Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Degree Course: SOFTWARE ENGINEERING

Marko Divéky

GENERATING DYNAMIC INTERACTIVE STORIES

Master Thesis

Supervisor: Professor Mária Bieliková

May 2009

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Študijný program: SOFTVÉROVÉ INŽINIERSTVO

Marko Divéky

GENEROVANIE DYNAMICKÝCH INTERAKTÍVNYCH PRÍBEHOV

Diplomová práca

Vedúca diplomovej práce: prof. Ing. Mária Bieliková, PhD.

máj, 2009

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

ZADANIE DIPLOMOVEJ PRÁCE

Meno študenta: **Bc. Marko Divéky**
Študijný odbor: SOFTVÉROVÉ INŽINIERSTVO
Študijný program: Softvérové inžinierstvo
Názov projektu: **Generovanie dynamických interaktívnych príbehov**

Zadanie:

Interaktívne prerozprávanie príbehov (angl. interactive storytelling) predstavuje moderný prístup k rozprávaniu príbehov pomocou počítačov. Tento prístup je založený na „vtiahnutí" poslucháča priamo do deja prerozprávaného príbehu ako hlavnú postavu, čím, na rozdiel od klasického (neinteraktívneho) prerozprávania (akým je napr. film či kniha), umožní poslucháčovi priamym spôsobom zasahovať do deja príbehu. Použitie interaktívneho prerozprávania príbehov nasleduje smer prechodu od statického k dynamickému, od jednotného pre všetkých k personalizovanému. Analyzujte súčasné prístupy v oblasti interaktívneho prerozprávania príbehov. Preskúmajte možnosti ich aplikovania na moderné počítačové hry, ktoré disponujú prakticky neobmedzenou mierou interaktivity. Pre najvhodnejší žáner hier navrhnite spôsob generovania dynamických interaktívnych príbehov. Vytvorte softvérový prototyp pokrývajúci vizualizačnú a plánovaciu úroveň navrhnutého riešenia a overte ho v rámci vybranej časti domény histórie počítania (napr. história programovacích jazykov alebo história internetového počítania či webu).

Odporúčaná literatúra:

Barros, L. M., Musse, S. R.: Planning Algorithms for Interactive Storytelling. Computers in Entertainment, Vol. 5, No. 1 (2007).

Cavazza, M., Pizzi, D.: Narratology for Interactive Storytelling: A Critical Introduction. In Proc. of the Second International Conference on Technologies for Interactive Digital Story-telling and Entertainment, Springer, Berlin (2004), 72-83.

Crawford, C.: Chris Crawford on Interactive Storytelling. New Riders Games, (2004).

Práca musí obsahovať:

- Anotáciu v slovenskom a anglickom jazyku
- Analýzu problému
- Opis riešenia
- Zhodnotenie
- Technickú dokumentáciu
- Zoznam použitej literatúry
- Výstupy celého diplomového projektu vrátane vlastnej diplomovej práce a vytvoreného softvéru (zdrojového kódu s dokumentáciou)

Miesto vypracovania: Ústav informatiky a softvérového inžinierstva, FIIT STU, Bratislava
Vedúci projektu: prof. Ing. Mária Bieliková PhD.

Termín odovzdania práce v letnom semestri: dňa 13. mája 2009

Bratislava, dňa 16. februára 2009

prof. Ing. Pavol Návrat, PhD.
riaditeľ ÚISI

LICENČNÁ ZMLUVA O POUŽITÍ ŠKOLSKÉHO DIELA

uzatvorená

podľa § 40 a nasl. zákona č. 618/2003 *Z. z.* o autorskom práve a právach súvisiacich s autorským právom (autorský zákon) v znení neskorších zmien a doplnení a § 51 školské dielo

medzi

Autorom:

meno a priezvisko: **Divéky Marko, Bc.**

ID študenta: 20442 Dátum a miesto narodenia: 29. 08. 1986, Bratislava

Trvalý pobyt: Čárskeho 1 , 84104 Bratislava

Študent fakulty : **Fakulta informatiky a informačných technológií STU, Ilkovičova 3, 842 16 Bratislava**
Stupeň štúdia[1]: ❶ ❷ ❸

Názov študijného programu: Softvérové inžinierstvo

Názov študijného odboru: Softvérové inžinierstvo

a

nadobúdateľom:

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE (ďalej STU)
Vazovova 5, 812 43 Bratislava

Zastúpená dekanom fakulty: **prof. RNDr. Ľudovít Molnár, DrSc.**

Osoba oprávnená konať: prof. Ing. Pavol Návrat, PhD.

Čl. I
Predmet zmluvy

Predmetom tejto zmluvy je udelenie súhlasu autora školského diela (ďalej aj dielo) špecifikovaného v čl. II tejto zmluvy nadobúdateľovi na použitie školského diela (ďalej len „licencia") podľa podmienok dohodnutých v tejto zmluve.

Čl. II
Určenie školského diela

1. Autor udeľuje nadobúdateľovi licenciu k tomuto školskému dielu[3]:

- ☐ **bakalárska práca**
- ☑ **diplomová práca**
- ☐ **dizertačná práca**
- ☐ **iná práca,** špecifikovaná ako

s názvom **Generovanie dynamických interaktívnych príbehov**

Zadanie práce je prílohou č. 1 tejto licenčnej zmluvy.

2. Školské dielo podľa odseku 1. bolo vytvorené jeho autorom – študentom STU, ktorá je nadobúdateľom licencie podľa tejto zmluvy, na splnenie študijné povinnosti autora vyplývajúce z jeho právneho vzťahu k nadobúdateľovi v súlade so zákonom č. 131/2002 Z. z. o vysokých školách a o zmene a doplnení niektorých zákonov v znení neskorších predpisov.

---

[3] vyznačte

3. Školské dielo podľa odseku 1 sa rozumie ako výsledný celok pozostávajúci z jednej alebo viacerých súčastí. Súčasťou sa rozumie textová časť publikovaná v papierovej a elektronickej podobe, softvér, hardvér, audiovizuálny záznam alebo akýkoľvek i podporný materiál, prostriedok pre vytvorenie školského diela.
4. Študent touto zmluvou dáva súhlas na zverejnenie svojho diela v zmysle § 17 ods. 1 písm. c/ Autorského zákona.
5. Prevzatím diela nadobúdateľom sa nadobúdateľ stáva oprávnený používať dielo v rozsahu a spôsobom uvedených v tejto zmluve.

Čl. III
Spôsob použitia školského diela a rozsah licencie

1. Autor udeľuje nadobúdateľovi súhlas na vyhotovenie digitálnej rozmnoženiny školského diela za účelom uchovávania a bibliografickej registrácie školského diela v súlade s § 8, ods. 2, písm. b) zákona č. 183/2000 Z. z. o knižniciach, o doplnení zákona Slovenskej národnej rady č. 27/1987 Zb. O štátnej pamiatkovej starostlivosti a o zmene a doplnení zákona č. 68/1997 Z. z. o Matici slovenskej v znení neskorších predpisov.
2. Autor udeľuje nadobúdateľovi licenciu na sprístupňovanie vyhotovenej digitálnej rozmnoženiny školského diela online prostredníctvom internetu bez obmedzenia, vrátane práva poskytnúť sublicenciu tretej osobe na študijné, vedecké, vzdelávacie a informačné účely.
3. Nadobúdateľ je oprávnený udeliť tretej osobe súhlas na použitie diela v rozsahu udelenej licencie.
4. Autor udeľuje nadobúdateľovi súhlas na použitie diela alebo jeho časti najmä na: vyhotovenie rozmnoženiny diela, verejné rozširovanie originálu diela alebo jeho rozmnoženiny alebo zaradenie diela do súborného diela.
5. Nadobúdateľ nie je oprávnený upravovať či inak meniť dielo či názov diela. Nadobúdateľ je oprávnený použiť dielo v súlade s jeho určením a za podmienok stanovených v tejto zmluve.
6. Licencia udelená autorom nadobúdateľovi podľa tejto zmluvy je nevýhradná, nie je dotknuté právo autora použiť dielo spôsobom, na ktorý nevýhradnú licenciu udelil a takisto nie je dotknuté právo autora udeliť licenciu tretej osobe pri rešpektovaní nároku autora podľa čl. III ods. 7 a 8 zmluvy s tým, že autor nesmie udeliť licenciu inému subjektu na taký účel, ktorý by bol v rozpore s oprávnenými záujmami školy.
7. Nadobúdateľ je oprávnený požadovať, aby mu autor diela zo získanej odmeny súvisiacej s použitím diela primerane prispel na úhradu nákladov vynaložených na vytvorenie diela, a to podľa okolností až do ich skutočnej výšky.
8. Autor udeľuje nadobúdateľovi licenciu na dobu trvania majetkových práv.
9. Nadobúdateľ môže školské dielo zverejniť a šíriť aj pod svojim menom.

Čl. IV
Odmena

1. Autor udeľuje nadobúdateľovi licenciu bezodplatne.

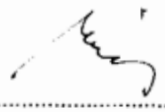
Čl. V
Pôvodnosť školského diela

1. Autor prehlasuje, že samostatnou vlastnou tvorivou činnosťou vytvoril školské dielo špecifikované v čl. II a že toto školské dielo je pôvodné.
2. Autor vyhlasuje, že pred uzavretím tejto licenčnej zmluvy neposkytol k dielu licenciu poskytovanú touto zmluvou žiadnej tretej osobe, a to ani výhradnú ani nevýhradnú.
3. Autor sa zaručuje, že všetky exemplare originálu školského diela špecifikovaného v čl. II bez ohľadu na nosič majú totožný obsah.

Čl. VI
Záverečné ustanovenia

1. Táto zmluva je vyhotovená v dvoch rovnopisoch, z ktorých po jednom vyhotovení obdržia autor a nadobúdateľ. Táto zmluva sa môže meniť alebo dopĺňať len písomným dodatkom podpísaným oboma zmluvnými stranami.
2. Táto zmluva nadobúda platnosť a účinnosť dňom jej podpisu zmluvnými stranami.
3. Na vzťahy, ktoré nie sú výslovne upravené touto zmluvou sa vzťahujú všeobecne záväzné právne predpisy platné a účinné na území Slovenskej republiky, najmä ustanovenia Autorského zákona a Občianskeho zákonníka.
4. Zmluvné strany vyhlasujú, že zmluvu uzavreli slobodne a vážne, nekonali v omyle ani v tiesni, jej obsahu porozumeli a na znak súhlasu ju vlastnoručne podpísali.

v Bratislave dňa 16. februára 2009



.....
autor

.....
nadobúdateľ

DECLARATION OF AUTHORSHIP

I, Marko Divéky, hereby confirm that I have authored this thesis independently and only with the use of resources indicated in this document.

All passages, which are literally or in general manner taken out of publications or other sources, are marked as such.

.....

ANNOTATION

Slovak University of Technology in Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: SOFTWARE ENGINEERING

Author: Bc. Marko Divéky
Master's thesis: Generating Dynamic Interactive Stories

Supervisor: Professor Mária Bieliková
2009, May

Interactive storytelling is a field of research in artificial intelligence that focuses on combining conventional stories with interactivity, resulting in immersing the reader inside stories by letting him shape the storyline in any desired direction through committing narrative actions. Despite the large amount of work that has already been done in this field, there have been only a few working solutions that found practical use other than being a proof-of-concept demonstrations. Today's popular computer role-playing games are an ideal medium for interactive storytelling, given their practically unlimited degree of interactivity and visual attractiveness. This thesis describes a new approach that aims to reduce the border between the field of interactive storytelling and modern computer games by making it possible to programmatically generate interactive stories with visually appealing computer role-playing games as the storytelling medium. Based on the proposed approach, we have implemented a software prototype in the domain of generating educational interactive stories related to the history of computing and basics of programming languages. The prototype was evaluated formally and empirically by a number of test players who filled out questionnaires covering various aspects of the generated interactive stories.

ANOTÁCIA

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: SOFTVÉROVÉ INŽINIERSTVO

Autor: Bc. Marko Divéky
Diplomová práca: Generovanie dynamických interaktívnych príbehov

Vedúca diplomovej práce: prof. Ing. Mária Bieliková, PhD.
máj, 2009

Interaktívne prerozprávanie príbehov (angl. interactive storytelling) predstavuje časť výskumu v oblasti umelej inteligencie s cieľom skombinovať klasické príbehy s interaktívnosťou takým spôsobom, ktorý poslucháča „vtiahne“ priamo do deja príbehov a umožní mu ovplyvňovať jeho priebeh ľubovoľným smerom na základe ním vykonaných akcií. Aj napriek značnému úsiliu odvedenému v tejto oblasti doposiaľ existuje iba veľmi malý počet riešení, ktoré našli praktické využitie a nezostali iba pokusnými prototypmi. V dnešnej dobe populárne počítačové hry typu role-playing games predstavujú ideálne médium pre interaktívne prerozprávanie príbehov, a to najmä vďaka prakticky neobmedzenej miere interaktívnosti a vizuálnej atraktívnosti. Táto práca opisuje inovatívny prístup, ktorého cieľom je zredukovať hranicu medzi oblasťou interaktívneho prerozprávania príbehov a modernými počítačovými hrami, a to vďaka generovaniu interaktívnych príbehov s využitím počítačových hier typu role-playing games ako médiom pre ich prerozprávanie. Na základe navrhnutého prístupu sme vytvorili softvérový prototyp, ktorý generuje interaktívne príbehy v doméne týkajúcej sa vzdelávania o histórii počítania a základoch programovacích jazykov. Výsledný prototyp bol overený formálne aj empiricky, a to skupinou testovacích hráčov, ktorí vypĺňali dotazníky s otázkami pokrývajúcimi rozličné aspekty vygenerovaných interaktívnych príbehov.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Stories and the art of storytelling have played an inevitable role in our lives ever since ancient, preliterate times. Storytelling is a very powerful and useful craft. Not only can it transport the audience on a thrilling journey into an imaginary world, but it can also transfer important knowledge and information from one generation to the next. Even in today's modern times, the educational potential of stories is widely used in many different areas.

Interactive storytelling is a new area of research in the field of artificial intelligence. Its aim is to tell stories with the use of computers in a new and interactive way, which immerses the reader inside the story as the protagonist and enables him to drive its course in any direction he desires. Interactive storytelling thus transforms conventional stories to dynamic and adaptive storyworlds with open endings.

Despite the large amount of work that has already been done in this field, there have been only a few working solutions that found practical use other than being a proof-of-concept demonstrations.

The goal of this work is to devise, prototype and evaluate a new and innovative solution to interactive storytelling that reduces the border between the field of interactive storytelling and modern computer games by making it possible to programmatically generate interactive stories with visually appealing computer role-playing games as being the storytelling medium.

This document reflects work carried out during the Diploma Project I, II and III phases and is divided into a total of eleven chapters.

Chapter 2 contains the fundamentals of both conventional stories and the field of interactive storytelling that are related to the nature of this work. Modern storytelling approaches in computer games, including a throughout analysis of computer role-playing games, the genre that is the ideal medium for storytelling, are described in chapter 3.

The analysis of the field of interactive storytelling spans chapters 4 and 5. Chapter 4 presents various approaches to interactive storytelling, including approaches used by modern solutions. Chapter 5 contains an evaluation of existing interactive story generation systems, and ends with a summary comparing and contrasting the reviewed solutions.

Chapter 6 defines goals of this work and the proposed novel approach to interactive storytelling, which is described throughout chapters 7 and 8. The architecture of the devised approach to interactive storytelling is described in chapter 7. Chapter 8 describes the process of generating interactive stories with computer role-playing games as their medium.

An overview of the implemented interactive storytelling prototype is, along with example screenshots, located in chapter 9. Chapter 10 discusses the results from both formal and empirical evaluation of the implemented prototype, whereas chapter 11 summarizes this project by describing the main contributions of the presented work.

I would like to thank my supervisor, Professor Mária Bieliková, for her valuable advice and helpful feedback. I would also like to thank everyone who helped to test and evaluate the implemented interactive storytelling prototype.

2 FROM CONVENTIONAL TO INTERACTIVE STORIES

The art of stories and storytelling has been around for centuries and is more-or-less built on the same principles as it was a thousand years ago. However, in today's modern age, storytellers have new tools available, such as computers, with the help of which new and compelling storytelling experiences can be achieved.

2.1 Interactive Storytelling

Interactive storytelling can be informally described as a modern approach to combining conventional stories with interactivity, thus transforming them from a static narrative structure to a new and compelling narrative experience, in which the reader is immersed inside the story and plays the role of the protagonist, influencing the story's direction from the very beginning to the end.

In formal words, interactive storytelling is best described as [42]:

> *"A narrative genre on computer where the user is one main character in the story and the other characters and events are automated through a program written by an author. Being a character implies choosing all narrative actions for this character."*

The initial impulse that started the research around interactive storytelling was the effort of using computers in the area of digital art, drama and literature. The first written attempts to understand interactive storytelling date back to the 1970s, whereas the 1980s delivered many key publications in this field, such as the doctoral thesis of Brenda Laurel or her latter book titled *Computers as Theater* [28].

Later on, more and more publications, papers and books appeared as the number of people involved in interactive storytelling increased gradually. It is considered the work of Brenda Laurel [28], Janet Murray [33], Chris Crawford [19],[67], Marc Cavazza and Fred Charles [68], Michael Mateas and Andrew Stern [60] that influenced the genre of interactive storytelling the most. However, there are many other people who contributed, more or less, to what interactive storytelling is today.

Since interactive storytelling is taken seriously, a number of both local and international conferences, including workshops, have taken place over the years, with new and more specialized ones appearing on a regular basis. Below is an incomplete list of conferences in common with interactive storytelling, ordered chronologically:

- Workshop on Interactive Fiction & Synthetic Realities (Boston, 1990)
- Interactive Story Systems: Plot & Character (Stanford, 1995)
- AAAI Workshop on Entertainment and AI (1996)
- First International Conference on Autonomous Agents (Marina del Rey, 1997)
- International Conference on Technologies for Interactive Digital Storytelling and Entertainment (2004, 2005, 2006)
- International Conference on Virtual Storytelling (2001, 2003, 2005, 2007)
- International Conference on Interactive Digital Storytelling (2008, 2009)

The above-mentioned conference list, especially the third and fourth conference, emphasizes the fact that interactive storytelling systems are formally classified under artificial agent systems in artificial intelligence, since they commonly utilize the concept of artificial agents (see section 4.4.2).

Over the years, people have used "interactive story," "interactive drama," "interactive narrative," "interactive fiction" and "interactive movies" [19] to describe the field of interactive storytelling[1], as depicted in Figure 1. Although these terms have much in common and principally describe the same idea, minor differences appear among them.

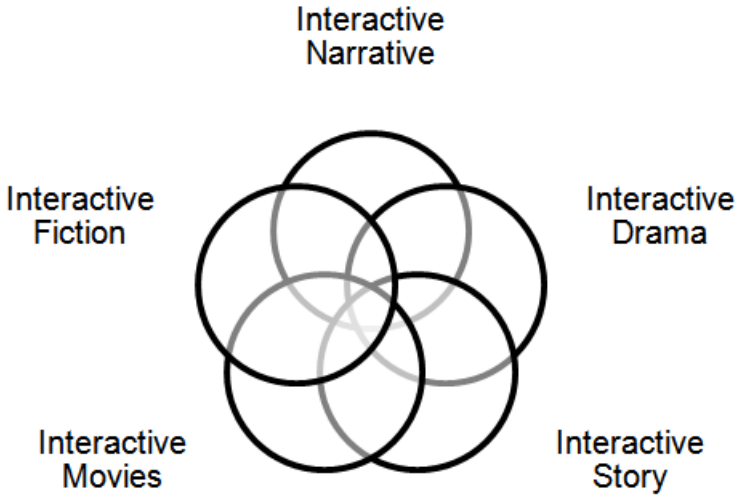


Figure 1. Fields related to interactive storytelling.

The process of combining stories with interactivity is not a trivial task. Many problems arise that need to be resolved in order to successfully achieve the goal of interactive storytelling. In order to fully understand the nature of such complications, one must have a deeper insight into the concept of stories, storytelling and interactivity itself.

2.2 Stories and Storytelling

This section contains a brief history background and a fundamental analysis of stories and storytelling necessary for the scope of this work. Some ideas and statements presented here are of Chris Crawford – a former computer game developer who dedicated a part of his life to the field of interactive storytelling [19].

2.2.1 History and Purpose

It is an undoubtable fact that stories have been part of the human experience from the earliest days of language [52],[65]. The origins of storytelling date back to ancient cultures and their languages, in which historians found first records of storytelling [65].

Anne Pellowski [35] states that storytelling had its origin in play activities, with gifted but ordinary folk entertaining their particular social group informally. Gradually, these activities were included in religious rituals, historical recitations and educational functions. Stories evolved from the human need to communicate experience and knowledge to other humans.

[1] The term "interactive storytelling" is used throughout this document because it most closely describes the nature of hereby presented work.

Moreover, Chris Crawford [19] suggests that storytelling was a natural, almost inevitable consequence of human evolution, since it was a result of interactions between the language and social mental module; two of the four most commonly recognized mental modules of the human brain – see Figure 2.

