AISB Journal 2(1), © SSAISB, 2010
http://www.aisb.org.uk

Petri Nets for Representing Story Plots in Serious Games

Cyril Brom*, Tomáš Holan*, Daniel Balaš*, Adam Abonyi*, Vít Šisler† and Leo Galamboš‡

* Department of Software and Computer Science Education, Charles University in Prague, Malostranske namesti 25, Prague 1, Czech Republic, *brom@ksvi.mff.cuni.cz ; holan@ksvi.mff.cuni.cz ; addabis@gmail.com ; adam.abonyi@gmail.com*

† Institute of Information Studies and Librarianship, Charles University in Prague, U Krize 8, Prague 5, Czech Republic, *vsisler@gmail.com*

‡ Department of Informatics and Telecommunications, Czech Technical University in Prague, Konviktska 20, Prague 1, Czech republic, *galambos@fd.cvut.cz*

Abstract

This article describes a novel technique for representing plots of stories in computer games and for unfolding stories according to these plots. The technique is based on Petri Nets. Its main advantages are that 1) it copes well with branching stories, which 2) can evolve in parallel in 3) large virtual worlds. While the issue of branching stories has been already studied in literature, the latter two issues have not been sufficiently addressed yet. The technique has been evaluated on a prototype implementation, and subsequently used in two serious games Europe 2045 and Karo. Europe 2045 is an on-line multi-player strategy game aimed at education of high-school students in economics, politics, and media studies, which is presently evaluated at high-schools in the Czech Republic. Karo is a storytelling simulation that is intended to engage students in decision making concerning social relations, and work on it still continues. While Europe 2045 simulates Europe in a coarse-grained way, Karo features a detailed virtual world inhabited by virtual humans. Besides presenting the Petri Nets modifications used in the prototype and in both of the games, the article details several scenarios from the games, and on a general level, it discusses the strengths and weaknesses of implementation of Petri Nets in virtual storytelling applications.

1 Introduction

The idea of using computer games to support training and learning objectives is more than 30 years old (de Freitas, 2006). Recent work has explored the potentialities of commercial strategy games and simulations in formal education and their alleged advantages over classical e-learning and edutainment tools, e.g. Egenfeldt-Nielsen (2005). Indeed, many of such games have been experimentally integrated to formal curricula in the last four years. Perhaps the most prominent case studies have been conducted with The Sims 2, Civilization III, and Europe Universalis II (Egenfeldt-Nielsen et al., 2006; Sandford et al., 2007; Squire, 2004), but other attempts exist as well. The results from these pilots are promising, but also ambiguous in some aspects, e.g. Squire (2004). Hence, so called "serious" or "educational" games are starting to achieve increasing amount of attention. These games are, contrary to commercial games, intentionally developed as educational tools, which makes their integration into formal education easier. For example, a role-playing game prototype Global Conflicts: Palestine has been recently evaluated in a Denmark high-school with positive outcome (Egenfeldt-Nielsen et al., 2006). Another studies are being conducted, including FearNot!, an anti-bullying educational game (Aylett et al., 2005), and Revolution, a multi-player educational role-playing game concerning American War of Independence (Revolution RPG, 2007; Francis, unpublished).

As a part of European funded project "Integration of IT Tools into Education of Humanities" we have developed an educational game Europe 2045 (Brom et al., 2007b), and are developing a game Karo. Europe 2045 is likely the first on-line multi-player strategy game worldwide aimed at education of high-school students in economics, politics, and media studies. Its core constitutes a social-economic simulation, i.e. it models Europe in a coarse-grained way. Karo is a storytelling simulation that is intended to engage students in decision making concerning social relations. Contrary to Europe 2045, Karo features a virtual world inhabited by virtual humans, that is autonomous agents (Wooldridge, 2002) imitating behaviour of humans embedded into the environment. We emphasise that Europe 2045 is a fully developed game, which is presently being evaluated by high-school students, while programming work on Karo still continues.

Importantly, narrative aspects are strong in the both games. The reason for that should be clear: storytelling has played an important role in humanities education since the advent of formal schooling (de Freitas, 2006). Stories help to build a learning context, students can better understand the problematic through them, stories increase their involvement, and consequently their motivation.

Specifying plots of stories and controlling the course of a game in accordance with these plots is a well known problem (e.g., Aylett, 2000; Louchart and Aylett, 2003). It was indeed one of the most challenging goals we faced during the development. Essentially, both games had to be designed in order to meet the following requirements:

a) The story plots to be branching.

b) The episodes to be triggered by various initial conditions depending on the time and state of the word (including virtual humans in the case of Karo).

c) The virtual world to be very large.

d) The episodes to can happen in parallel. Europe 2045 features more than 20 countries, which could be played simultaneously, each having defined different episodes. A prototype scenario developed recently for Karo features about 10 different virtual humans, which are simulated all the time, being located in different parts of the virtual world.

e) The technique for specification of the plots to be intuitive enough for a high school teacher or another user (typically an undergraduate university student of humanities) to be able to design new scenarios for the game.

In our previous work, we investigated Petri Nets as a plot specification technique (Brom and Abonyi, 2006). We found this technique extremely user-friendly and very fitting for large virtual words and stories evolving in parallel, contrary to other approaches described in literature (which have, it must be acknowledged, other advantages for other kinds of virtual worlds). More or less, advantages of Petri Nets match the requirements mentioned above. Hence, we have adopted Petri Nets for these games.

This article is a report on this work. The overall goal is to introduce Petri Nets as a new technique for representing plots in storytelling applications, and to discuss their applicability. In the whole text, the different perspectives of a designer, who use Petri Nets as a specification tool, and a programmer, who use it as an architectural underpinning of the story manager, are emphasised.

We start with introducing Petri Nets in general in Section 2. Section 3 analyses various methods for representing plots and controlling stories, contrasting them with Petri Nets. Section 4 provides a formal account of a Petri Nets modification we developed formerly in Brom et al. (2006). This section is aimed at explaining in depth a particular refinement of Petri Nets that can be used for storytelling purposes. An example of a prototyped story is given as well.

Section 5 introduces Europe 2045 and Petri Nets modification used in there, which differ a bit from the formal model described in Section 4. This part of this article is based on our work reported in Brom et al. (2007b). In Section 6, Karo is overviewed and its Petri Nets modification sketched. Since Karo should be regarded rather as a prototype at present moment, we will visit it relatively briefly here. Nevertheless, the results we gained so far allow us to conclude that Petri Nets can be scaled for applications featuring virtual humans. Section 7 discusses the strengths and weaknesses of our method and concludes.

2 Petri Nets

This section gives a brief description of Petri Nets in general. Petri Nets is a specification technique frequently used in software engineering. A basic variant of it consists of *containers* (or places, represented by a circle: ○), *tokens* ("the pellets": ●), *actions* (or transitions, ▭), and *transition function* (→). The containers contain the tokens. If a sufficient number of tokens is contained in specific containers, an action is triggered. After firing an action, the tokens that helped to fire this action are removed, and some new tokens are generated (see Fig. 1a). Which tokens fire which action and which action generates tokens to which containers is specified by the transition function (and depicted by arrows). At one instant, several containers can contain tokens, which allows for concurrent triggering of actions. Obviously, a conflict between two applications of the transition function may appear (see Fig. 1b). Such a conflict can be solved in various ways. For example, we can choose one of the actions non-deterministically.

This basic kind of Petri Nets can be extended by introducing different types of containers, tokens, and transition functions. For example, tokens can have a state: such modification is typically called *coloured* Petri Nets (the colour meaning the state). For more thorough introduction to Petri Nets, we recommend the reader to consult Petri Nets World (2007).

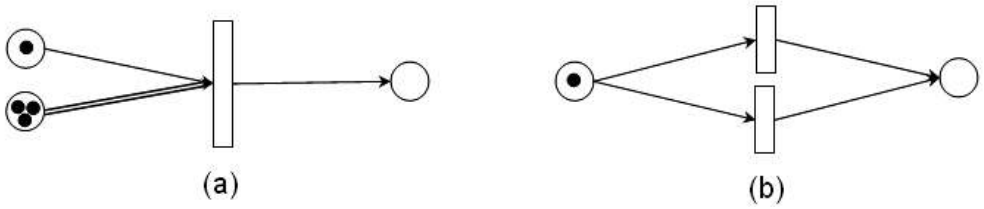

Figure 1: Petri Nets examples. a) The action generates one token if there is one token in the upper container and two in the lower container. b) The two actions are in conflict.

2.1 Petri Nets and a Story Manager

In this article, we discuss a particular kind of storytelling applications, that having plots of stories specified in advance by a designer. Actually, one may also be interested in automatic generation of stories based only on some pre-specified constraints, see e.g. Reidl and Young (2006), or in emergence of stories from interaction of autonomous human-like agents, as discussed in Aylett (2000), but this is not our case for our stories must precisely fit into the formal curricula (Brom et al. (2007b); note, however, that they still are branching and can evolve in parallel).

For explanatory reasons, let us view the architecture of the kind of applications we are discussing as being three-layered. The individual layers stand for a GUI, a simulator of the physical world, which presents the setting for the story (and possibly is inhabited by virtual humans), and a story manager (see Fig. 2). Each of the layers is more or less an autonomous component. The purpose of the GUI and the simulator are obvious. The story manager holds the representations of definite plot scenarios and a state of the story being unfolded. The state of the story can be changed based on the plots, and on the state of the simulated world. Conversely, state of the world can be changed based on its previous state and the state of the story. Importantly, not all changes of the state of the world are caused by the story manager! For example, in an application featuring virtual humans, the virtual humans can be controlled by the simulator most of the time (or they can be autonomous to some extent), and the story manager can only alter their high-level goals from time to time, i.e. only influence the virtual world, which is driven by the simulator primarily. Note, that also a user interaction changes the world (by definition of interaction). Hence, we can describe the interaction among the user, the world and the story manager by the following functions:

$$\text{story\_next}: \quad W \times S \rightarrow S \tag{1}$$

$$\text{world\_next}: \quad S \times W \times I \rightarrow W \tag{2}$$

Here, S is a set of possible states of the story, W is a set of possible states of the world, and I is a set of possible interactions of the user. The results of the function $story\_next$ is "computed" by the story manager, while those of the $world\_next$ by the simulator.

