Cybernetics, pp. 625–630 (2008)
18. Leyking, K., Chikova, P., Loos, P.: Competency- and process-driven elearning - a model-based approach. In: International Conference of E-Learning (ICEL 2007), New York (2007)
19. Marton, F., Trigwell, K.: Variatio est mater studiorum. Higher Education Research and Development 19(3), 381–395 (2000)

20. Munoz-Avila, H., Cox, M.T.: Case-based plan adaptation: An analysis and review. IEEE Intelligent Systems
21. Nebolsky, C.: Corporate training in virtual worlds. Journal of Systemics, Cybernetics and Informatics 2(6), 31–36 (2004)
22. Richter, M.M., Aamodt, A.: Case-based reasoning foundations. Knowledge Eng. Review 20(3), 203–207 (2005)
23. Schank, R.C.: Goal based scenario: Case-based reasoning meets learning by doing. In: Case-Based Reasoning: Experiences, Lessons and Future Directions, pp. 295–347 (1996)
24. Smyth, B.: Case-base maintenance. In: Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (1998)
25. Smyth, B., Keane, M.T., Cunningham, P.: Hierarchical case-based reasoning integrating case-based and decompositional problem-solving techniques for plant-control software design. IEEE Transactions on Knowledge and Data Engineering 13(5) (2001)
26. Smyth, B., Keane, M.T.: Adaptation-guided retrieval: questioning the similarity assumption in reasoning. Artificial Intelligence 102, 249–293 (1998)
27. Smyth, B., McClave, P.: Similarity vs. diversity. LNCS, pp. 347–361. Springer, Heidelberg (2001)
28. Soh, L.K., Blank, T.: Integrating case-based reasoning and meta-learning for a self-improving intelligent tutoring system. International Journal of Artificial Intelligence in Education 18, 27–58 (2008)
29. Spalazzi, L.: A survey on case-based planning. Artificial Intelligence Review 16, 3–36 (2001)
30. Spiro, R.J., Feltovich, R.P., Jacobson, M.J., Coulson, R.L.: Cognitive flexibility, constructivism, and hypertext: Random access instruction for advanced knowledge acquisition in ill-structured domains. In: Constructivism and the Technology of Instruction: A Conversation, pp. 57–76 (1992)
31. Tobias, L.A.D.: Identifying employee competencies in dynamic work domains: Methodological considerations and a case study. Journal of Universal Computer Science 9(12), 1500–1518 (2003)
32. Van Eck, R.: Digital game-based learning: It's not just the digital natives who are restless. EDUCAUSE Review (2006)