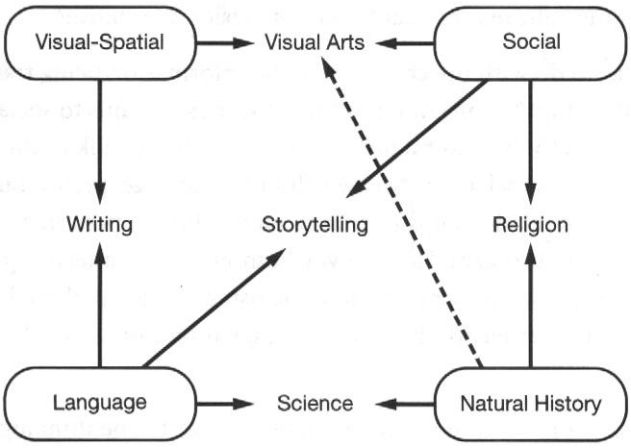
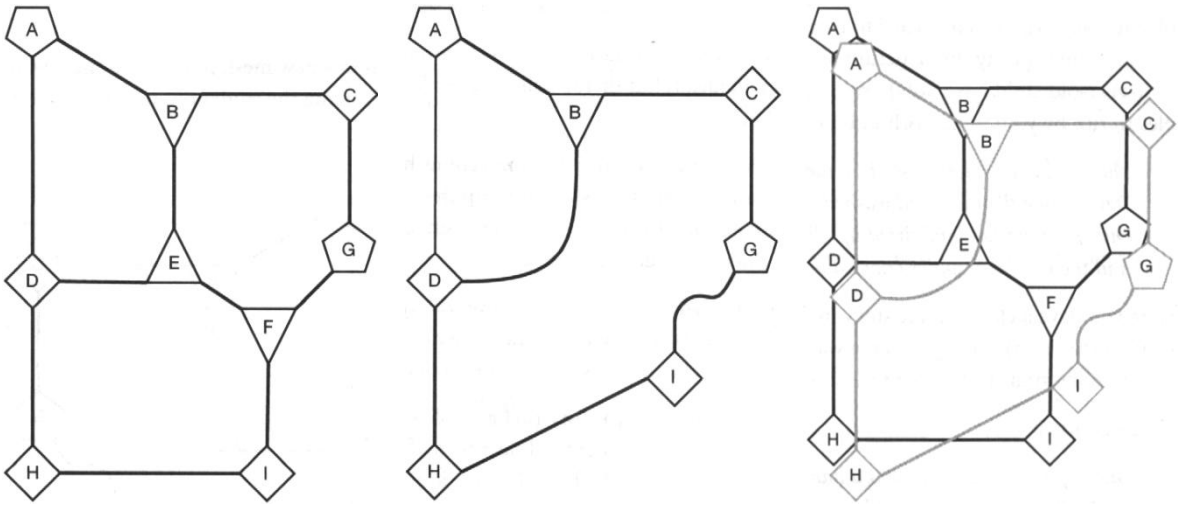


Figure 2. Interactions between the mental modules [19].

How were stories actually invented is not as important as the fact that storytelling is a vital component of our culture, since stories were, and still are, the vehicle by which cultural knowledge is communicated from one generation to the next [19]. One would argue that stories are, nowadays, not the only vehicle, however they still play an important role in transmitting information [65].

One theory that tries to answer the question why stories are so effective in transmitting information is that of Chris Crawford [19]. His theory, described below, is based on the fact that people have associative memory which can, in an conceptual way, be represented by a meshwork of interconnected ideas – as illustrated in Figure 3a.



(a) Student's mesh of ideas. **(b)** A wise mesh of ideas. **(c)** Overlay of the meshes.

Figure 3. Meshes of ideas [19].

The process of learning then consists of adding new ideas (represented by vertices in Figure 3a) and linking them with existing ones in a new and cleaner arrangement. The change of shape in the mesh is often described as "everything clicking into place" or an "Aha!" moment.

To give an example, let us suppose that a teacher wants to teach his student a set of ideas represented by a mesh in Figure 3b. The teacher must first figure out how the student's memory (mesh of ideas) looks like (e.g., Figure 3a), by asking him questions. Afterwards, the teacher must introduce new ideas to the student in such a way that correctly reorganizes his mind according to the communicated mesh of ideas (see Figure 3b).

Stories are not isolated facts, but rather a connected system of facts. Despite the fact that stories are presented in a linear form, they are perceived as a mesh of interrelated ideas. If we express the set of ideas in Figure 3b as a story, the student perceives it as a complete mesh and lays that mesh over his own, as depicted in Figure 3c.

Since the human brain is good at pattern recognition, the student's mind instantly recognizes the changes required to reorganize his own memory according to the mesh of ideas expressed as the story. Stories are therefore complete patterns that communicate a special kind of knowledge to our pattern-recognizing mental modules [19].

Regardless of whether Crawford's theory is true or not, the reason why storytelling proves to be an effective medium for educating is that the human mind is programmed much more for stories than for abstract facts[2] [40].

What is more, some of the top thinkers and knowledge management leaders all over the world consider storytelling spread knowledge amongst employees in order to help them become much more productive and collaborate with one another. For example, John Kotter, a professor at the Harvard Business School who is widely regarded as the world's foremost authority on leadership and change, made a number of noteworthy statements regarding the potential of storytelling and stories [27],[56]:

> *"Tales with a little drama are remembered far longer than any slide crammed with analytics."*
>
> *"Stories are key. If you want people to remember ideas so they can change and get better results, tell them stories."*

The educational potential of storytelling is widely utilized in business [22], where the term "business narrative" is often used as its synonym [21]. Stories are also used in many other ways: in policy, in process, in pedagogy, in critique and as a foundation and as a catalyst for change [39].

2.2.2 The Nature of Stories

Since describing the structure of stories and storytelling in detail is beyond the scope of this work, only aspects that are most important in relation to interactive storytelling are mentioned below. The principles of stories and storytelling are fully described in [31].

There has been a countless number of facts and information written about stories and storytelling. Below is a list of points, as stated by Chris Crawford [19], which are fundamental to the field of interactive storytelling:

1. **Stories are complex structures that must meet many hard-to-specify requirements.** A story cannot be easily defined as any sequence of events. Many have tried to

[2] However, this assertion is too strong to be valid in general for every domain, since there are cases when knowledge from a particular domain cannot be easily translated into a story [40].

formally define stories (see section 4.3), however no one has succeeded in describing all known and yet to be written stories.

2. **Stories are about people.** References to people are indirect or symbolic, whereas objects never play central roles, e.g., *The Lord of the Rings* by J. R. R. Tolkien [45] is not about the ring, but about the protagonist's struggle, Crichton's *Jurassic Park* [20] is not about dinosaurs, but about the conflict between a mathematician and a businessman. Moreover, all animals in Disney's *The Lion King* [47] symbolize humans and their personalities.
3. **Puzzles are not a necessary component of stories.** Puzzles often form a large part of mystery stories, but they are actually devoted to detailing the machinations in getting people to reveal crucial clues to the mystery. A story always contains some kind of problem or challenge that the protagonist needs to resolve. If this problem is an intellectual one requiring a logical solution, it is called a puzzle, but the puzzle itself is seldom central to stories.
4. **Spectacle does not make stories.** Exotic imagery as a form of entertainment dominates the movies. Many people associate spectacle with story so strongly that they believe spectacle is a necessary component of stories. However, Aristotle [2] ranked spectacle as the least important of the six elements of story (plot, character, though, diction, song and spectacle).
5. **Visual thinking should not dominate storytelling.** Many people have observed that our culture is being increasingly dominated by the image [41]. In many ways, this is good, but perceiving stories only through the visual side is not enough.
6. **Stories take place on stages, not maps.** When locations and time in which the story takes place are irrelevant to the dramatic matters addressed in the story, they are undefined. For example, Homer's *Odyssey* [23] does not say where all the places that the protagonist travelled to actually are, nor contains a map of any kind.
7. **Stories have temporal discontinuity.** Stories break up time, jump forward, backward or skip time altogether. Years are disposed of with a simple note "Many years later…" Simultaneity is presented sequentially with a phrase "Meanwhile…" What is more, characters never eat or sleep during stories, since such activities are dramatically insignificant.

2.3 Interactivity

Interactivity is a widely used term, especially in the field of computer science. Up until now, many definitions of interactivity have been introduced. It is a well-accepted assumption that interactivity will improve the entertainment and/or learning value of media [37]. This section describes interactivity from Chris Crawford's point of view [19], since it is closest to the focus of interactive storytelling.

With regard to interactive storytelling, the best definition of interactivity is as follows (see Figure 4 for a visual depiction):

> *"A cyclic process between two or more active agents in which each agent alternately listens, thinks, and speaks."*

The terms "listen," "think" and "speak" are metaphors. For example, a computer "listens" to input from its mouse and keyboard, and "speaks" by means of output on its screen. Metaphorically speaking, a computer "thinks" by processing data or running calculations.

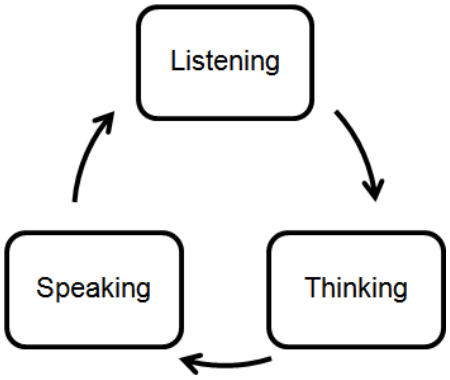


Figure 4. Interactivity as a cyclic process.

A reaction, no matter how intense, is not an interaction. If someone is watching a movie, he is not interacting with it, since the movie is not "listening" to what the viewer is "saying," nor "thinking" about anything. The movie is only "speaking." To sum up, the movie is doing all the "speaking" and the audience does all the "listening" and "thinking," what violates the definition of interactivity, stated above.

The overall quality of interactivity depends on the product, not the sum of the individual qualities of the three steps that interactivity is composed of. A good interaction must have good listening, good thinking and good speaking.

For example, if the students in a class do not "listen" to their teacher, then the overall interactivity is poor, despite how good the teacher "listens," "thinks" and "speaks" to his pupils. A conversation with someone who does not comprehend what you are saying (i.e. he is unable to "think" about the spoken topic) is considered bad conversation (thus bad interactivity), no matter how hard you try to explain what you are saying by "listening," "thinking" and "speaking" to that person. Moreover, when talking to a "deaf genius" who does perfect "listening" and "thinking," but is unable to properly "speak," it is a cumbersome conversation and bad interactivity.

Three factors determine the degree of interactivity:

- **Speed.** Considering software as an example, slow applications frustrate users, thus destroying interactivity. Fast and responsive software is great to use and interact with.
- **Depth.** Some activities require more mental exertion and hence provide deeper interaction. For example, a game of chess moves slowly, but provides deeper interaction than a game of tic-tac-toe.
- **Choice.** The quality of any interaction depends on the "richness" of choices available to the user. "Richness" breaks down into the following two factors:
  - *The functional significance of each choice:* the degree to which a choice satisfied users' desires, needs and interests. For example, a word processor could offer a feature that randomly changes fonts and font sizes while typing, but such choice would be useless (for the vast majority of users), thus providing it would not improve interaction at all.
  - *Perceived completeness:* the number of choices in relation to the number of possibilities the user can imagine. The absolute number of choices is not important; it's the number of choices offered, compared to the number of possibilities the user can imagine.

To conclude his theory, Chris Crawford introduces a rule he considers the most important building block of interactivity:

> *"Interactivity depends on the choices available to the user."*

This rule can be rephrased as:

- The choices available to the user determine the quality of the interactivity.
- If the user does not have good choices, the interactivity is bad.
- Giving the user all the right choices makes perfect interactivity.
- If the interactivity is bad, it is probably because it does not let the user make the choices they want.
- Denying choices to the user is the surest way to ruin the interaction.

2.4 Interactive Stories

Having established concepts of stories and interactivity, it is possible define what interactive storytelling and interactive stories are about, in more detail. Interactive storytelling lies at the conjunction of stories and interactivity; its goal is to combine these two distinct domains. However, there is a fundamental conflict between plot and interactivity, known as the problem of "Plot versus Interactivity" [19], which is very similar to the problem of combining determinism and free will [25].

The problem lies in the fact that a plot is a fixed sequence of events, and interactivity requires altering this predetermined sequence. In other words, if the story is to be truly interactive, the player must be able to change the story, but if the player changes the story, the author cannot control its development, and the player will likely ruin the story [19].

It is without doubt that every story is an instance that communicates a principle, i.e. the information, message or higher truth that the story's author wants to tell the audience, despite that the events of the story were often made up and never happened [19]. Moreover, every story can be told a thousand different times without altering its principle [8] (e.g., Shakespeare's *Romeo and Juliet* was transformed into a movie *Romeo + Juliet* with a completely different setting, whilst respecting the principle of the original story).

This leads to an important conclusion that *a story's principle is independent of the actual plot*, as depicted in Figure 5.

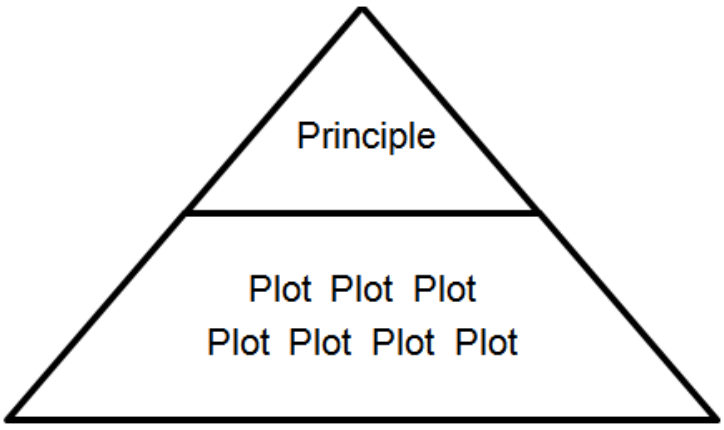


Figure 5. A story is composed of a principle independent of the actual plot.

Keeping this in mind, the solution to the above-mentioned problem of "Plot versus Interactivity" comprises moving into a higher level of abstraction by describing the story's principle with a set of rules.

That is, instead of specifying the data of the plotline, we must specify the processes of the dramatic conflict. For example, instead of defining who does what to whom, we must define how people can do various things to each other [19].

Such solution enables the creation of interactive storyworlds, an abstraction of stories that enable players to experience many different stories, all having the same principle – see below.

2.4.1 Stories versus Storyworlds

The nature of communication between the storyteller and the audience is different in conventional stories than it is in storyworlds – see Figure 6 and Figure 7.

Many aspects of stories have changed since the very first records of storytelling in ancient times. However, even in today's modern world, the fundamental idea behind conventional stories, explained below, remains unaltered.

Let us suppose that the storyteller seeks to communicate some truth or information (principle) to a desired audience. Instead of just telling the principle, the storyteller translates it into an instantiation (a story), then communicates the story to the audience which translates the instantiation back into the principle [19] – see Figure 6. This is truly a roundabout way to get the job done – but it is what works best with people, as described in section 2.2.1.

Figure 6. The process of telling a conventional story.

Interactive storytelling differs from conventional stories in a way that the process of transforming the principle into the instance (story) is delegated to the computer. Formally speaking, the computer transforms the principle into a *storyworld*, which operates on rules rather than predefines events. A single playing of a storyworld generates a single story. In other words, when a player goes through a storyworld, he produces a linear sequence of events that makes a story. Different playing of the storyworld can yield many different stories, but all of them share one common principle [19] – see Figure 7.

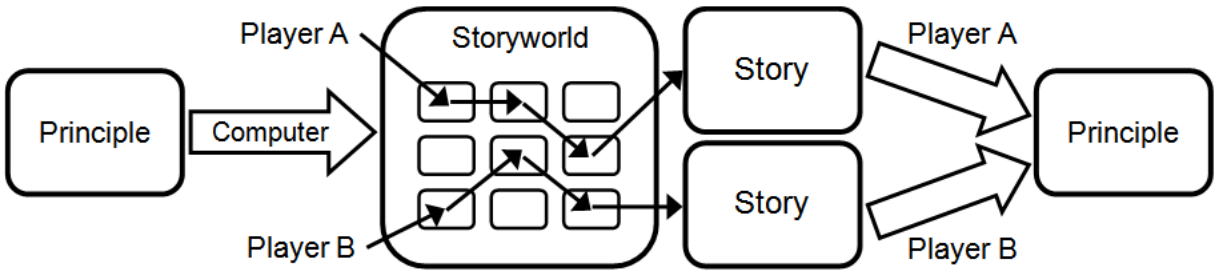


Figure 7. The process of playing through an interactive storyworld.

Because stories are generated in real time in direct response to each player's actions, the resulting story is customized to the audience's needs and interests [19], thus being a perfect target for personalization.

The above-mentioned process of creating interactive stories is, by simple means, done all the time by parents or grandparents who tell their children or grandchildren bedtime stories [19]. Exaggeratedly speaking, the goal of interactive storytelling is to automate this process with the use of computers.

2.4.2 Additional Problems

As stated above, players are given choices while playing through a storyworld. By choosing among such choices, players make dramatically interesting decisions that shape the storyline, this making the stories interactive and solving the "Plot versus Interactivity" problem. However, three major problems go hand-in-hand with such solution [19]:

- **How to generate enough interesting decisions?** The storyworld should be composed of closely balanced decisions that could reasonably go either way – unlike conventional storytelling, which gives characters decisions that can be made in only one way.
- **How to pare away the uninteresting decisions?** All consequent events that are a part of the reaction to the player's decision ought to be bundled together, advancing the story ahead to the next interesting decision.
- **How to keep the storyworld interesting?** An interesting storyworld must contain multiple, but connected themes. If the storyworld is confined to a single theme, it can develop and conclude in only a few ways.

2.4.3 Limitations

An important limitation of interactive storytelling that needs mentioning is the fact that conventional storytelling cannot be replaced by interactive storytelling, since the field of artificial intelligence is currently not close to making computers deeply understand the human nature, as it is a necessity in order for computers to triumph over humans in terms of storytelling.

Consequently, all existing interactive storytelling solutions require the storytellers, i.e. authors of storyworld to define data in the form of rules and specialized data structures that serve as input for the interactive storytelling systems, which afterwards generate interactive stories based on these restrictions. In order for interactive storytelling solutions to generate interesting stories "from nothing," computers would have to understand more intensely the processes of the human mind.

A limitation of interactive storytelling closely related to the previous statements is the absence of complex logical twists that come as a complete surprise to the audience [19].

The second limitation of interactive storytelling is the absence of real-time play, or unbroken time flow. If stories operate in real time, players might lose the opportunity to make their dramatically significant decisions, since the story will continue on without them. In other words, the storyworld must come to a halt whenever it presents a decision to the player and wait for him to respond [19].

The latter problem was, in fact, successfully solved in Façade [60], an interactive storytelling system that operates in real time – see section 5.7 for more details.

3 COMPUTER GAMES AS A STORYTELLING MEDIUM

Games have been known to the human race for thousands of years, just like stories and the art of storytelling. The field of game design predates to ancient times and is related to our ability to pretend. Pretending, i.e. creating and playing in an artificial world, is at the heart of all games [38]. However, it was not until the late 20th century with the invention of the personal computer that started the ever-growing computer game industry.

3.1 Computer Games versus Stories

Computer games were created as an art form based on the fundamental human activity of play [50], in contrast to storytelling, which originated from the need to communicate experience and knowledge among humans – as mentioned in the previous chapter. Consequently, there are many differences between computer games, stories and movies as medium for storytelling that need to be pointed out in the context of this work:

- Stories and movies are about *empathy*, since the reader *identifies himself to a character* and can only "see" the consequences of the character's actions [31].
- Computer games are focused on *immersion*, because the player *embodies a character* and can "feel" the consequences of his actions throughout the world of the game [17].

What is common to both stories and computer games is that the reader (and player) engages in a phenomenon called the *suspension of disbelief*, which refers to the willingness of a person to accept as true the premises[3] of a work of fiction, even if they are fantastic or impossible to believe [31],[48].

3.2 Storytelling Approaches in Computer Games

Although computer games and game design have their purpose and structure different from stories and storytelling, stories and narrative do play an important[4] role in computer games [58]. Below is a list of five distinct ways of approaching storytelling in games, as summarized by Spector [66]:

- **Rollercoaster storytelling** uses a predetermined narrative, which players traverse in a linear manner without having the option to make any narrative actions that have an impact on the course of the story.
- **Retold stories** are abstract games with no story at all, other than the highly individual, remembered narrative recounted by each player after having played through such games.
- **Sandbox storytelling** provides tools created by the developers for players to construct their own stories. There is no predetermined, overarching narrative.
- **Shared authorship** lays at the conjunction between sandbox and rollercoaster storytelling by giving the player some freedom while having predetermined goals. The player decides the order in which he achieves the goals or he ignores them altogether.

[3] Suspension of disbelief also refers to the willingness of the audience to overlook the limitations of a medium, in order for them not to interfere with the acceptance of the fiction's premises [16],[25].

[4] One could argue that many modern computer games have a weak story, if any. This is a consequence of the fact that for the past decade, the gaming industry has been producing "endless iterations of the same ideas" [58] with the story stagnating and only the visual aspect of games advancing. Many respected game designers are well aware of the fact and are trying their best to overcome this problem [66].

- **Procedural story generation** aims to give players complete freedom to explore game worlds and develop interpersonal relationships with game characters through making narrative actions.

So far, computer games have followed the *rollercoaster* or *sandbox storytelling* approach, or a mixture of the two [66]. The last approach to which Spector refers, *procedural story generation*, is in fact interactive storytelling, as described in the previous chapter.

3.3 Computer Games and Interactive Storytelling

Researchers in the field of interactive storytelling often disregard computer games as a suitable storytelling medium [19]. Still, there are some that see them as the ideal form for storytelling, such as Janet Murray, who writes that computer games are "a new medium of expression [that] allows us to tell stories we could not tell before, to retell the age-old stories in new ways" [32]. To support her opinion, Murray states that computer games are "a medium that includes still images, moving images, text, audio, three-dimensional, navigable space – more of the building blocks of storytelling than any single medium has ever offered us" [32].

The storytelling potential of computer games differs from one genre to the next. Throughout the many diverse genres of computer games available today, role-playing games (RPGs) and adventure games[5] have the most detailed and involving storyline, thus being the most appropriate genre for storytelling [29],[38]. Figure 8 shows the importance of storytelling and narrative in genres of computer games.

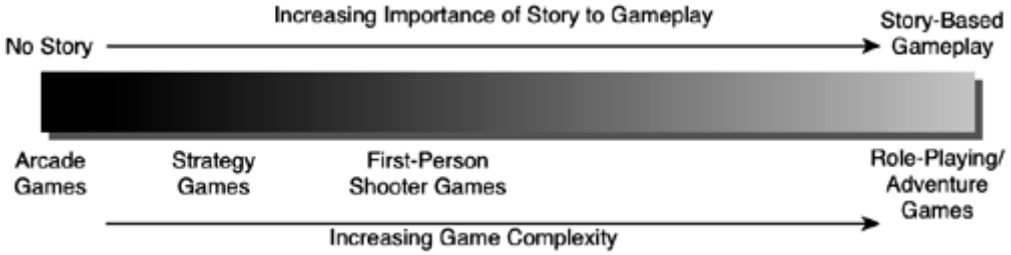


Figure 8. The story spectrum [38].