This notion of a separate storytelling component is actually not new. Various authors employ it, prescribing it various roles based on the intended features of the final application - see e.g. Magerko (2006). Following this architectural metaphor, Petri Nets are exploited in two ways. First, they present the mechanism for representing story plots, i.e. the function $story\_next$, a specification tool for the purposes of a designer. Second, in run-time, they also represent the state of the story being unfolded, i.e. $s \in S$. This means, that they are also the technical underpinnings of the story manager.

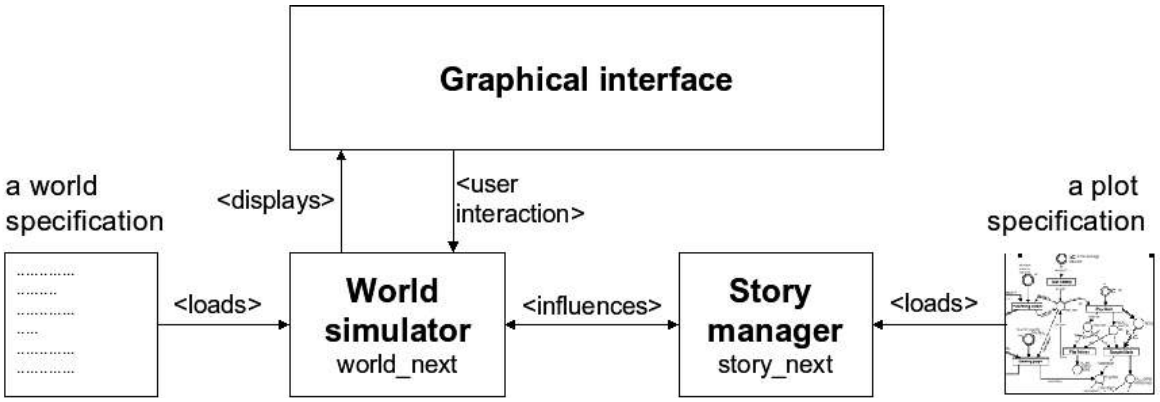

Figure 2: Architecture of a "typical" storytelling application that features a virtual world simulator and a story manager, which works with pre-specified plots of stories. Note that by virtual world, we mean both a world inhabited by virtual humans, which is the case of Karo, and that not inhabited by them, which is the case of Europe 2045.

3 Analyse of Plots Representational Techniques

The issue of generating/controlling stories in games and storytelling applications is notoriously known. Most techniques come from games and experimental simulations featuring human-like actors. This is also a case of Karo, but not Europe 2045. In this game, the story events are more abstract; they deal with whole populations, with a country economy etc. However, formally, the problem is very similar in both cases. Also the architectures of both application matches the schema at Fig. 2. In this section, we analyse several plot representational techniques, contrasting them with Petri Nets. We remind that our aim is evolving a story based on pre-specified plots with several requirements detailed in Sec. 1.

3.1 Rule Based Systems

A story manager viewed in an abstract manner as a component computing the function $story\_next$ (1) is actually a reactive component that responds to some events by triggering new events that change the state of the virtual world. Once we have this metaphor, we can easily come up with an idea of computing this by if-then rules matching a state of the story and of the world and executing an action in the world. Because of the requirement on large worlds (c), also large story plots, we argue that using just a list of such if then rules as a method for specifying plots is not a good idea. The reason is that for large plots hundreds of rules are needed. A designer that has to write down the rules needs a methodology, i.e. constraints that would guide him or her in the space of possible sets of rules that can be written down (see req. e)). A pure rule based system does not posses any such methodology.

Fortunately, there are several techniques that are a sort of rule based system, which present such a methodology. One of them is Petri Nets. Another one, which will be discussed here because it is already used in storytelling, is deterministic finite-state machines.

3.2 Finite State Machines

A well known branch of techniques for specifying plots are deterministic finite-state machines (dFSMs) (e.g., Sheldon, 2004; Silva, 2003). The formal specification of a dFSM

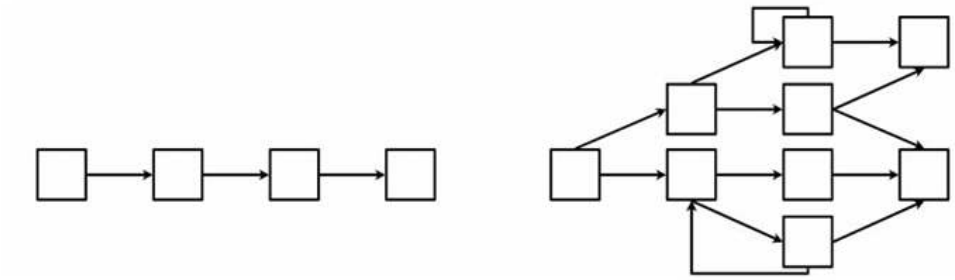

Figure 3: Story plots as dFSMs. The linear plot is on the left, the branching on the right.

constrains a manner of specifying the if-then rules, i.e. it presents the desired methodology. Each state of a dFSM represents a story episode, and a transition is a trigger that detects the end of the episode and starts a next one. Natural advantage of dFSMs is that they are formal, and yet graphical (Fig. 3), which makes them easily intelligible.

However, a classical dFSMs was not suitable for our purposes, since they cannot cope with the issues of parallelism (req. d)). On the other hand, non-deterministic FSMs can cope with it, but they are not easily comprehensible (e)).

Sometimes, so-called branching graphs or branching trees are used in storytelling, e.g. Sheldon (2004); Gordon (2004). We view them, more or less, as a sort of dFSMs, with similar advantages and disadvantages. Literally, they are not dFSMs, but describing subtle differences would be unnecessary for the purposes of this article.

3.3 Beat approach

Both for Europe 2045 and Karo, our aim was to author plots and to keep an unfolding story as close to one of "optimal stories" as possible. This objective is similar to this of Mateas (2002). However, his beat approach fits better to the domains of small worlds featuring virtual humans. We could not find out how to scale his solution to cope with the parallelism issue, and the large world issue (reqs c, d)).

3.4 HTN planning

In the field of emergent narrative and narrative generation, a planning formalism is frequently used, hierarchical task network planning being the most prominent (Aylett et al., 2005; Cavazza, 2002; Reidl and Stern, 2006). This technique can cope well with the requirements a), b), c), and d). Actually, it goes far beyond a pure reactive rule based story evolving in that a planner can perform a look-ahead search. However, HTN planning is not too friendly for a non-AI expert (req. e)).

To tackle e), one could introduce a "presentation layer" for a formal planning system to disguise the underlying representation and develop an authoring interface; however, this is time-consuming activity. In fact, Petri Nets (and dFSMs as well) naturally feature this presentation layer: as said above, they are a sort of rule based system, and it is this underlying rule based system, that they innately present in a graphical form.

Since we were interested in pre-specified plots but neither in a pure emergent narrative, nor in automatic story construction, and because of the unintelligibility disadvantage of the HTN planning, we decided not to use this approach. This does not mean, however, that for other domains, planning may not be the best option.

3.5 Petri Nets

Petri Nets have natural comprehensibility advantage of the dFSMs. They are formal, and yet allow for graphical depiction, which mirrors all of their (or most of their, depending on the complexity of a particular Petri Nets variant) formal features. Hence, they fit well as a specification interface between a game designer and a programmer. Indeed, we were able to use them both in Europe 2045 and Karo. First, they were employed in an informal manner as a specification tool for the designer, i.e. a tool for specifying the function $story\_next$ (1). Second, a rigorous counterpart of this informal specification tool presented the architectural underpinnings of the story manager developed by our programmer, i.e. a mechanism for computing the results of this function.

For the story manager, it is important to hold the state of the story, i.e. $s \in S$. How s and S are represented by Petri Nets? Let us contrast it with the representation of s and S in a dFSM. For a dFSM, S is a set of its states, and s is the state the machine is currently in. This actually holds for a Petri Net as well. However, a state of a Petri Net, i.e. s, is represented by which tokens are in which containers, and hence S is a set of all possible combinations of allocations of tokens to containers. This is much more flexible representation, and finally allows us to cope with the parallelism issue.

Petri Nets have been already employed in storytelling. Natkin (2003) used them to a retrospective analysis of a computer game story. They were also used for prototyping purposes (Brom and Abonyi, 2006), and for plot monitoring in a simple game (Delmas et al., 2007). However, to our knowledge, none of these work implemented a story manager for a real full-fledged game.

4 Petri Nets for Representation of Story Plots

As said in Section 2, Petri Nets are a whole class of specification techniques; many modifications exist. Not surprisingly, we have found that for purposes of each particular storytelling application, it is beneficial to develop a particular modification that fits best for what the application demands. For the explanatory reasons, we proceed now as follows. We first describe in detail a particular refinement of Petri Nets, based on so-called timed coloured Petri Nets. In fact, we used this refinement for prototyping purposes formerly, which allowed us to conclude that Petri Nets are a fruitful technique in the storytelling domain (Brom and Abonyi, 2006). In Section 5, we describe Europe 2045, and overview the Petri Nets modification it uses. Note that although our technique was intentionally designed for storytelling applications featuring virtual humans, neither the prototype, nor Europe 2045 actually exploit them. However, in Section 6, we introduce Karo, where the method is really used for controlling a story employing virtual humans.

To set the terminology, we say that a *PNA-model* is the type of Petri Nets formalised in this section, and a *PNA-plot* is a specification of an individual plot by means of the PNA-model. Sections 5 and 6 then introduce *PNB-model* and *PNC-model* respectively.

Every PNA-plot can be loaded from an xml-file by an application called TEST we developed, which features an abstract story manager for running and verifying PNA-plots. TEST works as follows: After a PNA-plot is loaded and the simulation started, events start to occur according to the PNA-plot in an abstract manner. That means that the TEST computes the $story\_next$ function (1), but there is no simulator of a virtual world to compute the $world\_next$ function (2). The result of this function is to be determined manually by a user through a check-box like interface in TEST; this mimics the changes that would be imposed on the world by the simulator or a user interacting with the world

in case of a full-fledged application.

4.1 Story example

For explanatory reasons, we first introduce a dummy story, whose plot was formalised as a PNA-plot and evaluated in TEST. The story is a simple fantasy narrative set in a village in an evening. It is a rather small story, just an episode from a larger tale. There are the following actors and groups of actors:

- MAGICIAN: a user-actor. She needs a puppet (for whatever reason).
- GUARDS: four bum-bailiffs. One of them is a friend of the magician; he has given her a note that there will be a puppet theatre coming this evening.
- PUPPET THEATRE: four artists. One of them is a twin of a guy who used to steal in the village several years ago, was arrested but managed to escape.
- ROBBERS: a band having pilfered in the village for a few weeks.
- VILLAGERS: citizens of the village. Some of them will mistakenly recognise the twin as his robber-brother.

Two things are going to happen in parallel. First, the robbers are going to pilfer this evening and the guards will try to capture them. Second, the troupe is going to perform a piece in a pub, and in the course of the play, some villagers will mistake the twin as his robber-brother and a brawl will flare up in the pub. Consequently, a fire might start, that would destroy the theatre and all the puppets.

Story can evolve in several ways. All events occur in a probabilistic manner: it is not sure whether or not the guards capture the robbers, or even notice them; whether or not somebody mistakes the twin; and whether or not the fire breaks out.

The magician is allowed to influence the story in the following ways: She can buy a puppet (before the theatre is reduced to ashes), or steal one in the course of the brawl. If she helps the artists in the brawl, she will be given a puppet for free, provided that no fire has broken out.

She can call the guards during the brawl using a spell. In this case, the guards interrupt the chasing and rush to the pub. This reduces the probability of the fire breaking out significantly, especially if guards have just caught the robbers. She can also put out the fire immediately by casting a spell.

A part of the PNA-plot for this story is depicted in Figure 4. The most important features of the portrayal are described in Section 4.2.

4.2 PNA-model

This section gives the formal description of the PNA-model and describes the most important features of the graphical representation of the PNA-plot depicted in Figure 4. An algorithm for driving a story according to a PNA-plot is also presented.

Notice that PNA-plot's portrayals are informative only; what is important is a background formal specification. Nevertheless, a designer works only with the graphical specification and informal textual descriptions; there is no need for him or her to know the formal delicacies, which must be known, on the other hand, to the programmer. As already said, it is the programmer whose job is to convert designer's plot specifications into the code, and clarify ambiguities that may have appeared.

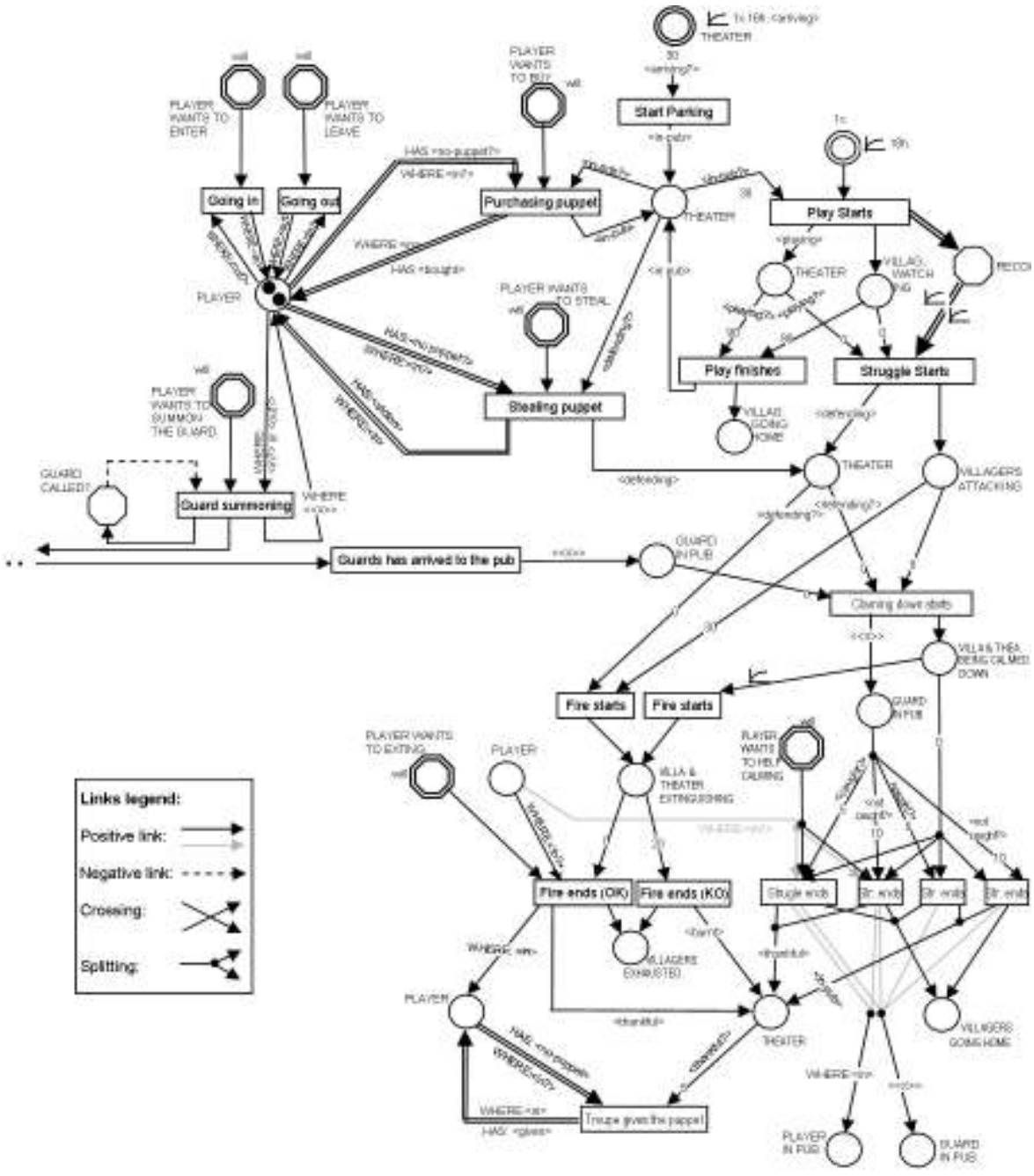

Figure 4: The "pub-plot" in the initial state is depicted. Notice that a plot can be much larger and several PNs can run in parallel. In particular, in our case, the "catching-plot" is connected to the "pub-plot" on the left (indicated by "…"). This part is not described here for brevity.

The PNA-model features following entities: *containers* (which are places – "the circles": ○, ◎, ▢, ▣), *tokens* (which are "the pellets": ●), *actions* (which are transitions – "the rectangles": ▭) and *triggers* (more or less, they correspond to a transition function – to "the arrows": →, ⇢). The purpose of a trigger is to fire an action according to tokens' location, or to add or to remove tokens from containers.

Tokens. Every *token* has a name, a colour, age, and a state. For every colour, there is a set of corresponding states. Actually, a colour is like a type of the token, but for conventional reasons, we keep the designation colour. Let us denote T_{ALL} a set of all possible tokens, N_{ALL} a set of all possible names, B_{ALL} a set of all possible colours, and S_b a set of all possible states for a colour $b \in B_{ALL}$.

Then, we say that *colouring* is a function: $\beta : T_{ALL} \rightarrow B_{ALL}$ – it gives a token's colour – and *token–status* is a function (defined for each colour): $\sigma_b : T_{ALL} \rightarrow N_{ALL} \times S_b \times \mathbb{N}$ – it gives a token's name, state, and age; and nothing if the token in question does not have the colour b. If an S_b is empty, we say that the token is stateless. A token can be located in a container. The age stands for how long the token is located there. In Fig. 4 only the initial tokens are shown. After the start of the simulation tokens would begin to appear or could be removed.

Containers. Every *container* has a name, a type and can be associated with a set of triggers, which are actually if-then rules. Let us denote C_{ALL} a set of all possible containers, Y_{ALL} a set of all possible types and I_{ALL} a set of all possible triggers. Then, we say that *container–status* is a function: $\gamma : C_{ALL} \rightarrow N_{ALL} \times Y_{ALL} \times \mathcal{P}(I_{ALL})$ – it gives a container's name, its type, and triggers. A container can contain more than one token in a given simulation time. We say that *containing* is a function $\kappa : C_{ALL} \rightarrow \mathcal{P}(T_{ALL})$ – it provides all tokens located in a given container in a given instant.