Below is a throughout analysis of aspects regarding computer role-playing games that are important for the scope of this work.

3.4 Analysis of Computer Role-playing Games

Computer role-playing games have originally derived from tabletop role-playing games and are considered as being one of today's most popular genres of computer games [1].

Themes. Games based on the role-playing genre are most often set in a fictional fantasy world closely related to classic mythology, or in a science-fiction world set somewhere in the future. On the other hand, there are a number of role-playing games set in historical or modern settings [4].

Avatars. In role-playing games, a player controls one in-game character called the *avatar*, and uses him as an instrument for interacting with the world that the game takes place in [7].

[5] Role-playing games were chosen in favor of adventure games mainly due to the increasing popularity of their online multiplayer versions, Massively Multiplayer Online Role-playing Games (MMORPGs) [1].

Some role-playing games also enable the player to control a group of characters besides the player's own avatar [46].

Character Development. Besides having the option to fully customize the appearance of his in-game character, the player is allowed to choose various attributes, skills, traits and special abilities that his avatar will possess – see Figure 9. These are given to players as rewards for overcoming challenges and achieving goals, most commonly for completing *quests*. Character development plays, together with stories, a key role in role-playing games [4].

Figure 9. An example of the player's avatar appearance and major skills [6].

Quests. A *quest* in role-playing games can be defined as a journey across the game world in which the player collects items and talks to *non-player characters* "in order to overcome challenges and achieve a meaningful goal" [24]. Quests often require the player to find specific items that he needs to correctly use or combine in order to solve a particular task, and/or require the player to choose a correct answer from a number of given answers to a certain question [7]. Upon solving a quest, the player is often presented with a reward, which can have many forms – i.e. ranging from a valuable item to a new skill, trait or ability for the player's avatar.

Many quests are optional, allowing for freedom of choice in defining the player's goals and intentions. Moreover, a set of quests may be mutually exclusive with another set, therefore forcing the player to choose which set of quests he will solve, having in mind the possible long-term effects these quests will have on the game world. Some quests can be solved in more than just one way and thus bring non-linearity into the game. Quests can also be linked together to form *quest chains*, i.e. groups of quests that the player can only complete in sequence [24].

What is noteworthy and important to realize with regard to the focus of this work is the fact that *quests are a fundamental structure by which the player moves the storyline forward in computer role-playing games*. In other words, a quest is a conceptual bridge between the open

structure of role-playing games and the closed structure of stories [26], thus being an ideal vehicle for interactive storytelling in computer role-playing games. Consequently, *it is possible to use computer role-playing games as a medium for interactive storytelling by dynamically generating non-linear quests*.

Non-player Characters. Role-playing game worlds are populated by *non-player characters* (NPCs) that cannot be controlled by the player [7]. Instead, their behavior is scripted by the game designers and executed by the game engine. Players interact with non-player characters through dialogue – see Figure 10.

Most role-playing games feature branching dialogue (or *dialogue trees*). As a result, when talking to a non-player character, the player may choose from a list of dialogue options where each choice often results in a different reaction. Such choices may affect the player's course of the game, as well as latter conversations with non-player characters. In other words, key non-player characters "remember" witnessed actions and dialogue responses previously committed and said to them by the player, and shape their relationship with him based on their memories [4].

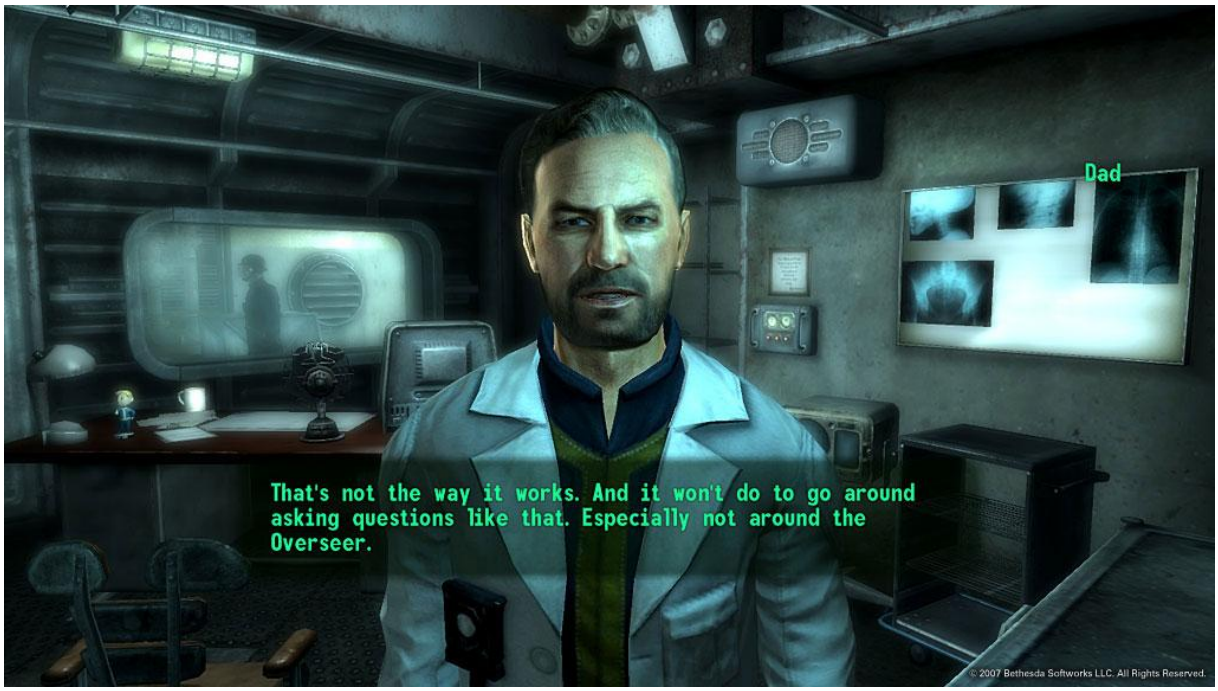


Figure 10. An example of a dialogue with a non-player character [5].

Items, Containers and Objects. Throughout playing role-playing games, quests require the player to find, collect and properly use various *items* scattered throughout the game world. Special items may be equipped on the player's avatar, improving his abilities, skills or other attributes. All items can be either created from other items, or obtained from *containers*, i.e. *objects* that can hold or carry items. The player himself is an example of a container, since he can carry items in his *inventory* – see Figure 11). Other typical examples of containers include non-player characters and objects representing treasure chests.

Figure 11. An example of a player's inventory [6].

Figure 12 contains an UML class diagram showing the basic semantic relationships between the aforementioned core elements of computer role-playing games.

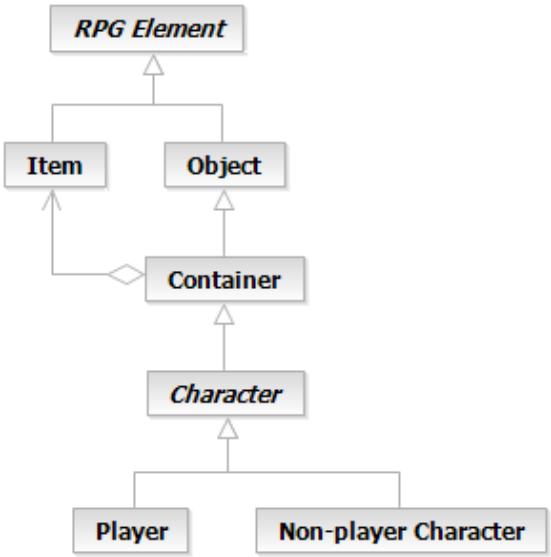


Figure 12. Semantic relationships between core elements of computer role-playing games.

To summarize, computer role-playing games provide an ideal medium for interactive storytelling, mainly due to their attractiveness, popularity and an impressive degree of interactivity.

4 APPROACHES TO INTERACTIVE STORYTELLING

Many different approaches to the generation of interactive stories have been tried throughout the years that people have been experimenting in the field interactive storytelling. This chapter addresses a number of them in more detail.

4.1 Simple Strategies

The *branching tree structure* [19] is considered the simplest strategy that can be used in interactive storytelling. Stories based on this structure start with a beginning and then attach a first choice to it, another choice to the first choice, and so on. The resulting story forms a tree of choices, as shown in Figure 13.

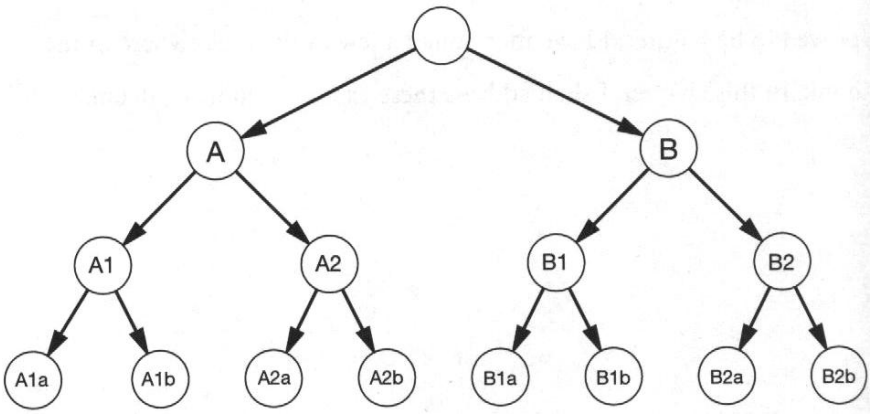


Figure 13. A simple branching tree structure [4].

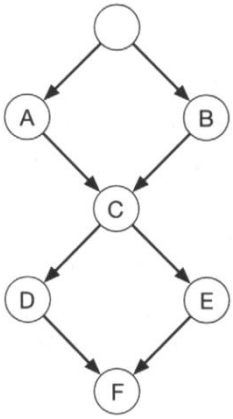


Figure 14. A simple foldback structure [4].

The problem with such approach to interactive storytelling is that the amount of paths the player is able to take grows exponentially with the number of available choices. Keeping in mind that real stories have a wide array of decisions, the usage of branching trees becomes unmanageable due to heavy workload involved in managing such a vast tree of choices [19].

Another simple strategy, called the *foldback scheme* [19], is based on a slight modification to the branching tree design. By rerouting the consequences of decisions and folding the storyline back to some predetermined path (see Figure 14), the problem of exponential growth disappears.

The problem of foldback themes is that they trick players of thinking that they made a choice, because all choices ultimately lead to the same state of the story (foldback), resulting in an unpleasant experience for players [19].

4.2 Emergent Stories

Another concept on how to generate interactive stories is the idea to use *emergent systems*. Research has shown that sufficiently complex systems have the potential to generate even more complex phenomena that the system's creators never expected.

On the one hand, according to Crawford [19] there are no known examples when such systems were able to produce emergent stories. On the other hand, the *character-centric approach* to interactive storytelling (described in section 4.4.2) shows some signs of emergence.

4.3 Narrative Theories

A number of existing interactive storytelling systems utilizes various *narrative theories* for knowledge acquisition. Below is a list of most frequently cited traditional narrative theories, most of which have been developed in the course of the 20th century [13],[19].

4.3.1 Aristotle and the Foundations of Drama Theory

Aristotle provided the earliest analysis of what became known as traditional drama, insisting in particular on its progression through climax and the final resolution. Aristotle's work provides a model for story progression that encompasses important aesthetic properties of the story. On the other hand, the model's descriptive power is not sufficient to be considered as a narrative formalism, since it does not include a fine-grained description, or even a proper formalization, of narrative actions.

4.3.2 The Aarne-Thompson Catalogues

A hundred years ago, Antti Aarne prepared an index of the various types of folktales and their motifs. Later in the 20th century, a folklorist Stith Thompson expanded and enlarged the scope of Aarne's work, and with his second addition to Aarne's catalogue in 1961, created the AT-number system that catalogues some 2500 basic plots from which, for countless generations, European and Near Eastern storytellers have built their tales. In 2004, Hans-Jörg Uther expanded the AT system to the Aarne-Thompson-Uther (ATU) system. There are not many interactive storytelling systems based on this catalogue due to the fact that such a number of motifs would require connectivity data associated with each motif, resulting in a matrix with millions of cells.

4.3.3 Barthes and the Interpretative Codes

Roland Barthes has produced comprehensive narrative analyses of classical novels. Barthes studied both syntagmatic and paradigmatic aspects of narratives. His syntagmatic approach extends the linear sequencing of Vladimir Propp's work (see below) to give the story an actual structure, possibly opening space for choice points. Barthes' narrative theory is based on five "codes" (ACT: action, REF: reference, SYM: symbolic, SEM: semantic, HER: hermeneutic), each of which indicating how to interpret the current text segment. There are a number of interactive storytelling systems that make use of the ACT and HER codes.

4.3.4 Bremond and the Reintroduction of Characters

Claude Bremond developed a narrative theory centered on the description of character's roles. Not unlike Greimas' work (see below), Bremond' theory starts with an opposition between an "Agent" and a "Patient". A "Patient" is any character that will be influenced by the narrative actions to occur, while an "Agent" is responsible for changes in the narrative universe (which can also affect other characters as Patients, in which case there are psychological changes rather than physical changes to the world). A central aspect of Bremond's model is that it reintroduces character's psychology in a quite sophisticated manner, with characters having beliefs, motivations and goals. Bremond's model has been used, for its communicative aspects, to generate dialogue acts in interactive storytelling systems, representing influences from one character to another.

4.3.5 Greimas: *A Linguistic Perspective on Narrative Analysis*

Algirdas J. Greimas developed his contribution to narratology as an extension of his work in (natural language) semantics. He introduced what can be described as the first role-based analysis of narratives, by proposing to define and categorize characters based on their actions,

and not according to their personalities. Despite being often cited in interactive storytelling work, only a few systems have really sought their inspiration in Greimas' work.

4.3.6 Propp: *The Morphology of the Folktale*

Vladimir Propp's work is the most cited amongst researchers in interactive storytelling. Propp was the first to uncover stable structures underlying Russian folktales and to describe these structures using the first ever formalism in narratology, together with a symbolic notation. He introduced narrative functions as the basic representational unit of a narrative and concluded that there were only 31 of them in Russian folktales. Propp also concluded that all the characters could be resolved into only seven broad character types in the tales he analyzed. Propp's narrative functions can be adopted almost as a ready-to-use narrative formalism, and there have been many examples of such use in interactive storytelling systems.

4.3.7 Polti: *The Thirty-Six Dramatic Situations*

Georges Polti reduced the basic storylines of all literature and theater to a core set of 36 categories, many of which are broken down into subcategories. Moreover, each category offers basic character roles necessary to each dramatic situation. Although Polti intended his list to be universal, it does reflect the cultural context of the year 1921, when his work was published. Although Polti's work has the potential to become base for an interactive storytelling system, none have been yet implemented.

4.4 Modern Strategies

Most modern interactive storytelling systems are based on two common strategies [30]: the *plot-centric approach* (founded on explicit plot representation) and the *character-centric approach* (founded on autonomous behavior of artificial characters). Both strategies are described below.

4.4.1 Plot-Centric Approach

In the plot-centric approach to interactive storytelling (also referred to as the *author-centric approach* or the *plot-based approach*), the plot is continually being monitored and generated by the computer, whereas all characters in the story are created according to the plot's requirements. In other words, an interesting plot is developed first, and afterwards characters that make such plot work are created [19].

The part of a plot-centric interactive storytelling system that oversees story development and somehow guides it in desirable directions, is a software agent called the *drama manager* [19]. Such concept was first introduced by Brenda Laurel [28], who prepared a list of 13 abstract functions that a drama manager must be able to carry out:

1. Model the plot in progress.
2. Specify the formal characteristics of upcoming events.
3. Change the storyworld.
4. Modify the goals of actors.
5. Access proposals for future actions of actors.
6. Simulate these proposals to determine their impact on the plot.
7. Evaluate the results of the simulations.
8. Mandate future events.
9. Formulate the script for the next event.

10. Direct the actors.
11. Control its own operation.
12. Remember past events.
13. Learn from past results.

This overlong list of functions was later on reduced into three fundamental steps[6] by Chris Crawford [19], who also proposed particular methods of their execution:

1. **Listen** by monitoring the story's progress and recognize the patterns of behavior that differentiate drama from tedium. This can be accomplished by relying on overview variables, which are numbers the storytelling engine calculates to assess the storyworld's overall state.
2. **Think** by determining how the story should progress forward from a specific point. This can be achieved by using:
    a. A well-designed set of *overview variables*, each of which indicates some significant factor in story development, or a scoring system.
    b. *Dramatic templates.* The author of the storyworld (called the storybuilder) defines a large set of templates that constitute well-formed stories. The drama manager then matches the story generated so far with each template, searching for the one that best fits the story so far. That template then becomes the guide for action next executed by the drama manager.
    c. A *story grammar*[7], which is a set of rules governing the sequencing of events in a story. Such grammar provides the basis for calculating story development.
3. **Speak** by changing the storyworld in a manner that helps the story evolve in the desired direction. A number of schemes are available in order to accomplish such goal:
    a. *Environmental manipulation* by making physical alterations in the storyworld. All manner of physical constraints can be applied to force players into the intended course of action. However, players often find such alterations transparent and insulting, and thus should be used only as a last resort to save the developing story from becoming a catastrophe.
    b. *Goal injection* by instilling a new goal into a chosen character, who then influences the actions of players. Such goal should be compatible with the character's personality and should be of a temporary nature so that its effects are localized to the immediate situation.
    c. *Shifting personalities* by altering the personalities of other characters in such a way that inclines them to make decisions that influence players in the desired direction. This scheme is the most indirect.
    d. *Setting up a sequence of timed plot points* that force the story forward regardless of the player's actions. This scheme is the simplest, but it must be camouflaged properly, otherwise it will appear heavy-handed.
    e. *Dropping the fourth wall* by directly advising players to actions that result in a satisfying development of the story.

[6] These steps are based on Chris Crawford's definition of interactivity, described on page 6.
[7] Vladimir Propp's work, described in section 4.3, is often cited as an example of a story grammar.

4.4.2 Character-Centric Approach

Interactive storytelling systems built upon the character-centric approach (also referred to as the *character-based approach*) first create a set of characters with different personalities. Each character is controlled by the computer and behaves autonomously. It is the dynamic interaction between characters that generates the actual plot from a generic storyline [16], unlike the plot-centric approach, in which the plot is fully maintained by the drama manager.

In other words, each character has his own goals that he actively tries to achieve by making plans and following them. Having a number of characters with different goals and plans behaving in the same environment implies the existence of conflicts among these plans, resulting in the need of replanning, and thus creating the plot of a story [16].

Comprehensively speaking, character's behavior is implemented through various real-time search-based planning techniques [3]. However, the "top-down" planning systems controlling the characters need to be complemented with appropriate mechanisms dealing with emerging, "bottom-up" situations of narrative relevance, such as situated reasoning and action repair [9].

The most commonly used planning formalisms are Hierarchical Task Network (HTN) planning and Heuristic Search Planning (HSP) [15]. An example of a HTN representing a character's plan is depicted in Figure 15.

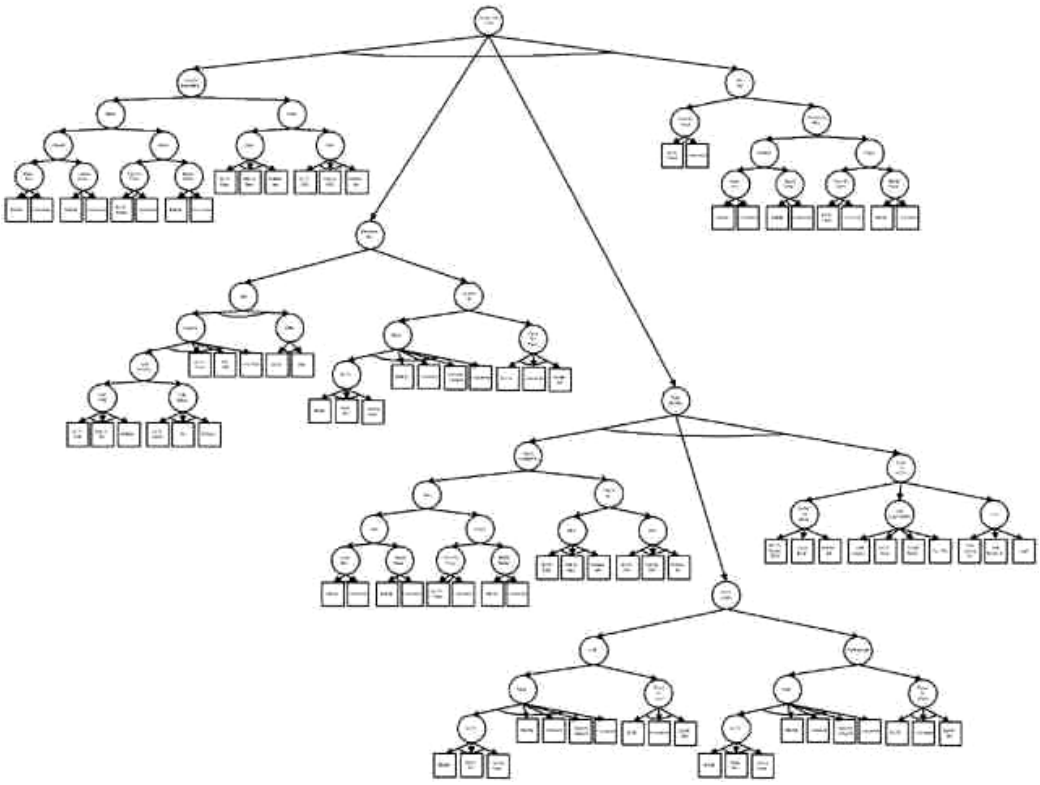


Figure 15. A character's plan represented by a HTN graph. The top of the plan contains goals, which keep being divided into subgoals, and eventually tasks that can be executed [15].