A semantic meaning of a token in a container is a denotation either of a state of a group of actors (i.e., an actor-token) or of a satisfied general precondition (i.e., a prec-token). In a drawing, containers are denoted according to their triggers and tokens they can contain as follows:

- ○: an actor-token & without any trigger,
- ◎: an actor-token & with triggers,
- ▢: a prec-token & without any trigger,
- ▣: a prec-tokens & with triggers.

There are following main token types in the PNA-plot from Fig. 4; italic denotes the initial state:

- "magician has" (colour m); $S_m = \{noPuppet, \mathrm{bought}, \mathrm{stolen}, \mathrm{given}\}$
- "magician where" (colour i); $S_i = \{inPub, \mathrm{outPub}\}$
- "guard" (colour g); $S_g = \{notCaught, \mathrm{caught}\}$
- "theatre troupe" (colour t); $S_t = \{\mathrm{arriving}, \mathrm{inPub}, \mathrm{playing}, \mathrm{defending}, \mathrm{burnt}, \mathrm{thankful}\}$
- "villagers", "robbers" (colour o); $S_o = \{\}$
- a prec-token: "has-called", "has-recognised" (colour o)

The meaning is obvious: for example, every "magician has" token represents that the magician has stolen, or has bought, or has been given a puppet, or does not have it, respectively. Similarly, the state of "magician where" stands for the magician in the pub or out of the pub. Notice, that these tokens can be located only in PLAYER container. Notice also, that a token is not a virtual human itself, it is just a representation of its state for the purpose of the story management!

Triggers. The most important primitive of the PNA-model is a *trigger*. A trigger can be associated with an action (an *action trigger*) or a *container* (a *container trigger*). Basically, a trigger is an *if-p-then-c* rule, where p is a precondition and c a consequence, which is to be performed when p holds. There are four types of triggers (both action triggers and container triggers).

- A *token-generating* trigger is a trigger that has a consequence of always adding some tokens to some containers while not removing any token.
- A *token-consuming* trigger has a consequence of always removing at least one token and possibly adding tokens to containers.
- An *action-firing* trigger neither generates nor removes any tokens, but fires an action.
- A *conflict-resolving* trigger's precondition tests whether two or more conflicting actions are to be fired at the same time, and its consequence resolves the conflict. What actually means a conflict between actions will be explained later.

Actions and triggers. Every *action* has a name, one action-firing, one token-generating and a token-consuming trigger, a "ready to fire" flag, and an effect. Essentially, firing of an action has two steps: first, the "ready to fire" flag is set, second, the action is really fired. The purpose of this separation is that there may be a conflict between several actions, and not all of them can fire.

The precondition of every token-generating trigger and every token-consuming trigger tests whether or not the flag is set. Let us denote A_{ALL} a set of all possible actions. Then, we say that *action–status* is a function $\alpha : A_{ALL} \rightarrow N_{ALL} \times \{0, 1\} \times I^3_{ALL}$ – it returns an action's name, whether or not its flag is set, and its triggers.

If an action-firing trigger holds, its consequence sets "ready to fire" flag. Then, the other two triggers can be triggered. Notice the word "can". In a given instant, more actions can be ready to fire, but not all of them can be allowed to fire for there can be a conflict between their token-consuming triggers: it is not possible to remove a token that has been just removed by another trigger. We decided to solve this "conflict issue" by introducing conflict-resolving triggers. A purpose of a conflict-resolving trigger is to detect such a conflict, its consequence is to simply unset the flag of some actions. Finally, when an action is really allowed to fire, its effect is performed, and its token-generating and token-consuming triggers are triggered. See Algorithm 1 below for details.[1]

There is no graphical primitive corresponding to a trigger directly. However, a container with a trigger is denoted as ◎ or ⯎, and an action trigger (action-firing, token-generating, and token-consuming) is depicted as a set of arrows, each from a container to

[1]We remark that conflict resolving triggers present a relatively complicated element. We noticed that designers of real scenarios for Europe 2045 and Karo typically specified plots avoiding the conflicts at the first place; they almost never used these triggers. Hence, we regard the conflict resolving trigger rather as a mean for having the model theoretically coherent than a way for really solving possible conflicts among actions.

an action, or vice versa. For example, ⊙→▭ denotes both an action-firing trigger and a token-consuming trigger at the same time, meaning "try to fire the action if there is at least one token in the container" and "consume one token when the action is fired". An example of a token generating trigger is ▭→○ (it means "generate one token when the action is fired").

Details of a precondition or a consequence of a trigger are indicated schematically next to an arrow, next to a container, or by a changed shape of the arrow. Precisely, a precondition can:

- test κ: i.e., whether some containers contain ($\rightarrow$) or do not contain ($\dashrightarrow$) some tokens; test β and σ_β: i.e., colouring ($< col? >$) and token-status ($< state? >$),
- test simulation time (HH hours),
- compare age (MM minutes) or simulation time (HH hours) to a random value generated using a given probabilistic distribution (📈),
- test whether a user will alter the simulation somehow ($will$),
- test whether a trigger has fired n times ($n\times$),
- test whether two or more actions are in a conflict ($======$).

A consequence of a token-generating trigger can generate a token of a specific colour and a state. This is indicated as $< state >$, $<< X >>$ or $< col >$ above the arrow. $<< X >>$ means that the generated token has the same state as the token that has been just consumed (see, for example, GUARD IN PUB container).

Notice that not all conflicts are always solved in Step 4 – there might exist a conflict not recognised by any trigger. In this case, the action to execute will be chosen randomly from the set of the actions in the conflict (Step 6).

4.3 Examples

Now, consider several examples taken from Figure 4. First, focus on the container THEATRE and the action *Start parking* (in the pub), which is redrawn at Figure 5a. There are three triggers indicated in the drawing. The first one generates a token "theatre" with *arriving* state into the container once at about 4 p.m. (Step 7, 8 of Algorithm 1). The second one is an action-firing trigger. It sets "ready to fire" flag when the token in the container is at least 30 minutes old (Step 1, 2). The third one is a token-consuming trigger, which removes "theatre" token when the action fires (Step 5, 6).

The next example concerns with the episode of the theatrical performance (Fig. 5b). The token-generating trigger of the *Play starts* action has generated four tokens; two "has-recognised" into the RECOGNISED? container, one "villagers" into the VILLAGERS WATCHING container, and one "theatre" with the state $playing$ into the THEATRE container. The action-firing trigger of the *Struggle starts* action tests whether or not there is a token with the state $playing$ in the container THEATRE, and a token in the container VILLAGERS, and two tokens in the container RECOGNISED? with the appropriate age. It is determined randomly for the both tokens, whether or not the token is old enough. If the action is fired, the next trigger – the token-consuming one of *Struggle starts* – consumes all four tokens. Notice that all durative episodes are represented in a similar way using the age of the tokens.

Fig. 5c represents the stealing of a puppet. If the magician is in the pub (denoted as WHERE: $< in? >$), and if she does not have a puppet (denoted as HAS: $< noPuppet? >$),

Algorithm 1 Unfolding a story in TEST using a PNA-plot.

Input: a PNA-plot.
The algorithm is performed in every time step t.

1. $I_{action,t} \leftarrow$ all action-firing triggers that fire in time t
2. Perform a consequence for all $i \in I_{action,t}$ in a random order (i.e., mark the "ready to fire" flag of the respective actions)
3. $I_{conflict,t} \leftarrow$ all conflict-resolving triggers that fire in time t
4. Perform a consequence for all $i \in I_{conflict,t}$ in a random order (it unsets the "ready to fire" flag of some of the actions being in conflict; it may include asking a user for selecting such actions)
5. $I_{remove,t} \leftarrow$ all token-consuming triggers that fire in time t (they test for "ready to fire")
6. for each $i \in I_{remove,t}$ (take i in a random order) do
   - if all desired tokens can be still removed do:
     - perform the consequence of i (it includes removing the tokens)
     - if i is an action trigger do perform the effect of the action
   - otherwise, if i is an action trigger do unset the "ready to fire" flag
7. $I_{generate,t} \leftarrow$ all token generating triggers that fire in time t
8. for each $i \in I_{generate,t}$ (take i in a random order) do
   - perform the consequence of i
   - if i is an action trigger do unset "ready to fire" flag

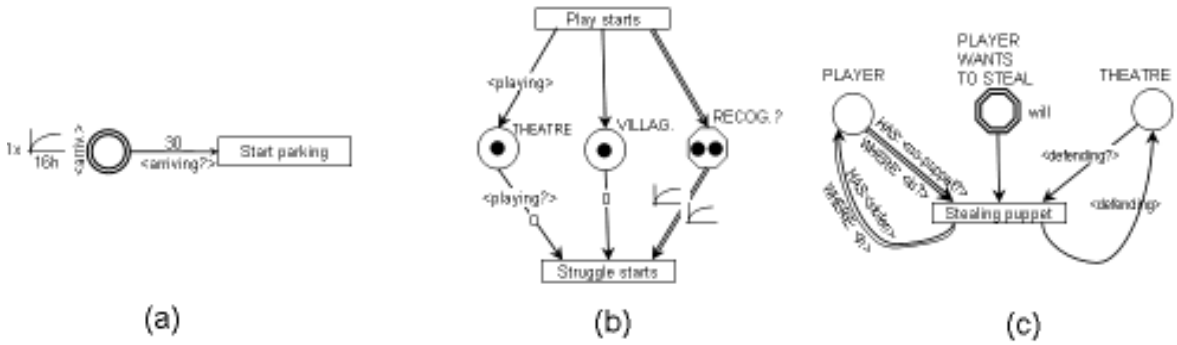

Figure 5: a) The theatre container example. b) The theatrical performance example. c) The stealing of the puppet example. Notice that the container THEATRE is depicted several times for convenience in 4.

and if she wants to steal, and if the brawl has already flared up (the state of "theatre" is defending), she can steal the puppet. Notice that the token "player wants to steal" is generated according to the choice of the user – this feature is achieved by checking a check box in the application TEST. Notice also, that from the perspective of the high-level plot, the action of stealing is instantaneous: it takes no time.