Another formalism used in this regard is a Fuzzy Cognitive Map [14], which not only concerns the relationships between the causes and effects, but also considers the relationships among the causes, and thus provides a stronger reasoning ability than HTN planning. Such approach, however, adds more complexity into the process of defining and generating stories.

5 EVALUATION OF INTERACTIVE STORYTELLING SYSTEMS

Up until now, there have been many different, more or less successful, interactive storytelling systems created. Below is a selection of such systems described in more detail.

5.1 Story Traces

Elizabeth Figa and Paul Tarau [43] at the University of North Texas have attempted to harness the power or WordNet [64] to build a library of "story traces" (skeletons of stories) and "story projections" (fragments of stories that can be treated as single dramatic atoms). The resulting library can be used for finding templates to which a developing story could be matched. This project has not reached a stage at which in can generate stories and is, as of today, abandoned.

5.2 HEFTI

The Hybrid Evolutionary-Fuzzy Time-based Interactive Storytelling engine (or HEFTI for short), made by Teong Joo Ong [34] from Texas A&M University, utilizes genetic algorithms and fuzzy logic in order to generate stories – see Figure 16 to Figure 18. The only storyworld that the author built with his system is an abbreviated version of "The Three Little Pigs."

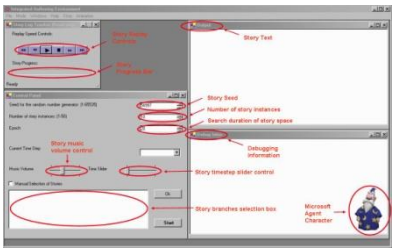


Figure 16. Authoring interface [34].

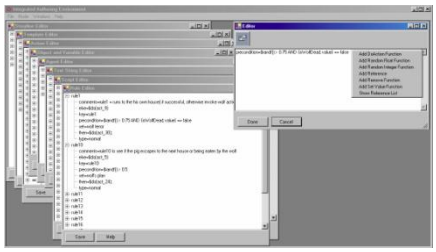


Figure 17. Reading interface [34].

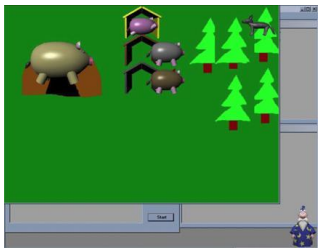


Figure 18. Visualization [34].

5.3 The Virtual Storyteller

The Virtual Storyteller [44] is a project carried out at the University of Twente in the Netherlands as a part of a much larger project call AVEIRO, which aims to present a complete virtual theater. The interactive storytelling system uses a set of autonomous agents organized by a director agent (drama manager) to populate the storyworld. The storyworld knowledge is stored in an ontology. The Virtual Storyteller is, however, unable to create interactive stories.

5.4 LOGTELL

Angelo E. M. Ciarlini, Cesar T. Pozzer, Antonio L. Furtado and Bruno Feijó from Brasil have created a tool called LOGTELL [18] that uses temporal logic and Prolog for story generation – see Figure 19 and Figure 20. This tool uses an OpenGL-based [63] 3D graphics engine created especially for visualizing the generated stories. These are, however, not interactive in any way.

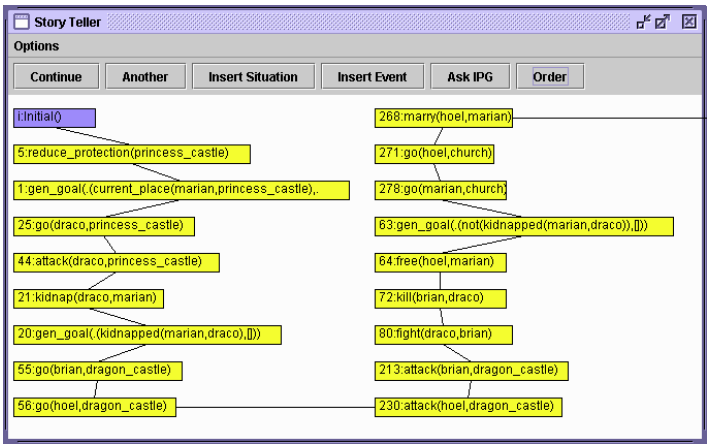


Figure 19. An example of a generated plot [18].

Figure 20. Story visualization [18].

5.5 Storytron

Chris Crawford, a former computer game developer, has been working on his interactive storytelling system called Storytron [67] (formerly called Erasmatron) since 1991. The system is based on the plot-centric approach to interactive storytelling, and thus uses a drama manager to control the plot of all created stories, which, despite being robust and flexible, are all text-based with limited visualization (see Figure 22).

There are two tools available for using the Storytron technology:

- **Storyworld Authoring Tool (Swat)** for creating storyworlds through a very rich, but rather complicated, textual format (see Figure 21).
- **The Storyteller** for playing the created storyworlds (see Figure 22). The interface shows faces of characters that the player is currently interacting with, plus a number of possible actions that the player is able to make, written in a language specially created for communicating with characters (called Deikto).

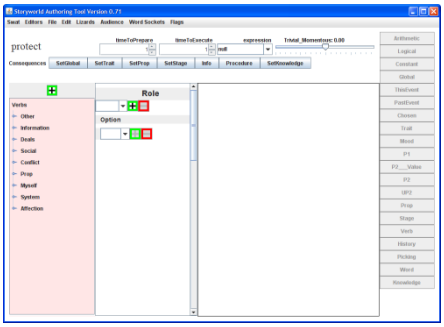


Figure 21. Swat [67].

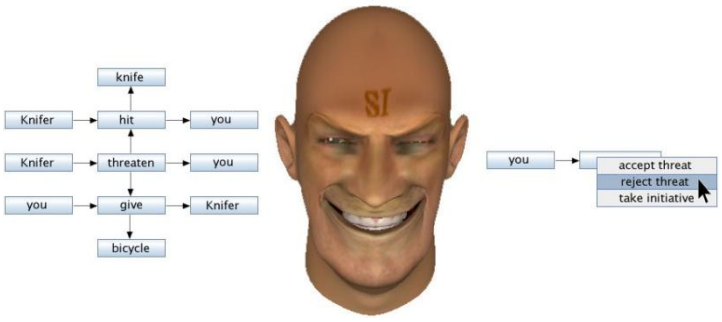


Figure 22. The Storyteller [67].

5.6 I-Storytelling

Marc Cavazza and Fred Charles at the University of Teesside [68] have developed their interactive storytelling system based on the character-centric approach (see Figure 23), with the use of Hierarchical Task Network (HTN) planning and Heuristic Search Planning (HSP) [11]. Moreover, the authors also take characters' emotions and feelings into account when generating stories [36]. However, users have only very limited ways of interacting with such stories, since they directly control no characters.

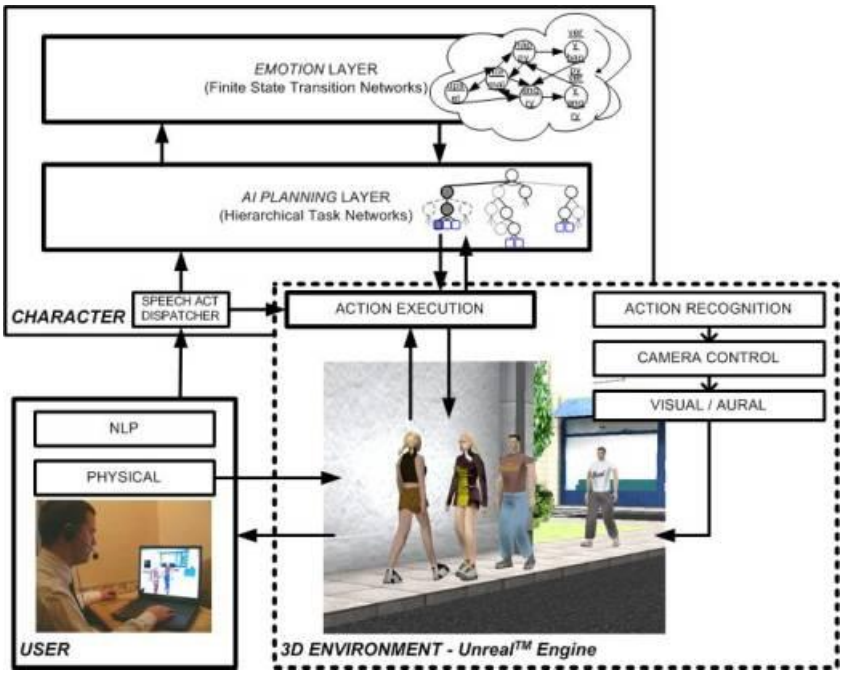


Figure 23. I-Storytelling architecture [68].

The authors have also created an immersive interactive storytelling prototype [12] that uses a CAVE-like virtual reality environment to tell stories – see Figure 24.

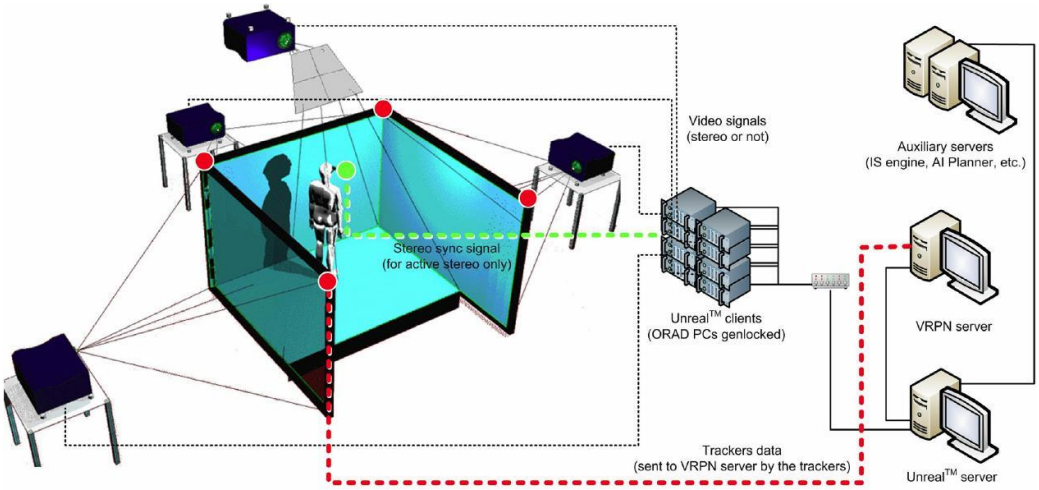


Figure 24. Hardware architecture for the PC-based CAVE-like immersive display [12].

5.7 Façade

Michael Mateas and Andrew Stern are the authors of Façade [60], an interactive drama system that is considered as being the best actual working interactive storyworld yet created – see Figure 26.

Façade uses a combination of both the plot-centric and character-centric approach to interactive storytelling, by letting characters behave autonomously and at the same time coordinating the overall plot by a drama manager that keeps choosing actions with increasing dramatic tension until the story's climax, whereafter actions with diminishing tension are chosen – see Figure 25.

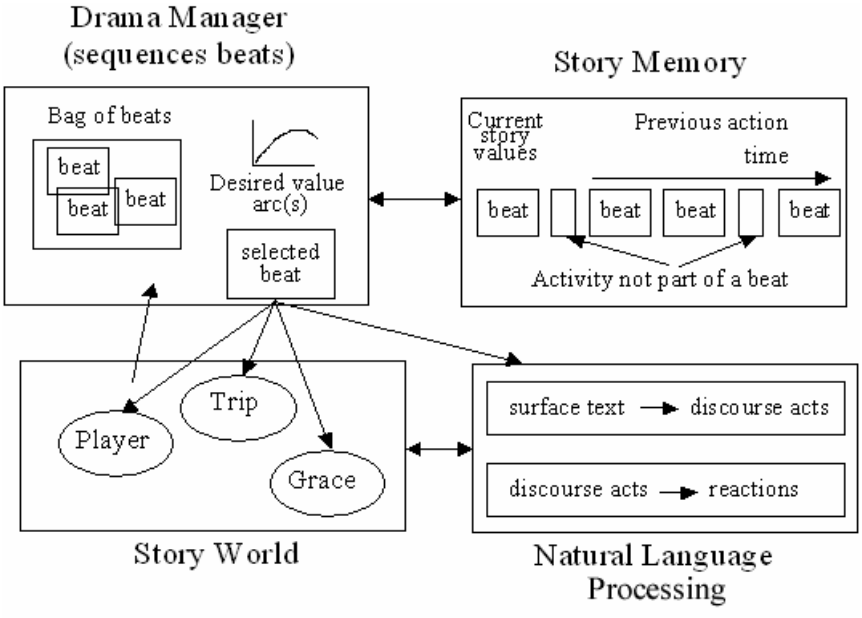


Figure 25. Façade architecture [59].

The authors have developed a 3D graphics engine that utilizes A Behavioral Language (ABL), a language for controlling bodily movements also developed by Mateas and Stern. Players have a first-person view of the scene and communicate with characters via typed-in speech, since Façade uses a Natural Language Processing (NLP) system. Façade runs in real time, in contrast to what is stated in section 2.4.3.

Figure 26. A screenshot from Façade [60].

Like other interactive storytelling systems, Façade has restrictions, too – it is able to generate only a single predefined dramatic scene in which the player visits Trip and Grace, a married couple that is experiencing a marriage crisis. By focusing their attention on a precisely defined dramatic context, Mateas and Stern were able to solve almost all problems that relate to interactive storytelling.

5.8 Summary and Comparison of Interactive Storytelling Systems

To summarize, it is possible to divide all evaluated interactive storytelling solutions (including other working solutions available today) into the following three categories:

1. **Systems that generate complex, but only text-based interactive stories.** Storytelling solutions falling into the first category are able to generate truly interactive stories with a complex and coherent plot.

   However, these systems have a very complicated and unintuitive form of defining rules and data structures according to which the resulting stories are generated, thus requiring anyone wanting to use these systems to become acquainted with the required abstract style of thinking.

   Another disadvantage of such solutions is the fact that all generated interactive stories, despite having an interesting and coherent plot, are text-based, what many people find uninteresting in today's digital age full of virtual reality and stunning 3D computer graphics.

2. **Systems that generate visually attractive, but not interactive stories.** The second category consists of storytelling solutions that generate stories visualized in either 2D or 3D computer graphics, or even in virtual reality environments.

   Despite being visually attractive, the generated stories are not interactive (according to what was defined in section 2.3), since users have only a limited number of ways how to change the course of such stories, if any.

3. **Systems that generate interactive stories, but only with a single dramatic conflict.** The third category contains solutions that generate ideally interactive stories, which let users shape their storyline not by choosing from a set of possible narrative actions, but more naturally by analyzing the desired actions and dialog options the user had directly typed in (or spoken to the microphone) via NLP and speech-to-text techniques.

   However, such stories are generated from a fixed set of rules and data structures predefined by the authors of these storytelling systems, restricting anyone else from defining their own input data and hence generating stories in a different domain.

Table 1 compares and contrasts the hereby-defined three categories of interactive storytelling systems existing today, marking the negative aspects **bold**.

Table 1. Comparison of existing interactive storytelling solutions.

Evaluated Aspect	Category 1	Category 2	Category 3
Definition of custom, domain-based input rules and data structures	**Difficult and unintuitive**	Possible	**Not possible**
Player methods of interacting with the generated stories and changing their storylines	Narrative actions and dialogue options	**Limited or none**	Written or spoken natural language
Story visualization and interface	**Text-based**	2D/3D/VR	3D

6 THESIS OBJECTIVES

A number of interactive storytelling systems have been reviewed and compared in the previous chapter. Based on this evaluation, the following major drawbacks of today's existing interactive storytelling solutions can be drawn:

- ✗ **Complicated and unintuitive forms of defining input data.** Solutions that generate complex interactive stories require anyone wanting to use such systems to master overly complicated rules and data structures, together with custom programming languages or story grammars.
- ✗ **Difficult or impossible to generate domain-specific interactive stories.** As a consequence to the previous statement, it is difficult for anyone to define his own domain-specific stories. Moreover, some solutions operate on hard-coded input data that cannot be altered in order to generate stories different from what the solution's creators have intended.
- ✗ **Text-based story visualization.** On the one hand, storytelling systems often present the generated interactive stories to the player via a text-based interface. On the other hand, systems that visualize stories using 2D/3D computer graphics or VR end up generating non-interactive stories, i.e. stories that the player cannot influence in any way from a narrative point of view.
- ✗ **Generated stories do not adapt to player's personality.** None of the existing storytelling solutions take the player's personality into account during the course of the generated stories.

It is mostly due to these disadvantages why existing solutions and approaches to interactive storytelling have found very little or no practical use at all.

Consequently, the goal of this work is to devise a new approach to interactive storytelling, which builds upon present-day techniques and formalisms in a way resulting in a coherent solution not having the above-mentioned drawbacks. In other words, the aim of this work is to define an approach to interactive storytelling conforming to the following four functional requirements:

- ✓ **Visual and intuitive form of defining input data.** By making it possible to define rules and data structures according to which stories are generated in a visual and intuitive way, it is easy for anyone to work with the resulting system and define his own input data.
- ✓ **Easy and possible to define domain-specific interactive stories.** Since defining custom input data is intuitive and does not require learning any special syntax or programming language, anyone can easily define interactive stories in a specific domain.
- ✓ **Story visualization by computer role-playing games.** Today's storytelling systems generate either text-based interactive stories, or stories visualized in 2D/3D/VR that are not interactive. Computer role-playing games present an ideal medium for visualizing generated stories in 2D/3D and yet making them interactive by letting players influence their course by committing narrative actions and choosing dialogue options.
- ✓ **Generated stories that adapt to the player's personality.** Narrative actions with which the player changes the course of all generated interactive stories are not chosen only according to input rules defined by the story's author, but also by taking account

the player's personality, i.e. what kind of actions and dialogue options had the player previously chosen (positive adaptation), or ignored (negative adaptation).

Table 2 summarizes the interactive storytelling solution that this work intends to devise, in the same manner as Table 1 summarized existing storytelling solutions.

Table 2. Summary of an interactive storytelling solution that forms the goal of this work.

Evaluated Aspect	Intended Goal
Definition of custom, domain-based input rules and data structures	Visual and intuitive
Player methods of interacting with the generated stories and changing their storylines	Narrative actions and dialogue options
Story visualization and interface	Computer role-playing games

The proposed approach to interactive storytelling is described in detail throughout the remaining chapters of this document.

Since there does not exist a suitable interactive storytelling solution that we could utilize or build upon, it is not possible to easily focus on all hereby-declared goals. Therefore, we focus mainly on describing the architecture of the devised interactive storytelling solution, including the utilized input rules and data structures, and the process of story generation.

Moreover, our aim is to develop a software prototype with which we are able to evaluate the proposed approach to interactive storytelling by generating interactive stories in computer role-playing games that are from an educational domain related to the history of computing and basics of programming languages. For that reason, the majority of examples described throughout this work belong to the presented domain.

7 ARCHITECTURE FOR INTERACTIVE STORIES IN RPGS

In this chapter, we describe a new and innovative approach to interactive storytelling the aim of which is to programmatically generate dynamic and interactive stories with computer role-playing games as their medium, therefore combining the dynamic and enthralling storyworlds created by interactive storytelling with the visual appearance, gameplay and popularity of computer role-playing games.

The proposed concept can be broken into three logical layers, as depicted in Figure 27.

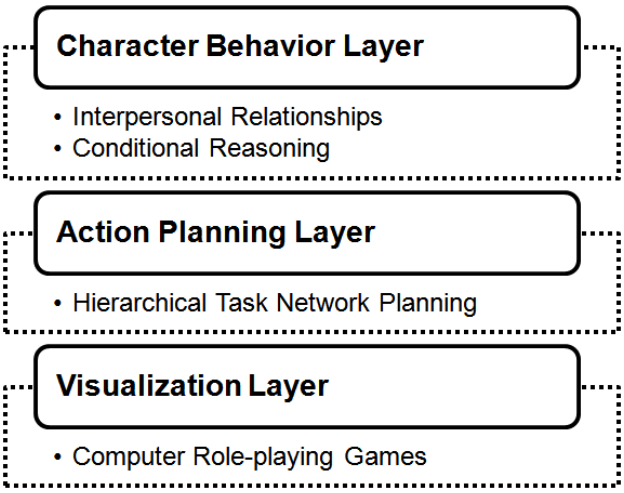


Figure 27. Logical structure of the proposed concept.

All generated interactive stories initiate in the topmost logical layer, named the *Character*[8] *Behavior Layer*, which is responsible for generating goals of all in-game characters, i.e. the player and all non-player characters. These goals are created according to *interpersonal relationships* among the player and non-player characters by conditionally matching their existing values to a set of *behavior patterns* and picking the resulting goals from the best matching patterns.

The behavior of characters results in creating plans and planning complex actions that move the story forward. The middle layer, called the *Action Planning Layer*, handles all the planning and replanning by transforming character goals set by the *Character Behavior Layer* into plans consisting of *atomic actions* that are to be visualized by the *Visualization Layer*. Moreover, the *Action Planning Layer* processes and evaluates actions previously committed by the player and all non-player characters that are reported back by the *Visualization Layer*.

The lowest logical layer called the *Visualization Layer* uses the concept of computer role-playing games as the visualization and storytelling medium, formalized into a set of *atomic actions* that a player (or a non-player character) is able to commit inside the game world. The *Visualization Layer* therefore provides graphical visualization of the generated interactive stories by executing actions planned for non-player characters and letting the player interact with the generated stories by letting him commit narrative actions, which are reported back to the *Action Planning Layer* immediately upon being committed.

[8] Since all stories are about people, despite the fact that references to them are often indirect or symbolic, and objects never play central roles in stories (as mentioned in chapter 2.2.2).

7.1 Visualization Layer

The lowest logical layer provides graphical visualization of the generated interactive stories to the player. This layer uses the concept of computer role-playing games as the visualization and storytelling medium and operates on *atomic actions*, described in detail below.

7.1.1 Atomic Actions

We have formalized computer role-playing games from an interactive storytelling point of view into a set of *atomic actions* having narrative impact that a player (or a non-player character) is able to commit inside the game world. Each atomic action consists of the following structure:

- **Name:** A string concisely describing the atomic action.
- **Source:** The type of an in-game element that can commit the atomic action, e.g., the player, or a non-player character.
- **Target:** The type of an in-game element that this atomic action is committed upon, e.g., an object or a container.
- **Parameter:** The type of an in-game element that the atomic action operates on. The presence of the parameter is optional.
- **Preconditions:** A set of statements regarding the specified source, target and parameter defining the circumstances under which the atomic action can be committed in the game world.
- **Effects:** A set of changes to the game world that are a consequence of the atomic action having been committed.

The typical set of atomic actions that describes a common computer role-playing game is shown in Table 3.

Table 3. Atomic actions that describe a typical computer role-playing game.

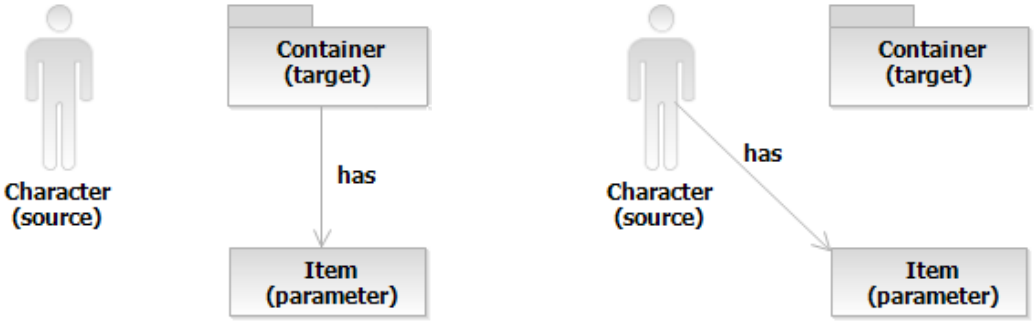
Source	Atomic Action[9]	Target	Description
Character	`GIVE (Item)`	Container	The character specified as the source, i.e. the player or a non-player character gives the item to a container set as the atomic action's target.
Character	`TAKE (Item)`	Container	The source, i.e. the player or a non-player character takes the item from the target container.
Character	`USE (Item)`	Element	The action's source, i.e. the player or a non-player character uses the item on the element specified as the target.
Character	`EQUIP ()`	Item	The player or a non-player character specified as the source equips the targeted item.
Character	`WALK_TO ()`	Element	The source, i.e. the player or a non-player character walks near the element specified as the atomic action's target.
Character	`TALK_TO ()`	Character	The player or a non-player character specified as the atomic action's source initiates a dialogue with the targeted character.

[9] The atomic actions are written in a shortened textual form with the following structure: `NAME (parameter)`.

As an example, the second atomic action mentioned in Table 3 is defined as follows:

Name: `TAKE`
Source: Character
Target: Container
Parameter: Item
Preconditions: Target has parameter
Effects: Target ~~has parameter~~[10]
Source has parameter

What is noteworthy to mention is that preconditions and effects of atomic actions can be easily represented visually. For example, the preconditions and effects of the atomic action `TAKE` (defined above) are depicted in Figure 28.



(a) Preconditions. **(b)** Effects.

Figure 28. Visual representation of an example atomic action.

7.2 Action Planning Layer

From a top-down perspective, the role of the middle logical layer, called the *Action Planning Layer*, is to transform character goals set by the *Character Behavior Layer* into plans consisting of actions that are to be executed and visualized by the bottommost logical layer – the *Visualization Layer*. From a bottom-up perspective, the *Action Planning Layer* is responsible for processing actions committed by the player and all non-player characters that were reported back from the *Visualization Layer*.

Since the described process is done on-the-fly in parallel while the player is playing a computer role-playing game, the hereby-described layer creates new quests based on the previous narrative actions committed by the player and seamlessly integrates them into the game, thus dynamically creating an interactive story, as perceived by the player.

The *Action Planning Layer* utilizes Hierarchical Task Network (HTN) planning in a similar way as described in [10]. Our algorithm searches for appropriate actions based on recursive matching of their effects with required preconditions. In other words, the planning algorithm finds all actions resulting in the required preconditions (see chapter 8 for details).

The *Action Planning Layer* operates on *simple actions*, *complex actions* and *action bindings*.

[10] Negation of the action's precondition (meaning that the target does not hold the parameter).

7.2.1 Simple Actions

A *simple action* refines the usage of exactly one atomic or simple action by specializing its source, target, parameter or by adding additional preconditions or effects. Unlike atomic actions, simple actions can alter *attributes* of characters and *interpersonal relationships* – features of the *Character Behavior Layer*. Every simple action has the following structure:

- **Name:** A string concisely describing the simple action.
- **Base action:** An atomic or simple action that this simple action refines.
- **Source:** The base action's source type or its subtype (see below).
- **Target:** The base action's target type or its subtype.
- **Parameter:** The type of an in-game element that this simple action operates on. The parameter is equal to or a subtype of the base action's parameter.
- **Preconditions:** A set containing preconditions inherited from the base action plus additional preconditions related to this simple action.
- **Effects:** A set containing effects inherited from the base action plus additional effects related to this simple action.

Below is an example of a simple action named `STEAL` that refines the atomic action `TAKE` (defined in section 7.1.1) and enables the player to steal items from non-player characters:

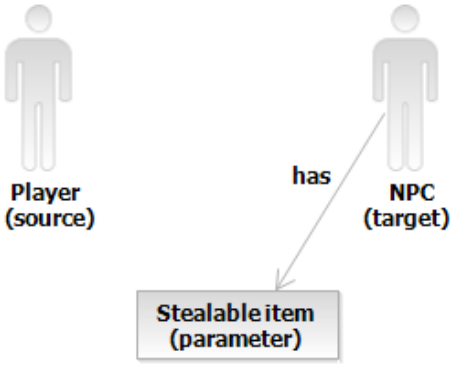
Name:	`STEAL`
Base action:	`TAKE`
Source:	Player (a character subtype)
Target:	Non-player character (a container subtype)
Parameter:	Stealable item (an item subtype, see below)
Preconditions[11]**:**	[Target has parameter]
Effects[11]**:**	[Target ~~has parameter~~] [Source has parameter] Parameter is "stolen" Target "dislikes" source Source's "karma" decreased

The definition of a simple action may seem complicated at first, however, both preconditions and effects of all simple actions are easily visualizable, and thus simple actions can be intuitively defined in a visual way. For example, Figure 29 depicts the preconditions and effects of the hereby-defined simple action `STEAL`.

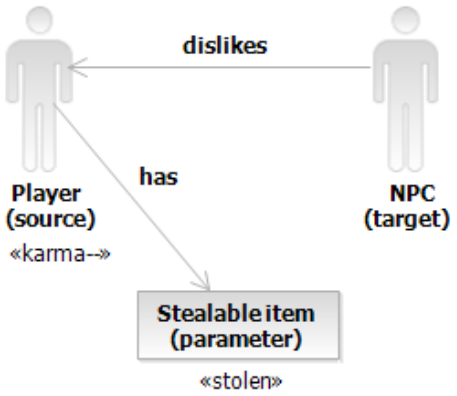
The last two effects of the example simple action alter the player's *attributes* and *interpersonal relationships* between the target non-player character and the player – see section 7.3.

The third effect of the example simple action marks the parameter as "stolen," since it is defined as being an item subtype named "stealable item," which has a "stolen" property defined. An item instance can belong to multiple item subtypes (called categories), each having defined custom properties (called tags) that can be set on the item instance – see Figure 30.

[11] The preconditions and effects inherited from the base action are enclosed in [square brackets].



(a) Preconditions.



(b) Effects.

Figure 29. Visual representation of an example simple action.

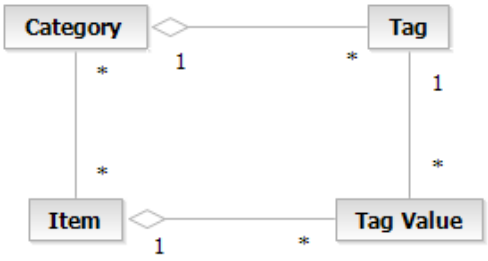


Figure 30. Item subtypes and properties.

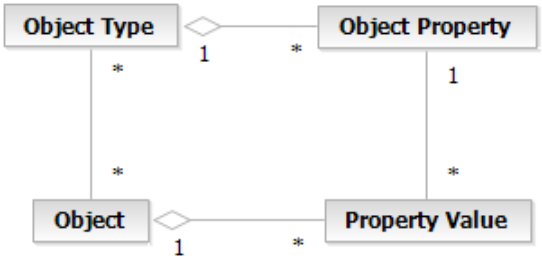


Figure 31. Object subtypes and properties.

Likewise, subtypes of all other core in-game elements (described in section 3.4), i.e. objects (see Figure 31), containers (see Figure 32) and characters (see Figure 33) are also supported, with each subtype having its own custom properties defined.

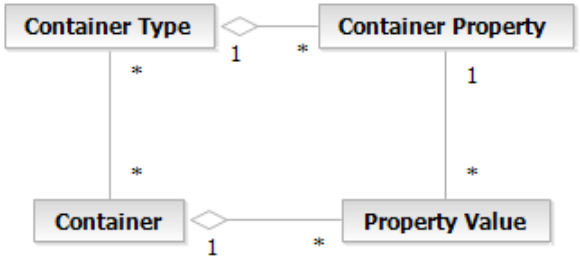


Figure 32. Container subtypes and properties.

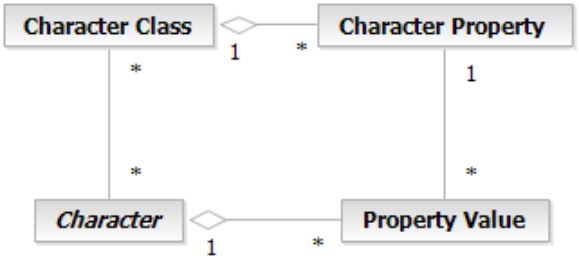


Figure 33. Character subtypes and properties.

All of the hereby-mentioned subtypes are based on a single analytical pattern that is depicted in Figure 34. For implementations details, consult the technical documentation included in the appendices.

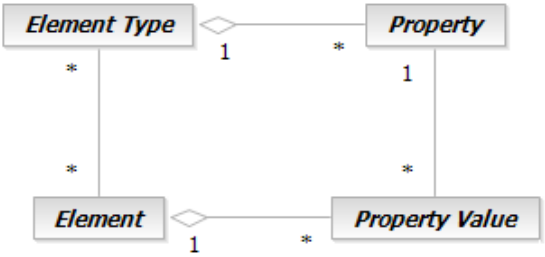


Figure 34. An analytical pattern as the basis of all subtypes.

7.2.2 Complex Actions

A *complex action* encloses multiple atomic, simple or complex actions in a sequence. Every complex action has the following structure:

- **Name:** A string concisely describing the complex action.
- **Source:** The type of an in-game element that is the source of all enclosed actions.
- **Targets:** Types of in-game elements that are targets of the enclosed actions.
- **Parameters:** A set of in-game elements that the enclosed actions operate on. Each parameter is identified by a unique name that distinguishes it from the other parameters.
- **Enclosed actions:** A set containing atomic, simple or other complex actions that this complex action encloses. The actions are executed sequentially and can be interrupted.
- **Preconditions:** A filtered[12] set containing preconditions of all enclosed actions plus additional preconditions related to this complex action.
- **Effects:** A filtered[12] set containing effects of all enclosed actions plus additional effects related to this complex action.

Following is an example of a complex action, with which a non-player character unlocks a locked item for the player if the particular non-player character likes the player:

Name:	`UNLOCK_LOCKED_ITEM_FOR_PLAYER`
Source:	Non-player character
Targets:	Player Lockable item `i` (an item subtype)
Parameters:	Lockable item `i`

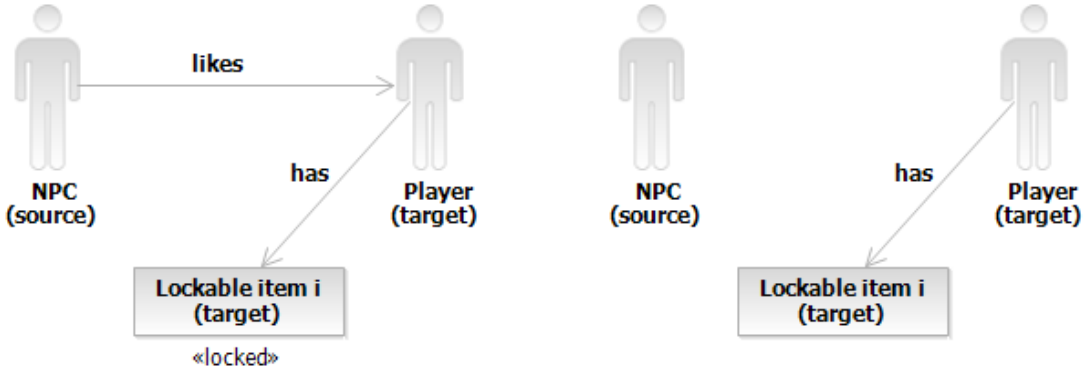
Enclosed actions[13]:	`Source : TAKE (i)` → `Player`
	`Source : UNLOCK()` → `i`
	`Source : GIVE (i)` → `Player`

Preconditions:	[Player has `i`] [`i` is "locked"] Source "likes" player
Effects:	[Player has `i`] [`i` ~~is "locked"~~]

Similarly as simple actions, complex actions are also easily visualizable in terms of their preconditions and effects, making them intuitively definable in a visual way. Figure 35 shows both preconditions and effects of the example complex action defined above.

[12] The sets do not contain preconditions nor effects that are created and at the same time annulled during the sequential execution of actions enclosed by the complex action.

[13] Each action is written in the form: `Source : NAME (`*parameter(s)*`)` → `Target`.



(a) Preconditions. **(b)** Effects.

Figure 35. Visual representation of an example complex action.

7.2.3 Action Bindings

The purpose of optional *action bindings* is to explicitly permit or disallow bindings of atomic, simple, and complex actions to actual in-game entities. Action bindings therefore define what can or cannot be done with all defined story elements, and what exactly can or cannot the player along with all existing non-player characters do.

The first pair of example action bindings given below denotes that the player cannot steal and can only take (in a peaceful way) the item "thyme syrup" from the non-player character representing a good doctor, whereas the second pair denotes that the player has no other option but to steal "thyme syrup" if he wants to obtain it from the evil doctor:

Actual in-game entities:

- Item "Thyme syrup"
- Non-player character "Good Doctor"
- Non-player character "Evil Doctor"

Action bindings[13]:

- `[CAN]  Player : TAKE  ("Thyme syrup") → "Good Doctor"`
- `[CAN'T] Player : STEAL ("Thyme syrup") → "Good Doctor"`
- `[CAN'T] Player : TAKE  ("Thyme syrup") → "Evil Doctor"`
- `[CAN]  Player : STEAL ("Thyme syrup") → "Evil Doctor"`

7.3 Character Behavior Layer

The *Character Behavioral Layer* is responsible for generating goals of all in-game characters, i.e. the player and all non-player characters. These goals are created according to *interpersonal relationships* among characters by matching their existing values[14] to a set of *behavioral patterns* (see below) and picking the resulting goals from the best matching patterns. All generated characters' goals are afterwards translated to plans by the *Action Planning Layer*.

[14] The storyteller sets the initial values of interpersonal relationships, if necessary. Otherwise, all relationships are initialized to their default values.

7.3.1 Behavioral Patterns

A *behavioral pattern* defines the circumstances that lead to a change in the behavior of characters from a narrative point of view. The set of all behavioral patterns defines the conditional reasoning of all in-game characters. Each behavioral pattern has the following structure:

- **Description:** An optional text concisely describing the behavioral pattern.
- **Preconditions:** A set of in-game elements, including their properties and relationships among them. Also included are the preconditions of actions that represent goals. Undesirable preconditions may be explicitly excluded from the set.
- **Goals:** A set containing goals describing the change in characters' behavior as a consequence to the situation portrayed by the pattern's preconditions. Goals can be represented by actions of any type, or by changes in properties of in-game elements and in relationships among them. Each goal is bound to a certain character.
- **Effects:** A set containing effects of all identified goals. Undesirable effects may be explicitly excluded from the set.

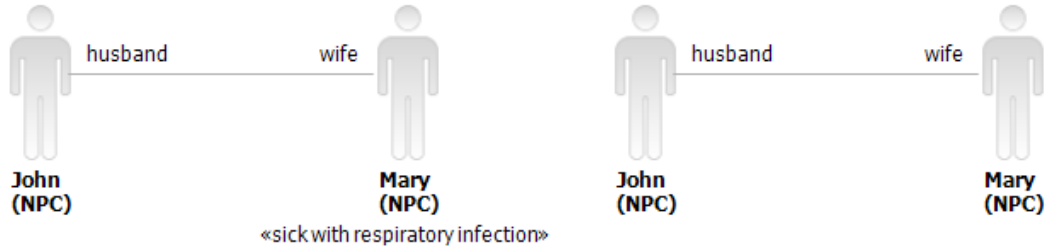
The behavioral pattern defined below describes a situation in which John, a husband of Mary with a respiratory infection, cures his wife with thyme syrup that the player had given him:

Preconditions: Non-player character `John`
Non-player character `Mary`
`John` is "husband" to `Mary`, `Mary` is "wife" to `John`
`Mary` is "sick with respiratory infection"
[…preconditions of the two actions that represent goals…]

Goals:
```
Player : GIVE ("Thyme syrup") → John
John   : USE  ("Thyme syrup") → Mary
```

Effects: […effects of the two actions that represent goals…]
`Mary` ~~is "sick with respiratory infection"~~

It is possible to define behavioral patterns also visually in an intuitive way, similarly as all types of actions described in the preceding pages. For example, the hereby-defined example behavioral pattern's preconditions and effects are depicted in Figure 36.



(a) Preconditions. **(b)** Effects.

Figure 36. Visual representation of an example behavioral pattern.

This behavioral pattern contains examples of two actual *interpersonal relationships*, namely "is husband to," "is wife to" and an *attribute* "sick with respiratory infection." Following is a detailed description of all types of character properties (i.e. interpersonal relationships and attributes) that the *Character Behavior Layer* operates on.

7.3.2 NPC → Character Relationships

Interpersonal relationships oriented from a non-player character[15] to any type of in-game character (i.e. the player or another non-player character) are identified with their unique label – see Figure 37.

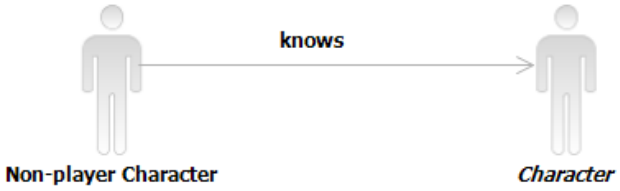


Figure 37. An NPC → Character relationship denoting that the left NPC knows the character on the right.

Under normal circumstances, NPC → Character relationships are either absent (the default initial value) or present. Certain types of NPC → Character relationships can be set to have a numerical value instead of being either present or absent. For example, the relationship "knows" depicted in Figure 37 can have a value from the interval <0,1>, with zero representing the default initial value.

In addition, there are cases when two interpersonal relationships are mutually exclusive (meaning that the presence of one disallows the presence of the other and vice-versa). Such pairs of relationships are merged into *relationships with two complementary values* – as depicted in Figure 38.

Figure 38. An NPC → Character relationship with complementary values denoting that the left NPC either likes or dislikes the character on the right.

Such interpersonal relationships can have only one of their two values present at a time, e.g., if the precondition "Tom likes player" is true at a given time, then the precondition "Tom dislikes player" is false at the same time. Relationships with complementary values can also have numerical values. For example, the hereby-defined relationship "dislikes OR likes" can have a value from the interval <-1,1>, where -1 denotes "dislikes" and +1 denotes "likes."

7.3.3 Character Roles

Another type of relationships between in-game characters are *character roles*. Unlike NPC → Character relationships, character roles can be oriented from the player as well, see Figure 39.

Figure 39. A character role denoting that the player is the teacher's student.

[15] Relationships oriented from the player are not considered, because monitoring such relationships and limiting the number of narrative actions the player can commit based on their presence would degrade the player's experience of the generated stories.

Moreover, character roles can be bilateral, as depicted in Figure 40.

Figure 40. A bilateral character role denoting that John is Mary's husband and Mary is John's wife.

Similarly to NPC → Character relationships, character roles are either absent (the default initial value) or present. If a *bilateral character role* is present, then preconditions testing either end evaluate to true, e.g., "John is husband to Mary" and "Mary is wife to John" are either both true, or both false in a given time. In contrast to NPC → Character relationships, character roles cannot have numerical values assigned.

7.3.4 Character Attributes

Every in-game character, whether being the player or a non-player character, can have various *attributes* defined. Such attributes are either unnumbered or numbered.

Unnumbered character attributes do not have any numerical value and are either absent (the default initial value) or present. An example of an unnumbered character attribute is "sick with respiratory infection" referenced in a behavioral pattern example mentioned at the beginning of section 7.3.

Numbered character attributes are present in every in-game character (i.e. the player and all non-player characters) and store a numerical value (individually for every character and from a custom-defined interval), unlike unnumbered attributes. An example of numbered attributes found in the majority of computer role-playing games is located in the left column of Table 4.

In addition, attributes of characters can be easily made domain-specific, e.g., an interactive storyworld created in the domain of teaching programming languages can have defined numbered character attributes located in the right column of Table 4.

Table 4. Examples of various numbered character attributes.

Common Attributes	Domain-specific Attributes
▪ Agility	▪ C++ proficiency
▪ Charisma	▪ Java proficiency
▪ Endurance	▪ Lisp proficiency
▪ Intelligence	
▪ Perception	
▪ Strength	

8 THE STORY GENERATION CYCLE

The architecture of the proposed interactive storytelling system, as described in the previous chapter, consists of three logical layers – the *Character Behavior Layer*, the *Action Planning Layer* and the *Visualization Layer*. These layers are cyclically used to generate interactive stories, as depicted in Figure 41.