4.4 Summary

The purpose of this section was an explanatory one. We introduced in detail the PNA-model, on which we evaluated fruitfulness of Petri Nets in storytelling in general. The evaluation took place on the application TEST we developed that allowed for running PNA-plots in an abstract manner, i.e. without any virtual world connected to it (Brom and Abonyi, 2006).

Several things ought to be commented. First, notice that a story can really evolve in parallel. For example, the skipped "catching plot" part at Fig. 4 is actually unfolded at the same time as the "theatre in pub" part. Second, notice the level of abstraction of the events generated from the PNA-plot in Fig. 4. Apparently, the PNA-plot presents a high-level narrative structure; many low-level events are assumed to be generated by the simulator of the virtual world, possibly based on the user interaction (see Fig. 2 again). This gives substantial degree of freedom in possible ways of evolving the story, and this also presents advantage from the design point of view, because splitting the design part into specifying a high-level plot and low-level behaviour of virtual humans and objects allows ones to concentrate their thinking on one problem, without being distracted by the second one. Additionally, the simulator can use a different representation than the story manager, not necessarily based on Petri Nets. For example, behaviour of virtual humans can be represented by means of hierarchical if-then rules (Bryson, 2001; Brom, 2005), or BDI (Bratman, 1987). The BDI is actually the case of Karo.

Third, there is no conflict-resolving trigger in the PNA-plot discussed: the reason for this is that conflicts were intentionally avoided during the design of the plot. Forth, all the properties that are only indicated in the portrayal, such as probabilistic distributions, must be specified precisely in the xml-file corresponding to the plot. It is the job of the programmer to create this xml-file, and not the designer's. This precise xml representation is to be created from a Petri Nets drawing of a designer and based on his or her supplementary text.

In the next two sections, we overview the modifications of Petri Nets used in Europe 2045 and Karo. In Section 7, a more general discussion follows.

5 Europe 2045

Europe 2045 is an on-line multiplayer game aimed at education of high-school students in economics, politics, and media studies (Brom et al., 2007b). The developmental phase is completed and the game is presently being evaluated. Seven preliminary studies have been already carried out, each involving about 10 high-school or undergraduate university students. A pilot evaluation has been conducted on two groups of 34 high-school students (19 females, 15 males) in Prague, the Czech Republic, in January 2008. The game is intended to be fully applied in spring 2008.

Europe 2045 contains economical and social model, which simulates population aging, migration, evolution of the market, transfers of industry and services, changes in environment, moods of citizens, and a substantial number of other variables describing

particular states and European Union as a whole (e.g. culture, infrastructure, education, etc.). The game features three layers of game-play. Each student (1) represents one EU member state in the game and is responsible for its governmental policies, economical development, and social issues. Basically, the player governs its state by deciding on the budget's priorities. Additionally, in cooperation with other players, (2) he or she is engaged in setting politics of the whole EU. Nevertheless, the essential feature of the game is that (3) each player faces various simulated scenarios and crises addressing contemporary key issues of the unified Europe, including migration, international relations, and energy independence.

The narrative structure of Europe 2045 serves for three purposes. First, it introduces new topics and agenda for students' discussions. Second, unfolding new events in accordance with players' previous decisions, it serves as a global feedback for the students and as a method for sharpening the discussion. Both these kinds of events are global, i.e. they are common for all the players and concern EU and international issues (e.g. conflict in Darfur has intensified). The third class of events provides individual players with a feedback about the results of their previous actions concerning their own states; hence, these events are local (e.g. citizens in France protest against university fees, or unemployment in Czech Republic has reached 15%).

The game proceeds in rounds, one round is one game year. An atomic "beat" of a scenario is called an *affair*. It is an event that takes place in one round and can be triggered by players' actions or results from the economical and social model or affairs from previous rounds. An affair is communicated to the player via a textual description in the game newspaper or via a short animation in TV, which is being displayed at the beginning of every round (*news item*). In some cases, an affair also has an impact on the economical and social model, i.e. it influences state of a country or the whole EU. Typically, an affair can result in increasing the EU budget, increasing the level of pollution in particular states, crippling agriculture production, etc.

Some affairs introduce issues that require decision to be taken by the players (e.g. accepting another state's proposal, sending humanitarian mission to the area of a conflict, etc.). These decisions are intended to be taken during a discussion, typically in the class under the teacher's supervision, and voted through a *ballot*. One affair often triggers more ballots, each constituting precisely formulated question ("Do you vote for sending European humanitarian mission to Darfur area?") with three possible answers (yes/no/abstain). The ballots chosen by the game designers aim to cover all the main possible solutions usually proposed by real politics in similar cases. When the answers can not be schematised to the yes or no option, the ballot contains number (3-4) of more detailed solutions. The decision chosen by the players influences the economical and social model and the affairs to be triggered in the next round.

The game offers more different campaigns to be played, each of them focusing on different problematic (e.g. energy independence, international relations, environment). For each campaign, specific affairs and a scenario describing relations between them have to be designed. New campaign also comprises distinctive animations for the TV, articles for the newspaper, items for the in-game encyclopaedia, teachers manual, and handouts for students.

The aim of this section is to describe how Petri Nets were modified for the storytelling purposes in Europe 2045, i.e. to introduce the PNB-model. Also an example of a particular plot (i.e. a PNB-plot) concerning with "Darfur crisis" scenario will be introduced.

Figure 6: Screenshots from Europe 2045. left) The interface, through which the player governs its country. right) The balloting interface.

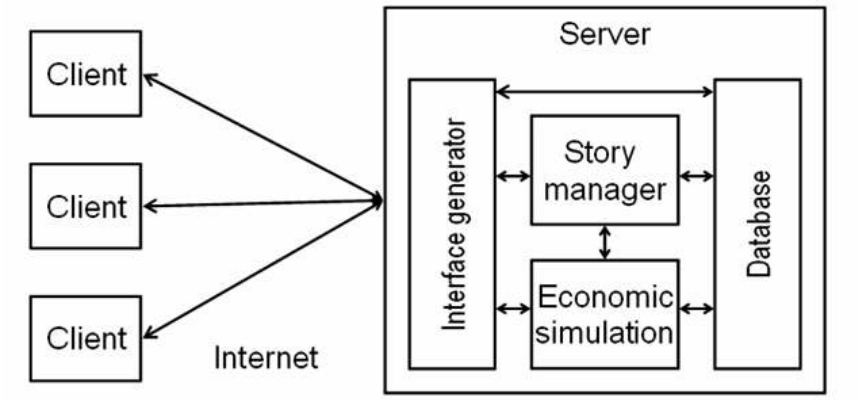

Figure 7: Architecture of Europe 2045. Notice, how it mimics the abstract architecture from Fig. 2.

5.1 Technical background of Europe 2045

Technically, the game is a client-server application; the students play the game via the Internet (Fig. 7). The server part comprises PHP scripts generating the game interface, the story manager, which is written in PHP as well, and the social-economical simulation, which is written in Java. Almost all parts of the interface are programmed in Flash (see Fig. 6). The social-economical simulation features a simplified model of EU economy. Basically, at the end of each round, this component computes the next state of each country (i.e. will the quality of infrastructure increase or decrease?), and carries out decisions in which country to build new factories, mines etc. based on particular variables of the countries (e.g. environmental tax, quality of environment, how well the infrastructure is developed etc.). We remark that this simulation does not feature human-like agents.

5.2 PNB-model

The PNB-model is stemming from the PNA-model. It was designed to mirror the features of stories of Europe 2045 described above. Particularly, the model must have seized the round-based nature of the game, the affairs, the ballots, and the presentation of news items. We now describe individual components of the model. Then, we describe how they are integrated together and introduce the story manager algorithm.

Actions. The model works with two types of actions: *affairs* (▭), and *news items* (⋯). Both of them can be started between two rounds by a trigger, as described later. When an affair is triggered, it can influence the game by its *game impact* (this is similar to an effect of an action of the PNA-model). An impact can be for example: "migration to EU decreases in this round by x". The result of the impact is computed by the economic simulator before the next round starts (see Algorithm 2). Contrary to affairs, news items never have a game impact.

Both kinds of actions can communicate to the player several news items in the next round, using either the game newspaper, or the game TV. Only an affair can invoke (immediately) one or more ballots about a proposal related to the affair.

When the next round finishes, both affairs and news items can generate a new token to a specific container (this was a job for token-generating triggers in the PNA-model; both approaches are equivalent). Fig. 8a depicts an action, which generates one token. In Fig. 8b, there is depicted an affair "Darfur conflict" invoking a ballot about "sending an EU mission to Darfur" proposal (which in case of agreement generates two tokens, each to a particular container – see next).

Ballots. In a typical *ballot* (⬭), each student has three possibilities: he or she can agree, disagree or abstain, but more complicated cases are also allowed. To determine how a result of a ballot influences the game (i.e. which game impact to apply), a *what-next function* is defined for each ballot. Basically, for each return value of a what-next function we define a *game impact* (this is again similar to an effect of an action of the PNA-model). Additionally, every ballot can generate one or more tokens similarly to actions based on on the result of the what-next function. More detailed description on what-next function element can be found in Brom et al. (2007b).

Tokens. We employ state-less aging tokens, in contrast to the PNA-model. A token starts to age after it is generated to a container, but not removed in the next round. The age is given in the number of rounds the token stays in the container.

Triggers & Containers. Triggers are evaluated at the beginning of each round. The PNB-model features three types of them. First, we have *token-generating* triggers. These are associated with containers, but not with actions as in the case of the PNA-model. Every time the condition of such a trigger holds, a new token is generated into the container. In Europe 2045, we use only the form of "when the round x starts $\rightarrow$ generate one token". In fact, these triggers start the game or schedule the episodes to particular rounds. A container with a trigger is depicted as ◎, while a container without a trigger as ○.

Second type of triggers is an *action-starting* trigger, that joins job of action-firing and token-removing triggers of the PNA-model (this was regarded as more intuitive for designers). As we already know, a conflict between two actions may occur (Fig. 8c). Hence, this trigger only marks its action to be started, and the respective tokens to be removed. A typical action-starting trigger has the following form: "if there are n tokens in a particular container & f $\rightarrow$ mark the desired action", where f is a function of game state returning a boolean value (e.g. f can be "is EU budget in this round bigger then x EUR?").

To cope with conflicts, we define *conflict resolving* triggers that unmark one of the marked actions in the same manner as in the PNA-model. In fact, they have not been used in real scenarios, remaining a mechanism making the PNB-model theoretically coherent.

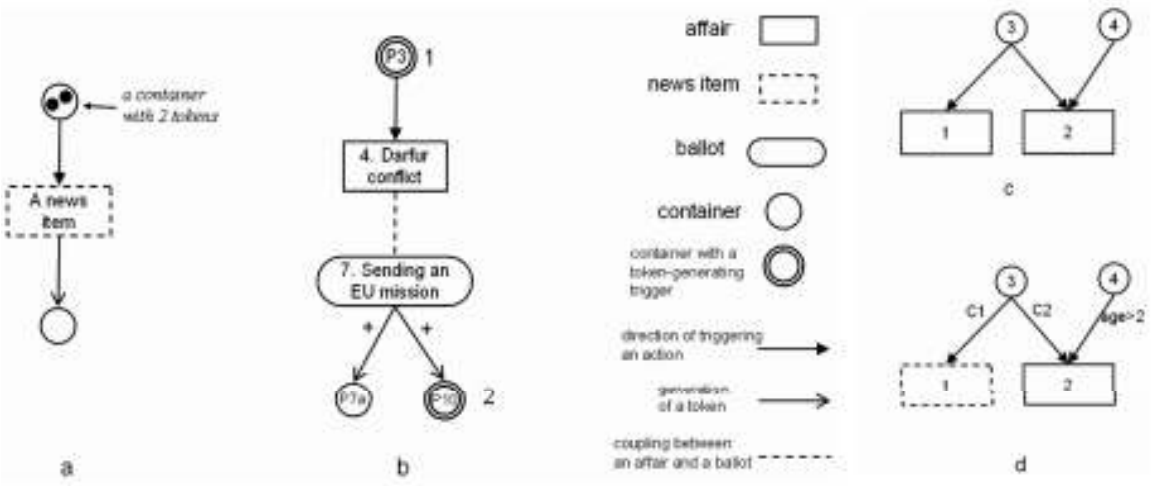

Figure 8: a) A simple net. One token is needed in the upper container to start the news item, which then generates a token to the bottom container. b) A token-generating trigger generates a token to Container P3 in the first round (notice 1 next to P3). In fact, this starts the scenario. Affair 4 invokes Ballot 7 immediately. Between round 1 and 2 and if the what-next function of Ballot 7 returns "+" (which is in this case when the ballot proposal has been agreed), two tokens are generated; one to P7a, one to P10. Additionally, a token-generating trigger generates one token into P10 in the second round regardless of the result of the ballot. c) If there is both one token in Container 3 and one token in 4, it is not clear whether to start Action 1 or 2. A conflict resolving trigger must be invoked. d) News item 1 is started when Condition C1 holds. Affair 2 is started when condition C2 holds and there is at least one token in Container 4 older than 2 rounds. Note, that in fact, the overall condition of trigger starting Affair 2 is "C2 & *age_of_token_in_Cont4* $>$ *2*". If there is a conflict between this trigger and the trigger starting News Item 1, the conflict-resolving trigger is invoked as in the case (c).

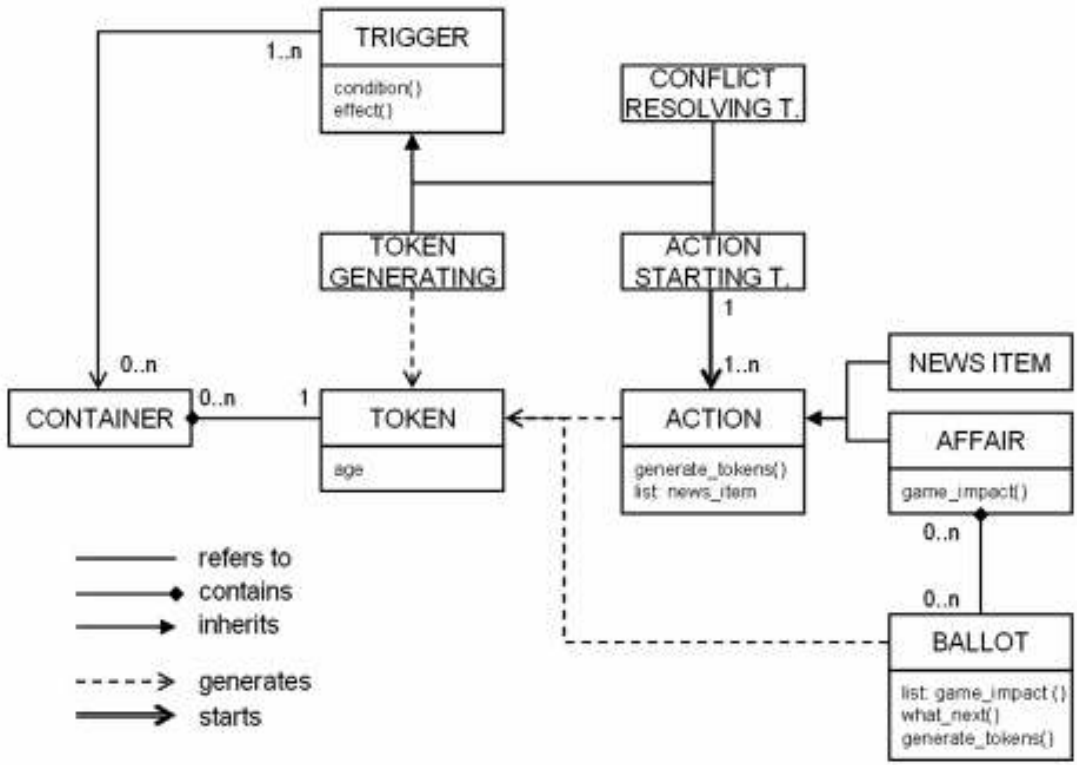

Figure 9: Overall description of PNB-model given in a UML-like class diagram. Based on this schema, the story manager in Europe 2045 works. Relations of the "conflict resolving trigger" are not depicted for clarity.

To sum up; in the PNB-model, we use tokens with aging, but without states (contrary to PNA-model), we use two types of actions (news items and affairs), and we can couple an affair with one or more ballots. We use three types of triggers, which can have relatively complicated conditions. Tokens can be generated by ballots, actions, and token-generating trigger.

A game designer specifies the plots graphically at the first place. Additionally, most entities of the model have a unique ID, which serves as a referencing mechanism to additional textual description (concerning e.g. conditions of triggers) as well as corresponding animations and news articles.

During a game, a particular PNB-plot is evaluated between each two consecutive rounds using Algorithm 2. Theoretically, some difficulties may be encountered in marking of tokens in Step 2 (for each trigger, we need to mark the tokens of appropriate age that has not been already marked, which may require searching for an appropriate ordering of triggers). Additionally, ordering of actions in Step 4b may matter (the total impact of "action A after action B" can differ from the total impact of "action B after action A"). Though these issues are of theoretical interest, we have not tackled them rigorously for they could be addressed easily in an *ad hoc* manner even for our largest scenarios (Fig. 10).

Fig. 9 overviews how all the components of the model are integrated together.

5.3 PNB-plot example: Darfur crisis

One of the largest and perhaps the most informative plots of Europe 2045 is Darfur scenario (Fig. 10). For brevity, we will not describe here the whole plot, rather we will illustrate how the building blocks of the PNB-model work on the plot.

Algorithm 2 The algorithm of the story manager in Europe 2045.

Notice, that the steps 1-5 are concerned with a round k, while 7-10 with the round $k + 1$.

1. Evaluate all token-generating triggers and generate tokens based on the results.
2. Evaluate all action-starting triggers and mark the respective actions to be started and the tokens to be removed.
3. For every token that is marked more than once: trigger the appropriate conflict-resolving trigger and consequently unmark one or more actions.
4. If:
   (a) there is no marked action, then end this scenario.
   (b) Otherwise, prepare all the marked actions to be run: generate newspaper & TV news based on the news items, consider the impact of the affairs, prepare the ballots of the affairs.
5. Remove all marked tokens, increase the age of remaining tokens.
6. New round:
   (a) Run new round. Display the news, and ballots, take the input from the user...
   (b) End the round.
7. Calculate the results of the ballots & their what-next functions.
8. Generate new tokens based on the run actions and on the what-next functions of the ballots.
9. Compute the game impacts of the ballots based in their what-next function.
10. Go to 1

This scenario starts in the first round by an affair communicating via TV that the crisis in Darfur has escalated and invoking four ballot proposals. Based on the results of the ballots, the crises further develop. The important point is that it can evolve in several branches at the same time. For example, if students agree both on a form of development aid (Ballot 8) and a humanitarian aid (Ballot 5), both the affairs "Development aid begins" (8a) as well as "Humanitarian aid begins" (5a) are triggered in the second round. Additionally, either Affair 10a "Big migration increase", or Affair 10b "Small migration increase" is started. Which affair is started depends on conditions 7A and 7B.