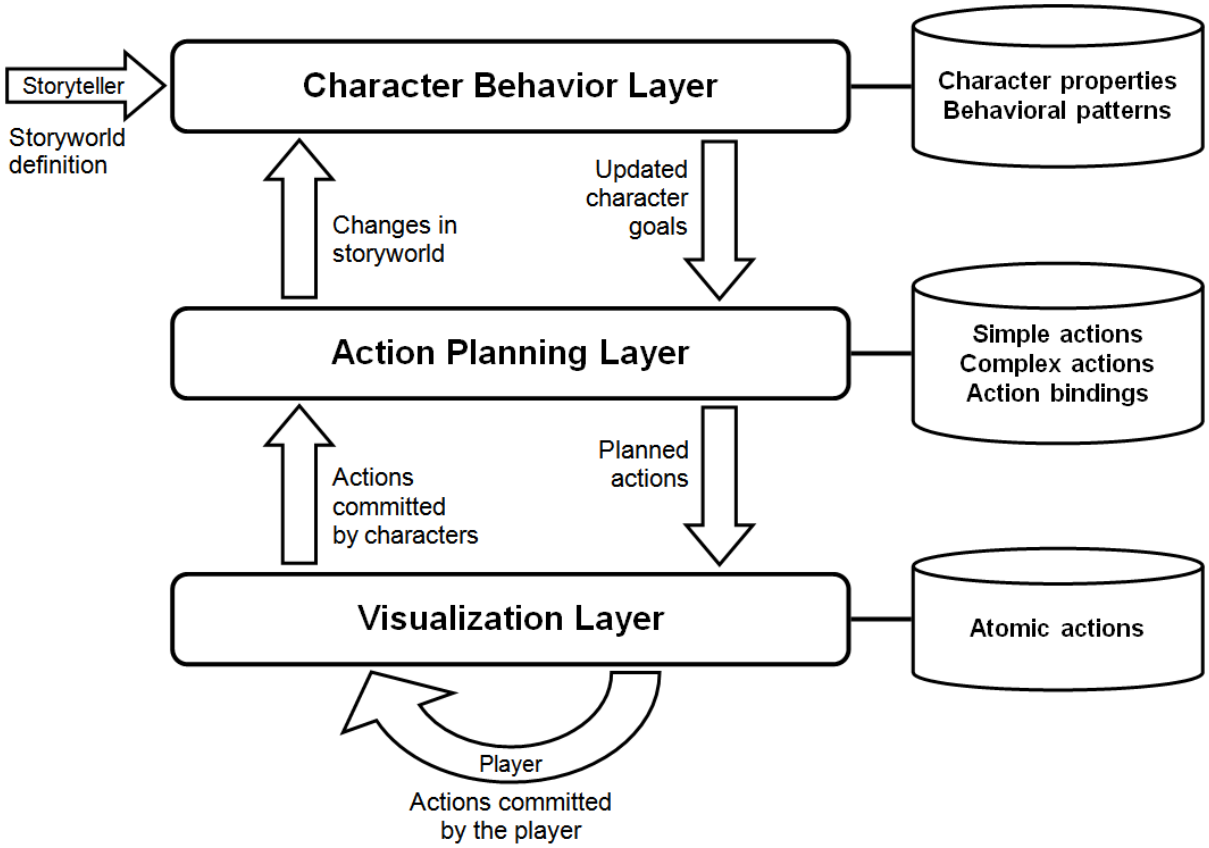


Figure 41. The story generation cycle.

The whole story generation process consists of three separate phases:

- *Preparation phase*, in which custom domain-specific rules and input data are created by the storyteller and used afterwards in the subsequent phases.
- *Initialization phase*, in which the storyteller defines the initial state of the storyworld by means of creating and placing non-player characters, objects and items throughout the storyworld.
- *Story generation phase*, during which interactive stories are cyclically generated by all three layers depicted in Figure 41. The *Character Behavior Layer* identifies goals of all non-player characters and the player, which are afterwards transformed into plans by the *Action Planning Layer* and sent to the *Visualization Layer* to be executed. The player executes his desired narrative actions (that may have not been planned, but must have its preconditions met), which are signaled back to the *Action Planning Layer* along with all actions committed by non-player characters. All changes to the storyworld are signaled to the *Character Behavior Layer*, which updates goals of all characters accordingly and the whole process continues until the player has accomplished all of his goals, in which case the story ends successfully.

8.1 Preparation Phase

Before any interactive stories can be generated, the author of the stories, i.e. the storyteller creates his domain-specific rules and input data based on which the proposed storytelling system operates. In other words, the storyteller defines all[16] necessary simple and complex actions, action bindings, types of possible character properties (attributes, relationships and roles between characters) and behavioral patterns – as described in the previous chapter.

Let us suppose that the storyteller wants to define a storyworld, based on which educational stories related to history of computing and programming basics are to be generated. We present and describe an example of such storyworld throughout the entire chapter.

The example storyworld requires the *Visualization Layer* to support the most common atomic actions mentioned in Table 3 (on page 34) that are typical for computer role-playing games. However, due to the educational nature of the example storyworld, the following two domain-specific atomic actions were additionally implemented in the *Visualization Layer*:

Table 5. Domain-specific atomic actions used in the example storyworld.

Source	Atomic Action	Target	Description
Player	`SOLVE_PROGRAM_CODE`	Computer[17]	The player is presented with a fragment of program code having multiple options of commands at various positions. The player is required to select the proper commands from the available options, resulting in a valid and error-free program code (see screenshot in Figure 47).
Player	`ANSWER_QUESTION`	Non-player character	The targeted non-player character asks the player a domain-specific question. The player has to answer correctly by choosing the correct answer from a list of multiple answers (see screenshot in Figure 50).

The core of the example educational storyworld is a malfunctioning mainframe computer that the player is to successfully repair. For simplicity's sake, let us presume that one way of repairing the mainframe computer is to reprogram it, as defined by the following simple action:

Name: `REPROGRAM`

Base action: `SOLVE_PROGRAM_CODE`

Source: Player

Target: Mainframe computer[17]

Preconditions: Target is malfunctioning

Effects: Target ~~is malfunctioning~~

[16] The storyteller can also use and alter a predefined set of input data, if available for the desired domain.
[17] An object subtype (see page 37) defined in the example storyworld.

In order for the generated stories to be more enthralling, let us define another way that the player can actually repair the malfunctioning mainframe computer. The following simple action denotes that replacing the circuit board also repairs the mainframe computer:

Name: REPAIR

Base action: USE

Source: Player

Target: Mainframe computer

Parameter: Circuit board[18]

Preconditions: [Source has parameter]
Target is malfunctioning

Effects: Source ~~has parameter~~
Target ~~is malfunctioning~~

Besides having an option of simply finding the circuit board item scattered throughout the storyworld, the player can also steal the circuit board (or any other item) from a non-player character, as defined by the following simple action:

Name: STEAL[19]

Base action: TAKE

Source: Player

Target: Non-player character

Parameter: Item

Preconditions: [Target has parameter]

Effects: [Target ~~has parameter~~]
[Source has parameter]
Target "dislikes" source

If we wanted to add special ways how the player can obtain the circuit board with which he is able to repair the mainframe computer, we could define the following simple action donating that a non-player character will give the circuit board to the player if he is trustworthy:

Name: GIVE_CIRCUIT_BOARD

Base action: GIVE

Source: Non-player character

Target: Player

Parameter: Circuit board[18]

Preconditions: [Source has parameter]

[18] An item subtype defined in the example storyworld.

[19] This simple action is identical to the example simple action defined on page 36, except that the hereby-defined action enables the player to steal any possible item (and not just "stealable item" subtypes) and lacks some of the effects (i.e. the stolen item is not marked "stolen" and the player's "karma" attribute remains unaffected).

Target is "trustworthy"

Effects: [Source ~~has parameter~~]
[Target has parameter]

Let us define how a player can become trustworthy, e.g., by correctly answering a question asked by a non-player character, as defined by the following simple action:

Name: `BECOME_TRUSTWORTHY`

Base action: `ANSWER_QUESTION`

Source: Player

Target: Non-player character

Preconditions:

Effects: Source is "trustworthy"

Other than answering a question, let us enable the player to become trustworthy also by finding and reading a mainframe handbook, as defined by the following simple action:

Name: `READ`

Base action: `EQUIP`

Source: Player

Target: Mainframe handbook[20]

Preconditions: [Source has parameter]

Effects: Source is "trustworthy"

Moving on to the *Character Behavior Layer*, where the storyteller defines a behavior pattern denoting that if a mainframe computer is malfunctioning, then his owner wants the mainframe to become functional again:

Preconditions: Non-player character `Scientist`
Mainframe computer `Old Mainframe`
`Scientist` is "owner" of `Old Mainframe`
`Old Mainframe` is "malfunctioning"

Goals: `Scientist : Old Mainframe` ~~is "malfunctioning"~~

Effects: `Old Mainframe` ~~is "malfunctioning"~~

8.2 Initialization Phase

After having defined all necessary custom domain-specific rules comes the initialization phase, in which the storyteller defines the storyworld that he wants the generated stories to take place in. He therefore creates the desired non-player characters along with their initial attributes, roles and relationships between them. The storyteller can also optionally scatter

[20] An item subtype defined in the example storyworld.

objects, containers and items throughout the storyworld as desired. Let us suppose that in the example storyworld, the storyteller defined and placed the following elements:

- Mainframe computer `Tim's Mainframe`, which is "malfunctioning"
- Non-player character `Tim`, who is "owner" of `Tim's Mainframe`
- Non-player character `John`, who has a `Circuit Board`
- A `Mainframe Handbook` contained in a wooden chest

8.3 Story Generation Phase

After the storyteller had defined the initial state of the storyworld, the *Character Behavior Layer* analyzes the storyworld and chooses one or more goals that best match its current state, i.e. goals having all preconditions met.

In the example storyworld, the goal to have the malfunctioning mainframe computer repaired is chosen and set as an active goal for the non-player character Tim, who asks the player to help achieve this particular goal after the game has started. Assuming that the player accepts, the player's goal is set to make the mainframe computer operable; or in other words, to make the mainframe not "malfunctioning" – as depicted in Figure 42.

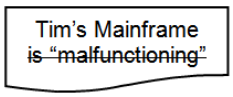


Figure 42. The player's active goal.

Such goals are afterwards processed by the *Action Planning Layer*, which translates all player's and non-player characters' active goals to plans. The planning process is based on the Hierarchical Task Network (HTN) planning formalism that recursively finds appropriate actions, effects of which meet the required conditions.

The planning process starts out with the effects of each active goal and finds any complex, simple or atomic actions (in this order) that have their effects match the goal's effects. In the example storyworld, the player's goal has only one effect (Tim's Mainframe ~~is "malfunctioning"~~), which is matched by effects of two simple actions – `REPROGRAM` and `REPAIR`, as depicted in Figure 43.

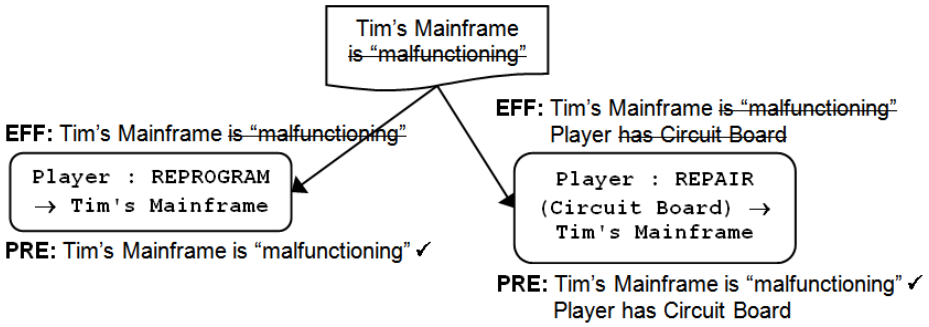


Figure 43. The player's active goal matched by simple actions `REPROGRAM` and `REPAIR`.

The whole planning process afterwards runs recursively to search for actions that have their effects match the unsatisfied preconditions[21] of the newly found actions. In the example storyworld, the newly found actions are REPROGRAM (with a single, satisfied precondition) and REPAIR, which has one[22] unsatisfied precondition (Player has Circuit Board). This precondition is matched by effects of two simple actions – STEAL and GIVE_CIRCUIT_BOARD, both of which are connected to the constructed plan, as shown in Figure 44.

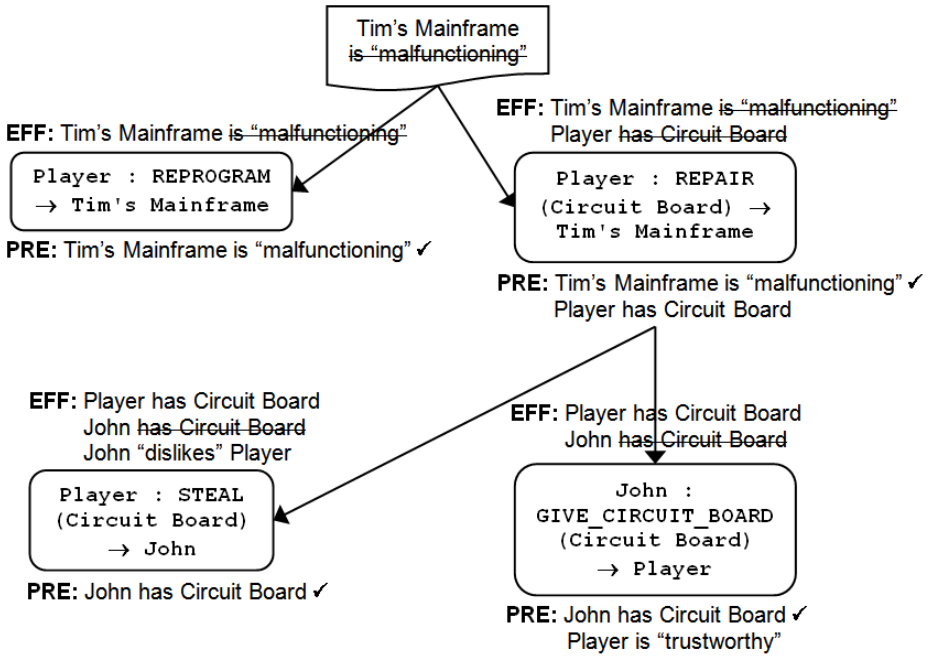


Figure 44. The simple action REPAIR matched by simple actions STEAL and GIVE_CIRCUIT_BOARD.

Likewise, the simple action GIVE_CIRCUIT_BOARD also has one[22] unsatisfied precondition (Player is "trustworthy") that the planning algorithm matches by effects of simple actions BECOME_TRUSTWORTHY and READ, both of which are also connected to the resulting plan, as depicted in Figure 45.

The newly connected simple action READ also has an unsatisfied precondition (Player has Mainframe Handbook), which is matched by the planning algorithms in the same manner as the previous actions, i.e. the planner looks for ways how the player can obtain the handbook. In the example storyworld, the handbook can be taken from a wooden chest.

The resulting plan for every active goal is a tree-like graph that contains multiple paths of complex, simple or atomic actions, which result in accomplishing the particular goal when followed in the storyworld – see Figure 45.

After plans for the player and all non-player characters are constructed, a subset of all available player's planned paths is chosen and delegated to the *Visualization Layer*, along with any planned actions that each non-player character should commit.

[21] Since the planner matches the required preconditions with effects of candidate actions, all actions depicted in figures in this chapter have preconditions placed below them and effects above.
[22] The other precondition is satisfied (marked with ✓) in the current state of the example storyworld.

The subset of the player's plan is chosen according to the player's user model, which contains records of all actions that he had previously successfully committed in other storyworlds. Thanks to these records, it is possible to filter out and ignore planned paths containing similar[23] actions that the player previously successfully committed (*positive personalization*), or actions he had not yet committed or failed to commit successfully (*negative personalization*). If the player's user model contains no records (i.e. the player is playing through his first storyworld), then the subset of player's planned paths is selected randomly.

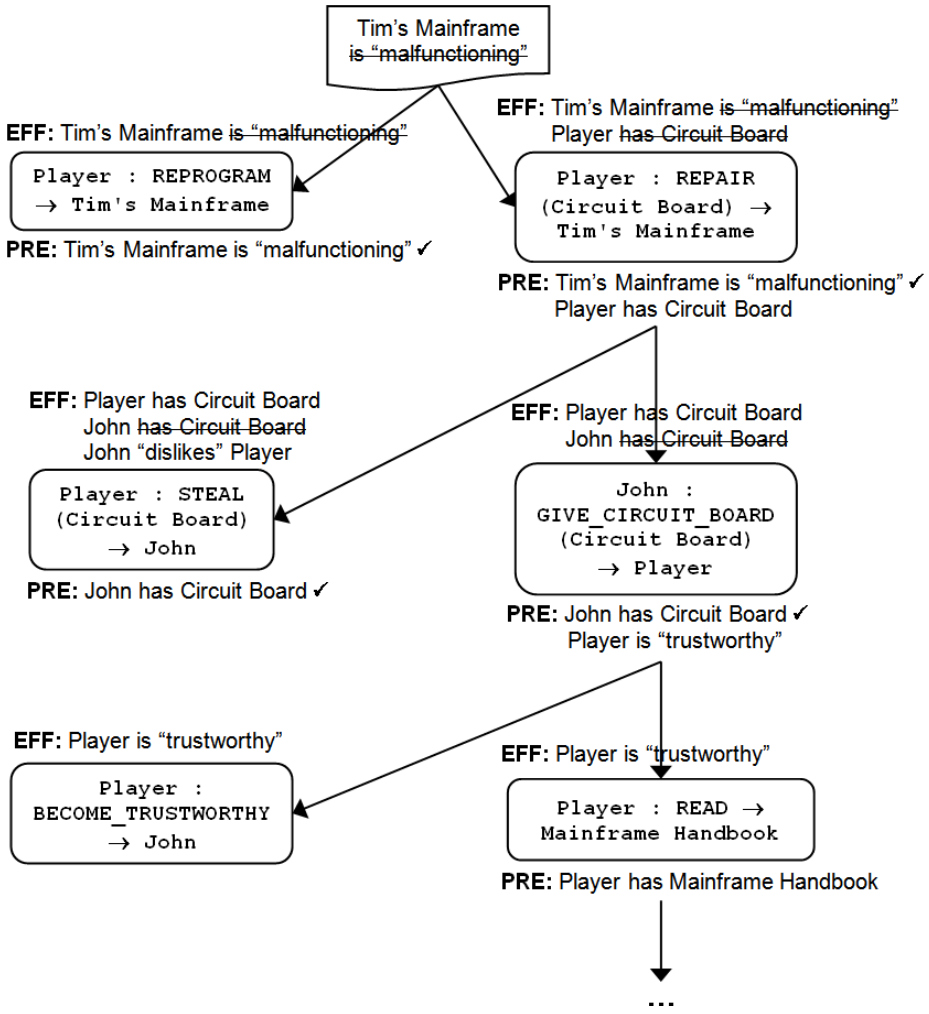


Figure 45. An example plan consisting of six simple actions that are interconnected by common preconditions (PRE) and effects (EFF), with the goal being the plan's root. The plan is constructed from top to bottom, but carried out from bottom to top.

In the example plan shown in Figure 45, presuming positive personalization and that the player's user model states that the player had previously committed the atomic action `SOLVE_PROGRAM_CODE`, then the path with the simple action `REPROGRAM` is chosen[24] and delegated to the *Visualization Layer*, as shown in Figure 46.

[23] Similar as in identical actions or actions based on a common atomic action, e.g., the simple action `STEAL` is based on (refines) the atomic action `TAKE`, and therefore `STEAL` and `TAKE` are considered being similar.
[24] Since the simple action `REPROGRAM` is similar to (refines) the `SOLVE_PROGRAM_CODE` atomic action.

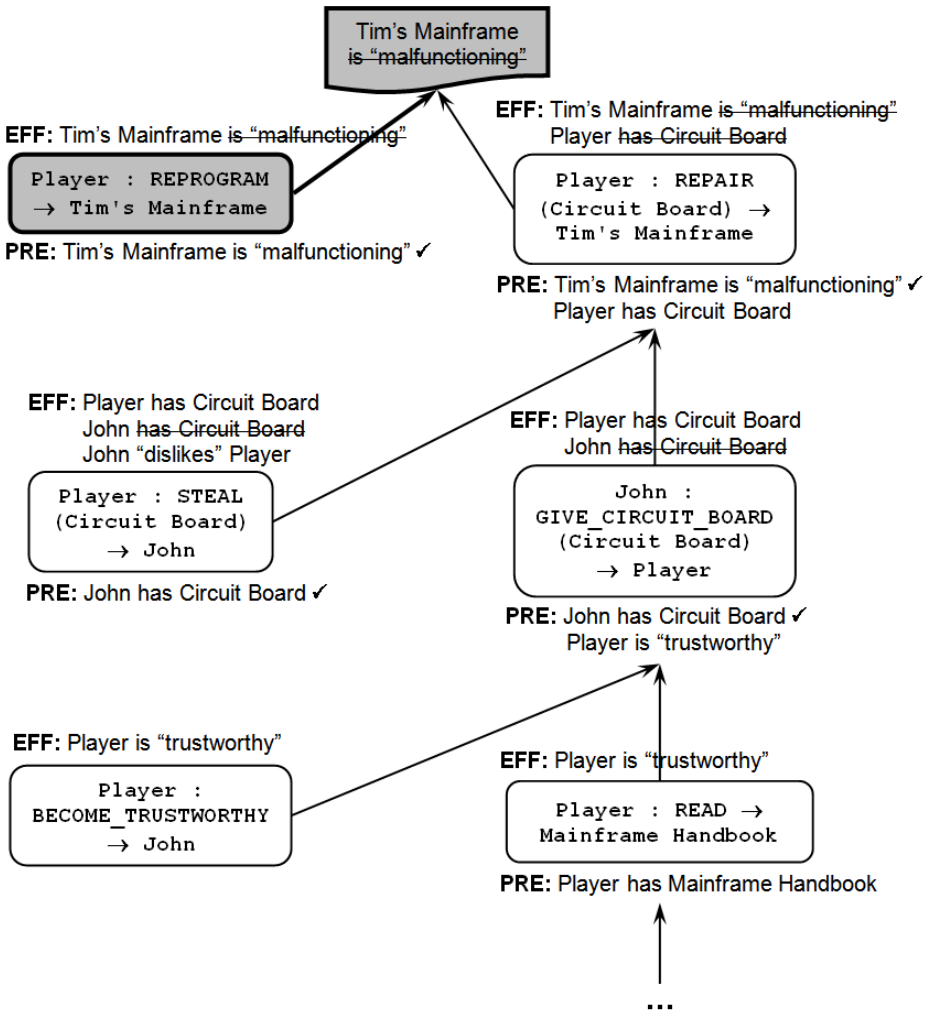


Figure 46. An example of a chosen path (colored gray) through the `REPROGRAM` simple action.