Notice that all elements have their ID referring for further textual description. For example, it is specified in this way that Condition 7A, i.e. the condition of the trigger starting Action 10a "Big migration increase", is "if there are less than 3 tokens in P10", while the trigger starting 10b wants "at least 3 tokens" (Condition 7B). Notice also, that Container P10 has a token-generating trigger, which generates one token into P10 in the second round. This means that even if all the proposals are disagreed, large increase in migration to EU still occurs.

Similarly, it is specified textually that in the ballot "Lost reaction" (12), students have three possibilities: to reinforce the mission, to pull out the mission or to ask NATO for help. To which container a token will be generated depends on the result of the ballot as specified by Conditions 12A, 12B, and 12C. It is also specified that Ballot 12 has no game impact, but the affair "Mission failed" (20) has the game impact: "migration to EU is increased by x".

The scenario ends in the fourth round by a final series of affairs. However, notice, that other scenarios can be unfolded in parallel. In the basic campaign of Europe 2045, there are two additional scenarios of the size of "Darfur", and several dozens of small scenarios comprising from one to three actions. The whole campaign lasts 10 rounds (game years).

5.4 Summary

We extended the preliminary model described in Section 4 by presenting the PNB-model, which was used as a plot specification technique in the full-fledged serious game Europe 2045. Presently, this game features one 10-round, fully developed campaign, which comprises about 70 game events (affairs, or news items). This allows us to conclude that in this section, we demonstrated fruitfulness of Petri Nets in the domain of social-economical serious games.

The plot of the mentioned campaign was actually coded directly in PHP. This helped us to start quickly, but it also presents a limitation. To facilitate the development process, we would benefit from a graphical authoring tool, especially because we aimed at creating a second campaign and several undergraduate humanities students, who do not know PHP, develop other campaigns as a part of their university course. Developing this tool presents our future work.

6 Karo

Karo is a simulation featuring virtual humans intended as a serious game for civic. Particularly, it should help students to understand dynamic of social relationships. Presently, it should be regarded as a research prototype, not a full-fledged application. Contrary to Europe 2045, programming work on Karo still continues. Speaking in terms of the three-layered metaphor introduced in Sec 2.1, the GUI, the simulator as well as the story manager are finished. However, only one case-study scenario (i.e. the specification of the

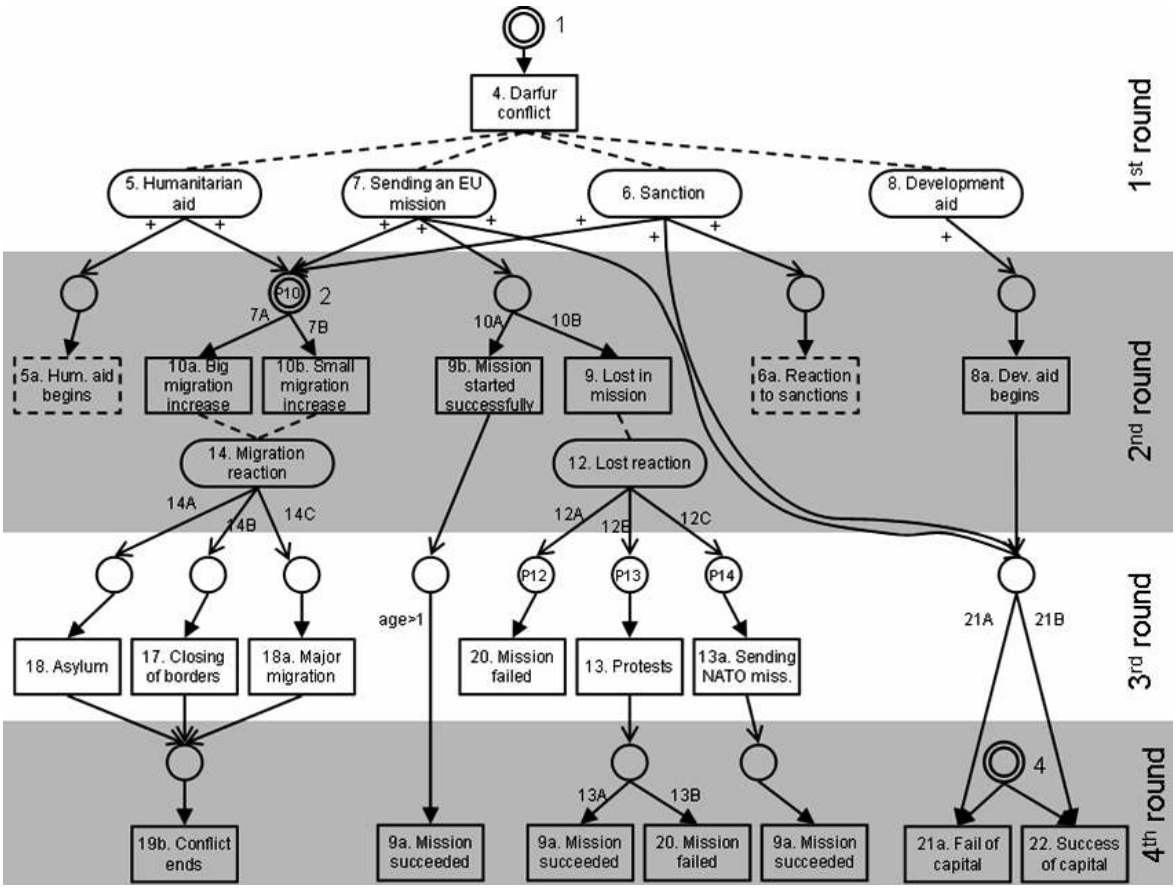

Figure 10: Darfur scenario. The elements of the plot are organised according rounds for clarity. Container's ids that are not referred from text above are not depicted. For clarity, several actions are depicted twice (9a, 20).

plot and its setting) is implemented. Moreover, this scenario has no educational aspect. Nevertheless, it showed that Petri Nets can be really scaled as a storytelling technique for a simulation employing virtual humans. After sketching technical background of Karo, we describe the case-study scenario and review the differences between the PNA-model (Sec. 4) and the Petri Nets modification used in Karo, the PNC-model.

6.1 Technical background of Karo

Technically, Karo is built upon Java framework IVE – intelligent virtual environment – we developed formerly (Brom et al., 2006), which serves as a simulator of virtual worlds. The IVE features grid worlds only; however, this has been regarded as sufficient for our purpose. The action selection mechanism of virtual humans is based on BDI architecture (Bratman, 1987), that is implemented using fuzzy if-then rules. The IVE was intentionally designed for simulations of large virtual environments inhabited by tens of virtual humans. It uses level-of-detail technique for simplifying space and behaviour of objects and virtual humans at the places unimportant at a given instant (Brom et al., 2007a).

6.2 Narrative in Karo

The narrative in Karo is partly emergent and parallel, very character-oriented and goes into the detail. Our case-study story contains many actors; some of them being background characters completely driven by underlying world simulator and never directly influencing the story (e.g. inhabitants of a village, who seem to have their own lives). On the other hand, behaviour of main characters can be influenced by the story manager.

We decided to solve "interactive–narrative" tension by limiting the degree of interactivity available to a user. We allow the user to participate only by changing emotions, moods, and relationships of the characters, and by modifying the environment slightly, e.g. by moving objects. Nevertheless, these small influences can change the outcome of the story radically. Our working hypothesis is that this approach has a strong educational potential because it allows a student to perceive that small, seemingly similar actions can have totally different outcome; however, this issue remains to be investigated.

The case study scenario is a dummy fantasy set taking place in a fictional medieval town, and in the surrounding rural area. It features seven main characters, and several background actors.

In the beginning, Anne Greyfox, a maiden, who is the daughter of a local baker, Bryant, is visited by her friend, Nerys Robertson. After a short conversation Anne and Nerys decide to go to the Lonely Tree, a romantic place outside the town. Possibly they are warned about the place by Bryant, which may influence their caution later.

The girls go through the town, finally coming to the house of Aunt Dawson, a good-hearted old woman. She asks them if they can take a lunch to her son Timmy, a city guard. After they agree, Jonas, Nerys' little brother, comes here. He is a spoiled child, fattish and rude. He tells the girls that he is coming with them. Nerys, depending on her mood, either ignores him or forbids him to follow them. He, however, heard where the girls are going, so if he is not allowed to follow them, he decides to go to the Lonely Tree on his own. He does not know that they are going to Timmy's watch post first, which is quite a detour, so he arrives to the Lonely Tree sooner.

Near the Lonely Tree, local villain and murderer, Rob The Robber, is hiding. When the girls arrive, they see him and naturally he decides to kill them. In the end, the girls are either alive, and Rob the Robber is dead, or other way round, and also Jonas is either alive or dead, depending on how the user was influencing the story.

6.3 PNC-model

The PNC-model is an extension of the PNA-model. Apart from minor modifications, there are two significant extensions. First, triggers can now test state of the world using a general pattern matching algorithm. This feature is actually similar to the function f introduced into the PNB-model (see Sec. 5.2, Triggers & Containers). Hence, a designer of a plot can specify a condition like "someone is near Jane" with the effect of "putting token to container X".