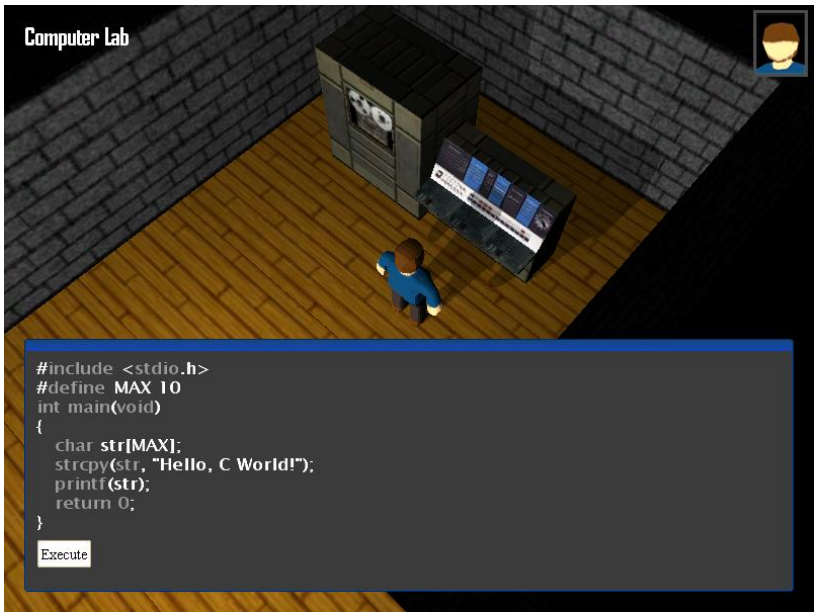


Figure 47. The player while reprogramming the mainframe computer.

The *Visualization Layer* receives planned actions for both the player and non-player characters from the *Action Planning Layer* and the actions destined for non-player characters are committed by the particular characters, whereas the actions planned for the player are presented to him as multiple options via the game's graphical interface, e.g., in the example storyworld, the player is guided to reprogram the mainframe computer – see Figure 47.

The player commits, either successfully or unsuccessfully, his desired action (that may not have been planned, but must have its preconditions met), what is afterwards signaled back to the *Action Planning Layer*, along with actions that were, either successfully or unsuccessfully, committed by any non-player characters.

In the example storyworld, let us assume that the player tried to reprogram the mainframe computer, but failed to solve the task successfully, i.e. he did not select the proper commands from multiple options (as required by the `SOLVE_PROGRAM_CODE` atomic action). This unsuccessful player's attempt (see Figure 48) is signaled back to the *Action Planning Layer*.

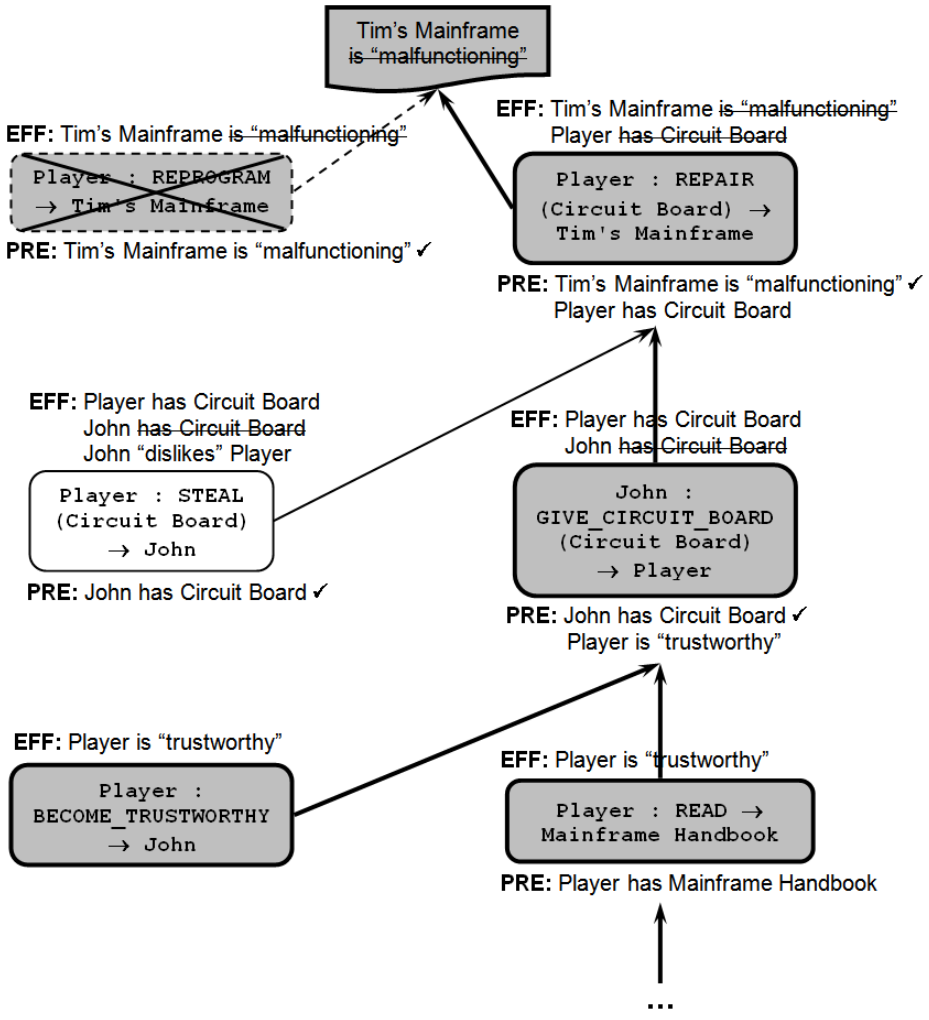


Figure 48. A new chosen path through the `REPAIR` simple action.

All signaled actions are processed by the *Action Planning Layer* that matches all committed actions to player's and non-players' existing plans, which remain either unaffected, or are replanned according to the new state of the storyworld, i.e. new planned paths are again chosen based on the player's user model, as described in the preceding paragraphs.

Let us presume that in the example storyworld, the player had unsuccessfully tried to reprogram the mainframe computer, which resulted in having the *Action Planning Layer* chose another path to accomplish the player's goal – by repairing the mainframe computer with a spare circuit board, see Figure 48.

The player can obtain the required circuit board by either stealing it or having the non-player character John give it to him. Let us suppose that according to the player's user model, the *Action Planning Layer* chose to ignore the path involving stealing the item and instead focused on the path involving John the non-player character. This path requires the player to become trustworthy, which is accomplishable by either correctly answering John's question (the simple action `BECOME_TRUSTWORTHY`) or by reading the mainframe handbook (simple action `READ`). In our example, both of these options have been chosen according to the player's user model, as depicted in Figure 48, and delegated down to the *Visualization Layer*, as described in the preceding paragraphs.

The *Visualization Layer* now presents the player with two options on how to obtain the circuit board with which the mainframe can be repaired – either by answering John's question correctly (see Figure 50) or by reading the mainframe handbook. Let us now suppose that the player tries, but fails to answer John's question correctly, i.e. the player has unsuccessfully committed the `BECOME_TRUSTWORTHY` simple action, as shown in Figure 49.

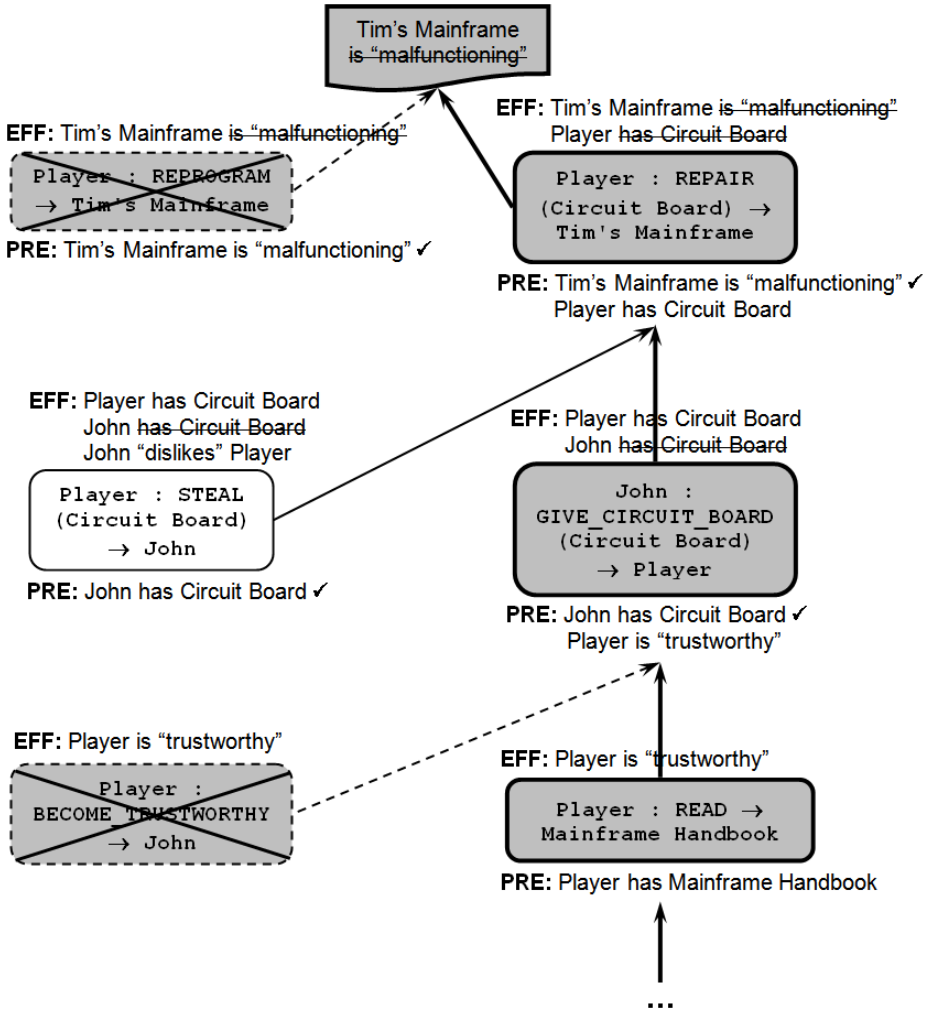


Figure 49. The path through the `REPAIR` simple action after the player had failed to answer John's question.

The other option the player has now left in order to accomplish his goal is to find and read the mainframe handbook. Let us suppose that the player reads the handbook (i.e. commits the simple action `READ`) and becomes trustworthy, receives the circuit board from John (simple action `GIVE_CIRCUIT_BOARD`), and finally repairs the malfunctioning mainframe with it (simple action `REPAIR`). All of these actions[25] are signaled to the *Action Planning Layer* from the *Visualization Layer*, as mentioned in the preceding paragraphs.

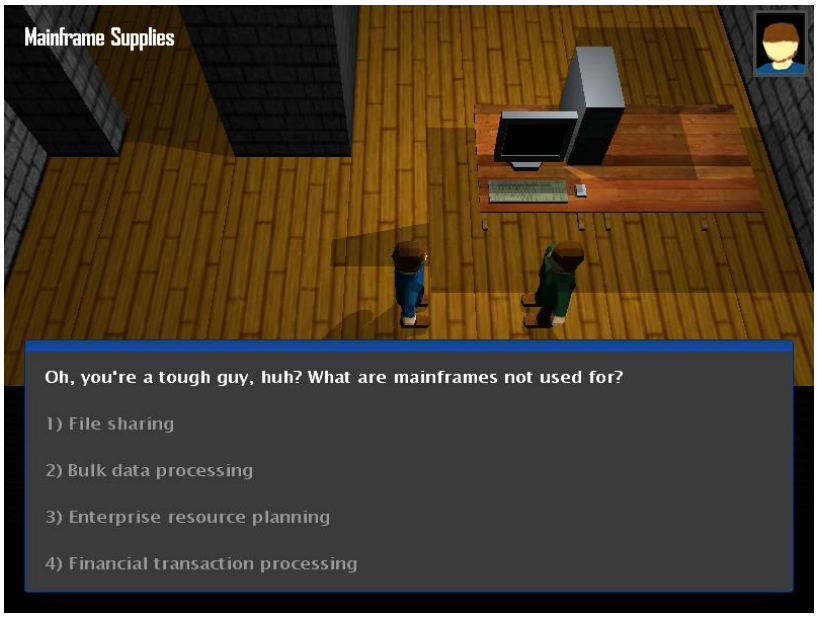


Figure 50. The player while answering John's question.

Any alterations to the state of the storyworld, such as changes in characters' attributes, roles or relationships are signaled to the *Character Behavior Layer*, which collects all changes to the storyworld from the *Action Planning Layer* and determines whether any of the existing goals are affected (in terms of having been accomplished or not being accomplishable anymore), or whether preconditions for any new goals are met. The current set of both the player's and non-player characters' goals is updated and any changes are delegated back to the *Action Planning Layer*.

The hereby-described process is repeated in a cyclic manner until the player has accomplished all of his goals, in which case the story ends. An ending also occurs if the player is unable to accomplish all of his existing goals and there are no more possible paths left to do so.

In the example storyworld, the mainframe computer is no longer malfunctioning as a consequence to the fact that the player had successfully repaired it, what the *Character Behavior Layer* intercepts as having accomplished the player's goal. Since the player has no more goals left unaccomplished and there are no other goals defined that match the current state of the example storyworld, the story ends at this point.

[25] See section 9.2 for screenshots showing all actions committable by the player in the prototype storyworld.

8.4 Comparison to Existing Approaches

When compared to existing approaches in the field of interactive storytelling, our proposed concept is an innovative combination of both modern interactive storytelling strategies described in section 4.4 – the plot-centric approach and the character-centric approach.

The proposed approach to interactive storytelling uses planning formalisms to create action plans for characters, similarly as the character-centric approach. However, in contrast to the character-centric approach, story characters do not behave autonomously, and their plans are created and shaped according to the main storyline goals as defined by the storyteller.

Likewise as the plot-centric approach, the proposed approach to interactive storytelling uses a drama manager (i.e. the *Character Behavior Layer*) to monitor the development of the generated stories and to influence their environment according to constraints set by the storyteller. However, in contrast to the plot-centric approach, actions of characters are planned by means of planning formalisms.

On the one hand, the proposed concept shares basic ideas behind existing approaches to interactive storytelling, however its uniqueness lies in combining these ideas in an innovative way to enable programmatic generation of interactive stories in the form of computer role-playing games.

9 PROTOTYPE OVERVIEW

We have implemented a software prototype based on the innovative approach to interactive storytelling described in the previous chapters.

The prototype reads from an input file a set of rules and data structures (described in chapter 7) defining a storyworld from which educational interactive stories are generated (as described in chapter 8) and presented to the player through a computer role-playing game. The stories progress on-the-fly by reacting to narrative actions that the player had committed throughout the game.

9.1 Implementation Overview

The software prototype is written in the Java programming language and its architecture is depicted by the component diagram shown in Figure 51.

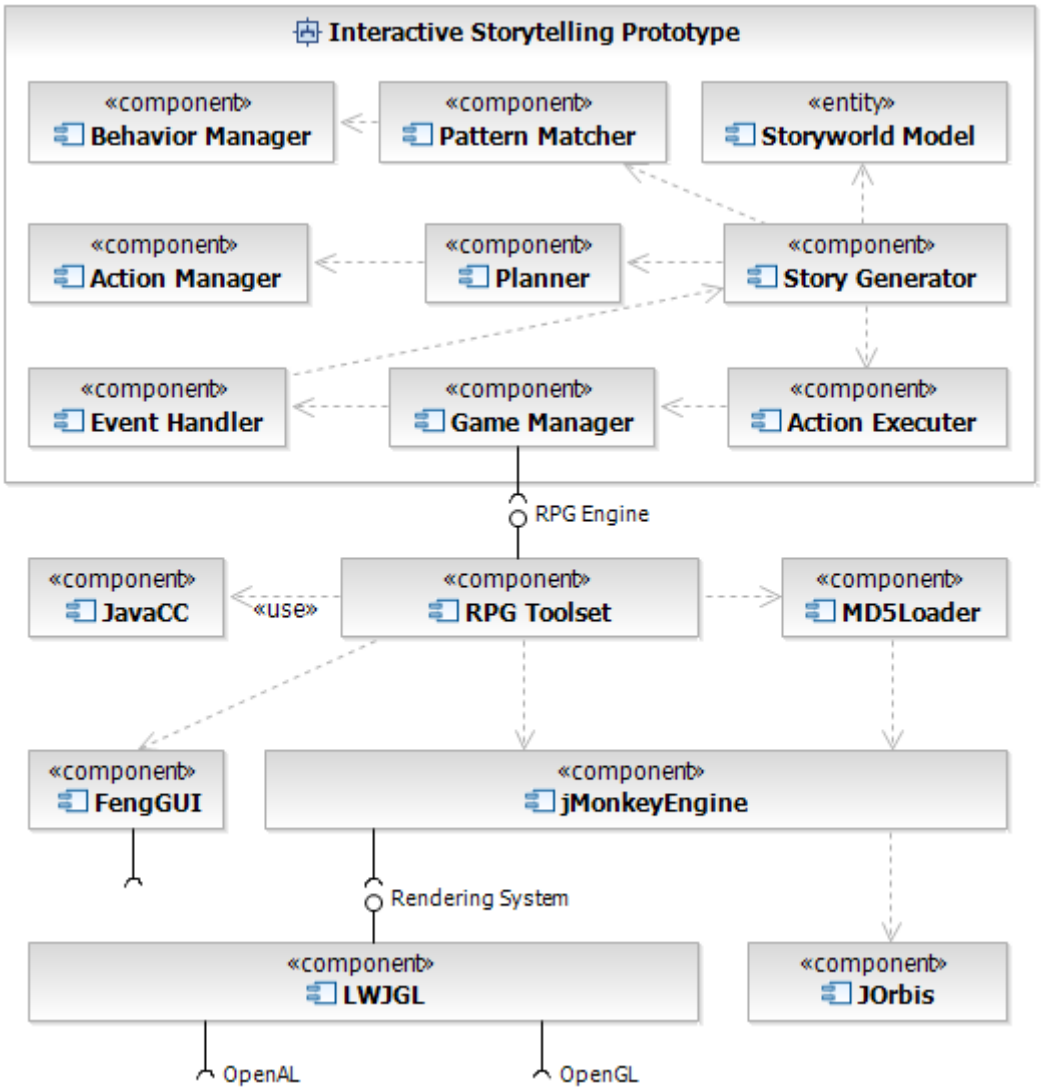


Figure 51. Architecture of the implemented prototype, including utilized third-party components.

As depicted above, the implemented interactive storytelling prototype has of a total of nine components. The core of the prototype consists of the following two components:

- **Storyworld Model** reads from an input file and provides access to all custom story elements and domain-specific element types (described throughout chapter 7) that define a storyworld (see the appendices for an input file with an example storyworld).
- **Story Generator** implements the story generation phase (described in section 8.3) and generates dynamic interactive stories based on the defined storyworld.

The *Character Behavior Layer* (described in section 7.3) is made of these two components:

- **Behavior Manager** reads and manages access to all behavioral patterns defined by the storyteller in the input file.
- **Pattern Matcher** is responsible for finding behavioral patterns that match the current state of the storyworld (see the appendices for the source code of the algorithm).

The following two components form the *Action Planning Layer* (described in section 7.2):

- **Action Manager** manages access to all available atomic actions and all simple and complex actions defined by the storyteller in the input file.
- **Planner** is the component that is responsible for creating and managing plans for the player and all non-player characters consisting of complex, simple and atomic actions. This component also handles replanning of existing characters' plans, as described in section 8.3.

The remaining three components belong to the *Visualization Layer* (described in section 7.1):

- **Action Executer** is responsible for carrying out and executing planned actions in the game environment. Actions planned for non-player characters are added to their list of actions to be carried out, whereas actions planned for the player are presented to him as multiple options via the game interface.
- **Event Handler** notifies the *Story Generator* component of important in-game events that have taken place, i.e. narrative actions that have been carried out by characters (the player and all non-player characters).
- **Game Manager** serves as an interface to the utilized computer role-playing game engine, since the proposed and prototyped interactive storytelling solution is designed to be independent of the actual game engine. The implemented prototype uses RPG Toolset (see below); however, other game engines can also be used.

As shown in Figure 51, our software prototype utilizes a tile-based role-playing game engine, which is a part of RPG Toolset [49], an open-ended set of tools for creating 3D computer role-playing games. RPG Toolset uses the Java Compiler Compiler (JavaCC) [53] for parsing script commands, MD5Loader [61] for loading models and animations saved in MD5 format, and FengGUI [51] for displaying in-game graphical user interface widgets.

The utilized role-playing game engine, i.e. RPG Toolset, runs atop of jMonkeyEngine [54], an open-source high performance scenegraph-based graphics engine written in the Java programming language. The engine uses the Lightweight Java Game Library (LWJGL) [57] as the rendering system, which provides native access to both OpenGL (Open Graphics Library) [63] and OpenAL (Open Audio Library) [62]. In addition, jMonkeyEngine uses JOrbis [55] for playback of Ogg Vorbis audio files.

For technical information regarding selected parts of the implemented interactive storytelling prototype, consult the technical documentation in the appendices at the end of this document.