Second, because graphical representations of plots became quite large, larger than the ones on Fig. 4 and 10, we introduced nesting of Petri Nets. This has been achieved through adding opening and closing triggers, which based on the state of the world and on the state of an open net can open or close a subnet. A subnet can also contain opening and closing triggers, a method for having deep hierarchies. This mechanism not only simplifies design, but also speeds up evaluation, because the closed subnets are not evaluated. It also prevents a designer to do unintended mistakes by using entities that are not related with the current part of the plot.

6.4 Summary

We extended the PNA-model described in Section 4 by presenting the PNC-model, which was used for developing a case-study scenario in Karo game. The main innovation in the PNC-model is hierarchical nesting of Petri Nets.

While this work is still in progress concerning its educational side and particularly the claims that "small influences can change the outcome of the story radically" and that this "has an educational potential", the experience we gained so far allows as to state that Petri Nets can be and in fact have been scaled for unfolding stories in an application featuring virtual humans.

7 Discussion & Conclusion

In this article, we have described a method for specifying branching plots of scenarios that can evolve in parallel, i.e. in several different places at the same time. The method is specifically designed to cope with stories being unfolded in large virtual worlds. The method exploits Petri Nets, which are a technique relatively novel in the field of story-telling (but see Natkin (2003); Delmas et al. (2007)).

The article proceeded as follows: after discussing related work, we have presented a formal model for representation of plots in Section 4, so called the PNA-model, and have introduced an application that allows for prototyping of these plots and evolving stories in an abstract manner, i.e. without any virtual world simulator. Then, in Section 5 and 6, we have extended this work by presenting the PNB-model and the PNC-model, which have been used and are being used in serious games Europe 2045 and Karo respectively. While the PNB-model have demonstrated the practical usefulness of Petri Nets as a plot representational technique in the domain of serious games, the PNC-model have showed their scalability for an application featuring virtual humans.

Apart from the ability to represent branching plots that can evolve in parallel, the strength of Petri Nets is that they are formal and yet can be presented graphically. We have found them as easily comprehensible by non-IT experts; not only the chief designer of Europe 2045 does not have IT background, but the technique was also explained to college students of humanities during a course (during about an hour and half) and they

were subsequently able to use it to specify their own "toy" campaigns. (However, we have not carried out a "control test" with another technique; the HTN planning might be a good candidate.)

Since the Petri Nets have all these strengths at the same time, we favoured them over deterministic finite state machines (Sheldon, 2004), beat-approach of Mateas (2002), and HTN planning formalism (e.g., Cavazza, 2002).

The question remains, what are the limitations of the technique. At the beginning, it must be acknowledged that Petri Nets fit well only for stories that are preset. We also think that the beat approach is better for controlling stories featuring relatively small virtual worlds inhabited by (a few) virtual humans – we regard this approach as more flexible in this domain. Additionally, if one needs a story based on a preset plot, which however do not evolve in parallel, a deterministic FSM would likely be sufficient.

It must be also stressed that Petri Nets is a branching technique: after all, the author must specify all the branches in advance. The potential risk of combinatorial explosion of branches must be avoided by manual "cutting" by the author. It is intriguing to think whether this problem can be overcame, perhaps by generating parts of Petri Nets automatically, hence empowering the technique with some ideas being exploited in the planning domain (e.g., Reidl and Young, 2006). This is, however, a pie in the sky.

Acknowledgements

The research on usage of Petri Nets in storytelling was partially supported by the Program "Information Society" under project 1ET100300517, and by the Ministry of Education of the Czech Republic (Res. Project MSM0021620838). The project "Integration of IT Tools into Education of Humanities" is financed by the European Social Fund, the state budget of the Czech Republic, and by the budget of Municipal House Prague. The authors would like to thank to all the partners of the project: Generation Europe, Ciant, gymnasium Sázavská, and Association for International Affairs, and to all the people who helped, most notably to Petr, Edita, Jakub, Lenka, Ondřej, Martin, and Michal.

References

Aylett, R. S. 2000, 'Emergent Narrative, Social Immersion and Storification' in *Proceedings of 1st Narrative Interaction for Learning Environments*, Edinburgh.

Aylett R.S., Louchart S., Dias J., Paiva A., Vala M. 2005, 'FearNot! – An Experiment in Emergent Narrative' in *Proceedings of Intelligent Virtual Agents*, LNAI 3661, Springer, pp 305–316.

Bratman, M. E. 1987, *Intention, plans, and practical reason*, Cambridge, Mass: Harvard University Press.

Brom, C. 2005, 'Hierarchical Reactive Planning: Where is its limit?' in *Proceedings of Modelling Natural Action Selection*, Edinburgh, Scotland, pp 235 – 242.

Brom, C., Abonyi A. 2006, 'Petri-Nets for Game Plot' in *Proceedings of AISB Artificial Intelligence and Simulation Behaviour Convention*, Bristol, Vol. 3, pp 6–13.

Brom, C., Lukavský J., Šerý O., Poch T., Šafrata P. 2006, 'Affordances and level-of-detail AI for virtual humans', in *Proceedings of Game Set and Match 2*, The Netherlands, Delft.

Brom, C., Šerý, O., Poch, T. 2007a, 'Simulation Level of Detail for Virtual Humans' in *Proceedings of 7th International Conference on Inteligent Virtual Humans*, LNCS Vol. 4722. Paris, France. Springer-Verlag, Berlin, pp 1–14.

Brom, C., Šisler, V., Holan, T. 2007b, 'Story Manager in Europe 2045 Uses Petri Nets' in *Proc. of 4th International Conference on Virtual Sotrytelling*, Springer-Verlag (to appear).

Bryson, J. 2001, *Intelligence by Design: Principles of Modularity and Coordination for Engineering Complex Adaptive Agents*, PhD thesis, Massachusetts Institute of Technology.

Cavazza M., Charles F., Mead S. J. 2002, 'Planning Characters' Behaviour in Interactive Storytelling' in *The Journal of Visualization and Computer Animation*, Vol. 13, pp 121–131.

de Freitas, S. 2006, 'Learning in Immersive worlds: A review of game-based learning', *JISC (Joint Informational Systems Committee) report*, [Online] Available at: `http://www.jisc.ac.uk/eli_outcomes.html`

Delmas, G., Champagnat, R., Augeraud, M. 2007, 'Plot Monitoring for Interactive Narrative Games' in *Proc. of ACE 07*, Salzburg, Austria, pp 17–20.

The Education Arcade: Revolution, a role-playing game 2007, [Online] Available at: `http://www.educationarcade.org/revolution`

Egenfeldt-Nielsen, S. 2005, *Beyond Edutainment: Exploring the Educational Potential of Computer Games*, PhD Thesis, University of Copehagen.

Egenfeldt-Nielsen, S., Buch, T. 2006, 'The learning effect of 'Global Conflicts: Middle East" in *Gaming Realities: A Challenge for Digital Culture*, eds. M. Santorineos, N. Dimitriadi, Athens: Fournos, pp 93–97.

Francis, R. 2008, *Revolution: Student's experiences of virtual role play within a virtual reconstruction of 18th century colonial Williamsburg*, an unpublished manuscript.

Generation Europe: Europe 2045, a multi-player strategy game 2007, [Online] Available at: `http://europe2045.eu`

Gordon, A. 2004, 'Authoring branching storylines for training applications' in *Proceedings of the 6th international conference on Learning sciences table of contents*, pp 230–237.

Louchart, S., Aylett, R. 2003, 'Solving the narrative paradox in VEs - lessons from RPGs' in *Intelligent Virtual Agents*, eds. T. Rist, R. Aylett, D. Ballin, 4th International Workshop IVA 2003, LNAI 2792 Springer 2003, ISBN: 3-540-20003-7, pp 244–248.

Magerko, B. 2006, 'Intelligent Story Direction in the Interactive Drama Architecture' in *AI Game Wisdom III*, pp 583–596.

Mateas, M. 2002, *Interactive Drama, Art and Artificial Intelligence*, Ph.D. Dissertation. Department of Computer Science, Carnegie Mellon University.

Natkin, S., Vega, L. 2003, 'Petri Net Modelling for the Analysis of the Ordering of Actions in Computer Games' in *Proceedings of Game-ON*, pp 82–92.

Petri Nets World: Petri Nets World, a web collection 2007, [Online] Available at: `http://www.informatik.uni-hamburg.de/TGI/PetriNets/`

Reidl, M. O., Stern, A. 2006, 'Believable agents and Intelligent Story Adaptation for Interactive Storytelling' in *Proceedings of TIDSE*, LNCS 4326, Darmstadt, Germany, Springer-Verlag, pp 1–12.

Riedl, M., and Young, R. M. 2006, 'From Linear Story Generation to Branching Story Graphs' in *The IEEE Journal of Computer Graphics and Applications*, May/June 2006, pp 23–31.

Sandford, R., Ulicsak, M., Facer, K., Rudd, T. 2007, *Teaching with Games. Using commercial off-the-shelf computer games in formal education*, Futurelab, Bristol, UK, [Online] Available at: `www.futurelab.org.uk/download/pdfs/research/TWG_report.pdf`

Sheldon, L. 2004, *Character Development and Storytelling*, Chapters 7 and 14, Thompson Course Technology.

Silva, A., Raimundo, G., Paiva, A. 2003, 'Tell Me That Bit Again... Bringing Interactivity to a Virtual Storyteller' in *The Proceedings of Virtual Storytelling II*, Springer-Verlag, pp 146–155.

Squire, K. 2004, *Replaying history: Learning World History through playing Civilization III*, PhD thesis, Indiana University.

Wooldridge, M. 2002, *An Introduction to MultiAgent Systems*, John Wiley & Sons.