9.2 Example Screenshots

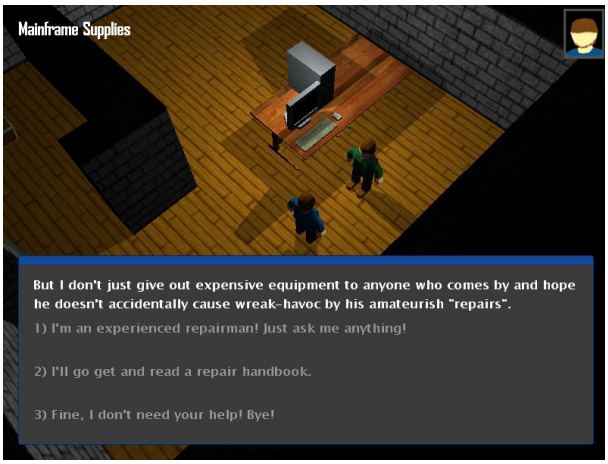
Below are screenshots from an example prototype storyworld in the educational domain related to the history of computing and programming basics (described throughout chapter 8 and located in the technical documentation included in the appendices).

In the example storyworld, the player is asked by a non-player character named Tim to repair a malfunctioning mainframe computer, either by reprogramming it (see Figure 52c) or by repairing it with a spare circuit board (see Figure 52i), which he can either steal (see Figure 52h) or get from John, a non-player character, upon becoming trustworthy by successfully answering a mainframe-related question (see Figure 52g) or by reading a mainframe handbook (see Figure 52f).

(a) The player wandering in the wilderness.

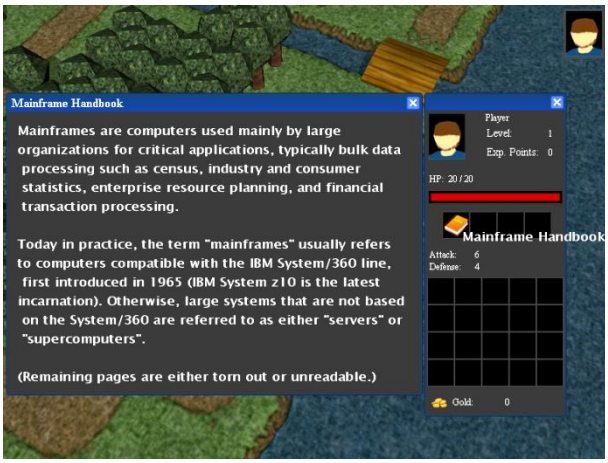
(b) Tim asking the player to fix the mainframe.

(c) The player is trying to reprogram the mainframe.



(d) The player is asking John for a spare circuit board.

(e) The player found a mainframe handbook.



(f) The player is reading the mainframe handbook.

(g) The player is answering John's question.

(h) The player is stealing a spare circuit board.

(i) The player fixing the mainframe with a circuit board.

(j) Tim thanking the player for fixing the mainframe.

Figure 52. Screenshots from the prototyped storyworld.

10 EVALUATION

Our goal was to evaluate the implemented interactive storytelling prototype both formally and empirically. Various storyworld examples were considered for evaluation of the prototype, as described below.

10.1 Example Storyworld Candidates

Examples that serve as demonstrations of interactive storytelling systems are most commonly custom folktale or fairytale storyworlds that best demonstrate the capabilities of that particular system. The goal of this project was to evaluate the implemented prototype based on a storyworld that, on the one hand, demonstrates the capabilities of the proposed approach to interactive storytelling, and, on the other hand, is from an educational domain at the same time.

Following are various candidates of such educational storyworlds that may serve as the example used for evaluating the implemented prototype:

- **Basics of Programming Languages.** An educational storyworld that teaches the fundamentals of various programming languages. The player's goal is to correctly construct code fragments and basic language structures, e.g., an array or a loop.
- **Abstract Data Types.** The purpose of this educational storyworld is to teach the player what kind of abstract data types exist, when and under which circumstances they are used and how they are constructed.
- **Software Design Basics.** Since the process of designing software products involves a series of steps and tasks, it is possible to form a storyworld that teaches the fundamentals of software design. The player's task in this storyworld is to design a software product for a non-player character resembling a customer. The story varies in the possible paths the player can follow, e.g., he can opt to follow a classic waterfall model, use agile techniques or apply extreme programming for the task. Each methodology has different affects on the resulting solution that the player presents to the customer.
- **History of Computing.** A storyworld in the domain of teaching the history of computing through questions asked by non-player characters that the player must correctly answer in order to move the story forward.
- **Interactive User's Guide.** The purpose of this storyworld is to instruct players how to define input rules and data structures for the proposed interactive storytelling solution. The player's goal is to define his own custom storyworld with the assistance and guidance of non-player characters.

The first and fourth storyworld candidates were chosen as being the ones most suited for evaluating the implemented prototype. Consequently, both formal and empirical evaluation of the implemented prototype was based on an example educational storyworld in the domain related to the history of computing and programming basics (described throughout chapter 8; see section 9.2 for example screenshots).

10.2 Formal Evaluation

Formal evaluation of the implemented interactive storytelling prototype consists of analyzing the correlation between the number of input rules defining an example storyworld and the number of possible interactive story variations that may be generated according to these rules.

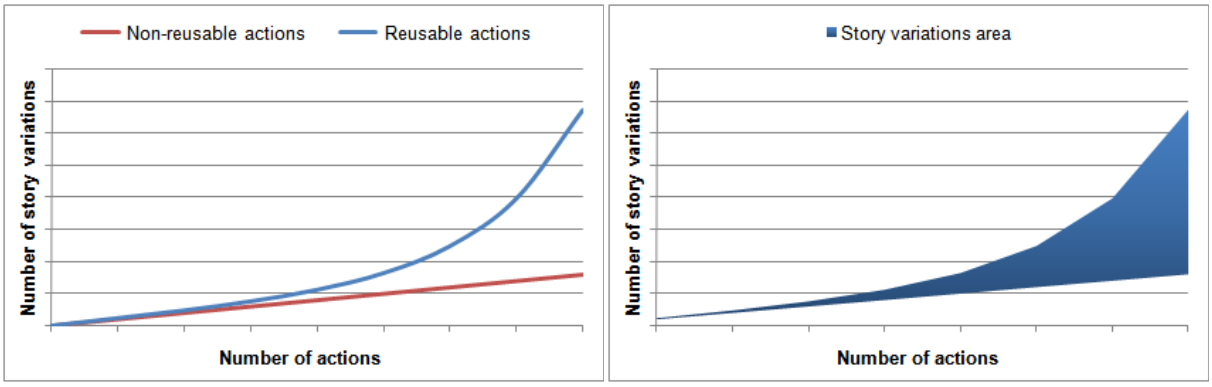
The story generation process described in detail in chapter 8 implies that the number of generated story variations is directly proportional to the number of defined input rules and data structures (e.g., actions, behavior patterns and others defined throughout chapter 7). Moreover, the initial state of the storyworld defined by the author (i.e. elements he placed in the storyworld such as items, objects, and non-player characters, including initial values of their defined properties) also affects the number of all possible generated story variations.

The formal evaluation of the implemented prototype described below focuses on actions (atomic, simple and complex), since they have the greatest impact (of all input rules) on the resulting number of all possible generated story variations.

Due to the nature of the proposed story generation process, it is not possible in general to calculate[26] the number of possible story variations only by knowing the number of defined actions. In order to be able to estimate the number of all generated story variations, we need to identify the *reusability* of the defined actions[27] and categorize them into two groups:

- **Reusable actions** with a small number of preconditions and high number of effects.
- **Non-reusable actions** having many preconditions and only a few effects.

Based on this classification, we are able to estimate the correlation between the number of defined actions and the number of generated story variations in an ideal[28] storyworld. By defining more non-reusable actions, the number of story variations increases linearly at best, whereas new reusable actions increase the number of story variations exponentially – see Figure 53a. In general, the variable number of generated story variations in relation to the number of defined actions (of any reusability) is represented by the plot area depicted in Figure 53b.



(a) Increasing of story variations per action reusability. **(b)** Plot area containing all generated story variations.

Figure 53. Correlation between input actions and generated story variations.

Considering the prototyped example storyworld (described throughout chapter 8), the number of generated story variations is equal to 2^4, since the storyworld operates on a single behavior pattern according to which a plan is generated with a maximum of 4 paths represents all possible story variations (see page 49).

[26] This is why many interactive storytelling solutions are formally evaluated by performing simulations on large amounts of input data, what is considered for future work.
[27] This classification is applicable to behavior patterns as well, which have a similar impact on the number of generated story variations as actions.
[28] A storyworld with *complete action coverage*, i.e. for every custom defined property of items, objects and non-player characters (see page 37) there exists an action that alters this particular property in its effects.

In other words, in order to be able to generate 16 possible story variations using the implemented prototype, the author needed to define a storyworld of only 6 simple actions and 1 behavioral pattern. Without using the interactive storytelling prototype, the author would have to script all 16 story variations manually, what is a considerable amount of work (as opposed to defining only 7 input rules) and requires to learn the scripting language of the utilized graphics engine.

In addition, by introducing a reusable action into the example storyworld, the total number of generated story variations increases exponentially to 32 (2^5). A significant increase of generated story variations can be also achieved by, for example, adding additional behavioral patterns to the example storyworld or by altering its initial state.

10.3 Empirical Evaluation

The implemented interactive storytelling prototype was also evaluated empirically by test players who played through the example educational storyworld and afterwards evaluated the generated interactive stories by filling out questionnaires containing questions regarding various dimensions of the generated stories.

There is no common metric for empirically evaluating player's experience in interactive storytelling systems. As suggested by Murray [33], player's experience can be evaluated in the following three dimensions:

- **Immersion** is the ability of the storyworld to seem realistic[29] for players, i.e. to involve them in the story's environment and to let them interact with it.
- **Agency** is the feeling that empowers players to commit narrative actions in order to fulfill their desired goals, and hence affect the course of the stories.
- **Transformation** denotes the variety of ways the stories can develop, since different players may experience different story paths mirroring their narrative actions and personalities.

All of the three dimensions mentioned above were addressed by a set of questions[30] contained in the questionnaires that were handed out to test players. The questionnaires also contained questions that evaluated the educational potential of the generated stories and interactive storytelling as a whole.

The implemented prototype was empirically evaluated by 14 test players. Resulting data of the evaluation is located in the appendices, whereas findings most significant for the scope of this work are discussed below, grouped by the evaluated dimensions they relate to.

Immersion. Half of all test players perceived the environment of the generated stories as being rather made-up than realistic, while the other half considered the environment being close to neutral. This is probably caused by the fact that the prototype storyworld uses cartoon-like models and textures that mimic a fairytale and at the same time mixes modern electronic equipment, such as mainframes, resulting in violating the players' suspension of disbelief. On the other hand, half of the players still found the setting of the generated stories believable.

[29] Realistic as in believable, but not necessarily real. Immersion depends on whether players experience a *suspension of disbelief*, as defined on page 13.

[30] See a copy of the questionnaire located in the appendices for texts of all questions and for all answers corresponding to each question.

Agency. Around 86% of players felt that their committed actions influenced the generated stories, what is a positive result. However, 71% of test players would welcome more narrative choices, what is a consequence to the fact that the example storyworld operates on a relatively small number of rules, i.e. actions. This can be overcome by introducing more rules into the example storyworld.

Transformation. Only 21% of players found the plot of the generated stories surprising, what may be a consequence to the fact that many test players have seen a demonstration of the example storyworld before they have actually evaluated it. In addition, 64% test players expressed that the generated stories developed only in a small number of paths and 43% of players found the paths repetitive. Both results can be improved by adding more rules into the example storyworld.

Educational Potential. Almost 72% of test players have learned almost nothing new by playing through the example storyworld. This negative result was caused[31] by the storyworld not giving any hints or feedback to players explaining where they made mistakes while trying to construct valid program code (in order to repair the malfunctioning mainframe computer) or what was the correct answer (when answering a question regarding mainframes asked by a non-player character). On the other hand, 50% of test players expressed that they would learn something interesting to some extent if the storyworld had contained the lacking feedback.

In addition, 86% of all test players can imagine the played games set in a different domain, whether educational or not, thus recognizing and acknowledging the domain-independence of the interactive storytelling prototype. What is more, 93% of players consider interactive storytelling a perfect medium for educating, thus confirming the practical usability of the proposed approach to interactive storytelling in the field of education.

Overall Impression. Overall, only one player found the evaluated storyworld rather uninteresting, and the majority of players remained interested in playing through interactive stories generated by the implemented prototype from the evaluated educational storyworld or from a different storyworld.

To sum up, empirical evaluation of the implemented prototype on an example educational storyworld brought positive results and invaluable feedback towards future development of the implemented prototype. Most of the negative results were caused by the evaluated storyworld operating on a relatively small number of rules based on which interactive stories were generated.

[31] Many players expressed this reason in their feedback and this issue is planned to be addressed in newer versions of the prototype.

11 CONCLUSIONS

The main contribution of this work is in devising a new and innovative approach in the field of interactive storytelling that makes use of techniques common in this field in such a unique way that enables to programmatically generate interactive stories with computer role-playing games being the storytelling medium. Such idea has not been realized by any of the solutions existing today.

A throughout analysis of existing interactive storytelling solutions and approaches has been carried out and based on the results a set of goals were identified for the resulting interactive storytelling concept. Moreover, computer role-playing games were recognized as an ideal medium for generating interactive stories.

According to these results and findings, a novel concept of generating interactive stories in computer role-playing was proposed and described in detail. Bridging the gap between the field of interactive storytelling and computer role-playing games required to create a new form of input data on which the proposed concept operates. In contrast to existing solutions, the proposed input rules are easily visualizable and kept as intuitive as possible for authors, i.e. storytellers, and at the same time enable to generate domain-specific interactive stories.

Selected parts of the proposed approach to interactive storytelling were submitted to three conferences (relevant papers are included in the appendices) and based on this approach a software prototype was implemented in the domain of generating educational interactive stories related to the history of computing and basics of programming languages.

The interactive storytelling prototype that we have implemented generates educational interactive stories based on domain-specific rules read from an input file, and presents them to the player through a computer role-playing game. The generated stories progress dynamically by reacting to narrative actions committed in-game by the player.

The resulting prototype was evaluated formally by analyzing the correlation between the number of input rules and the number of generated interactive stories, and empirically by test players who rated their experience from playing through the example educational storyworld by filling out questionnaires with questions covering various aspects of the generated stories.

Future work lies in developing a visual editor for the input rules and data structures on which the devised interactive storytelling approach operates, and in introducing advanced heuristics into the story generation process that would bring non-deterministic behavior to the generated interactive stories, and make them less predictable and more exciting. In addition, by adding more examples and questions into the prototyped storyworld, we could evaluate the learning impact of the generated educational stories in a throughout and more detailed manner.

Last, but not least, combining the edifying potential of interactive stories with today's popularity of computer role-playing games results in making the field of interactive storytelling more popular and attractive for a broader audience, and at the same time brings beneficial use to modern computer role-playing games.

REFERENCES

[1] ACHTERBOSCH, L., PIERCE, R., SIMMONS, G.: Massively Multiplayer Online Role-playing Games: The Past, Present, and Future. *Computers in Entertainment*, Vol. 5, No. 4 (2008) 1-33.

[2] ARISTOTLE: *Poetics*. Penguin Classics, (1997).

[3] BARROS, L. M., MUSSE, S.R.: Planning Algorithms for Interactive Storytelling. *Computers in Entertainment*, Vol. 5, No. 1 (2007).

[4] BARTON, M.: *Dungeons and Desktops: The History of Computer Role-playing Games*. A K Peters Ltd, Wellesley (2008).

[5] BETHESDA GAME STUDIOS: *Fallout 3*. ZeniMax Media, Rockville (2008).

[6] BETHESDA GAME STUDIOS: *The Elder Scrolls IV: Oblivion*. 2K Games, New York (2006).

[7] BIELIKOVÁ, M., DIVÉKY, M., JURNEČKA, P., KAJAN, R., OMELINA, Ľ.: Automatic Generation of Adaptive, Educational and Multimedia Computer Games. *Signal, Image and Video Processing*, Springer, Vol. 2, No. 4 (2008) 371-384.

[8] CAMPBELL, J.*: The Hero with a Thousand Faces*. Princeton University Press, Princeton (1972).

[9] CAVAZZA, M., CHARLES, F., MEAD, S.J.: Emergent Situations in Interactive Storytelling. In *Proc. of the 2002 ACM Symposium on Applied Computing*, ACM, New York (2002), 1080-1085.

[10] CAVAZZA, M., CHARLES, F., MEAD, S.J.: Planning Characters' Behaviour in Interactive Storytelling. *Journal of Visualization and Computer Animation*, Vol. 13, No. 2 (2002) 121-131.

[11] CAVAZZA, M., CHARLES, F., MEAD, S.J.: Sex, Lies, and Video Games: An Interactive Storytelling Prototype. In *Proc. of the 2002 AAAI Spring Symposium*, AAAI Press, Menlo Park (2002), 13-17.

[12] CAVAZZA, M., LUGRIN, J-L., PIZZI, D. et al.: Madame Bovary on the Holodeck: Immersive Interactive Storytelling. In *Proc. of the 15th International Conference on Multimedia*, ACM, New York (2007), 651-660.

[13] CAVAZZA, M., PIZZI, D.: Narratology for Interactive Storytelling: A Critical Introduction. In *Proc. of the Second International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, Springer, Berlin (2004), 72-83.

[14] CAI, Y., MIAO, C., TAN, A., SHEN, Z.: Fuzzy Cognitive Goal Net for Interactive Storytelling Plot Design. In *Proc. of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ACM, New York (2006).

[15] CHARLES, F., LOZANO, M., MEAD, S.J. et al.: Planning Formalisms and Authoring in Interactive Storytelling. In *Proc. of the First International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, Fraunhofer IRB Verlag, Darmstadt (2003).

[16] CHARLES, F., MEAD, S.J., CAVAZZA, M.: Interactive Storytelling: From Computer Games to Interactive Stories. *The Electronic Library*, Vol. 20, No. 2 (2002) 103-112.

[17] CHENG, K., CAIRNS, P.A.: Behaviour, Realism and Immersion in Games. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, ACM, New York (2005), 1272-1275.

[18] CIARLINI, A.E., POZZER, C.T., FURTADO, A.L. et al.: A Logic-Based Tool for Interactive Generation and Dramatization of Stories. In *Proc. of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ACM, New York (2005), 133-140.

[19] CRAWFORD, C.: *Chris Crawford on Interactive Storytelling*. New Riders Games, (2004).

[20] CRICHTON, M.: *Jurassic Park*. Ballantine Books, (1991).

[21] DENNING, S.: *The Leader's Guide to Storytelling: Mastering the Art and Discipline of Business Narrative*. Jossey-Bass, (2005).

[22] DENNING, S.: *The Springboard: How Storytelling Ignites Action in Knowledge-Era Organizations*. Butterworth-Heinemann, (2000).

[23] HOMER: *The Odyssey*. Penguin Classics, (1996).

[24] HOWARD, J.: *Quests: Design, Theory, and History in Games and Narratives*. A K Peters Ltd, Wellesley (2008).

[25] INWAGEN, P. VAN: The Incompatibility of Free Will and Determinism. *Philosophical Studies*, Vol. 27, No. 3 (1975) 185-199.

[26] JUUL, J.: *Half-Real: Video Games between Real Rules and Fictional Worlds*. MIT Press, Cambridge (2005).

[27] KOTTER, J.P.: *A Sense of Urgency*. Harvard Business School Press, Boston (2008).

[28] LAUREL, B.: *Computers as Theatre*. Addison-Wesley Professional, (2003).

[29] MALLON, B., WEBB, B.: Stand Up and Take Your Place: Identifying Narrative Elements in Narrative Adventure and Role-play Games. *Computers in Entertainment*, Vol. 3, No. 1 (2005).

[30] MATEAS, M., SENGERS, P. (Eds.): *Narrative Intelligence (Advances in Consciousness Research)*. John Benjamins Pub Co, (2003).

[31] MCKEE, R.: *Story: Substance, Structure, Style and the Principles of Screenwriting*. Methuen Publishing, London (1999).

[32] MURRAY, H.J.: From Game-Story to Cyberdrama. *FirstPerson: New Media as Story, Performance and Game*, MIT Press, Cambridge (2004), 2-11.

[33] MURRAY, H.J.: *Hamlet on the Holodeck: The Future of Narrative in Cyberspace*. The MIT Press, Cambridge (1998).

[34] ONG, T.J.: *Interactive Storytelling Engines*. Texas A&M University, 2006. Dissertation.

[35] PELLOWSKI, A.: *The World of Storytelling*. H. W. Wilson, New York (1991).

[36] PIZZI, D., CHARLES, F., LUGRIN J-L. et al.: Interactive Storytelling with Literary Feelings. In *Proc. of the Second International Conference on Affective Computing and Intelligent Interaction*, Springer, Berlin (2007).

[37] RICHARDS, D.: Is Interactivity Actually Important? In *Proc. of the 3rd Australasian Conference on Interactive Entertainment*, Murdoch University, Murdoch (2006), 59-66.

[38] ROLLINGS, A., ADAMS, E.: *Andrew Rollings and Ernest Adams on Game Design*. New Riders Games, (2003).

[39] SANDERCOCK, L.: Out of the Closet: The Importance of Stories and Storytelling in Planning Practice. *Planning Theory & Practice*, Vol. 4, No. 1 (2003) 11-28.

[40] SCHANK, R.: *Tell Me a Story: Narrative and Intelligence*. Northwestern University Press, Evanston (1995).

[41] STEPHENS, M.: *The Rise of the Image, the Fall of the Word*. Oxford University Press, Oxford (1998).

[42] SZILAS, N.: The Future of Interactive Drama. In *Proc. of the Second Australasian Conference on Interactive Entertainment*, Creativity & Cognition Studios Press, Sydney (2005), 193-199.

[43] TARAU, P., FIGA, E.: Knowledge-Based Conversational Agents and Virtual Storytelling. In *Proc. of the 2004 ACM Symposium on Applied Computing*, ACM, New York (2004), 39-44.

[44] THEUNE, M., FAAS, S., NIJHOLT, A. et al.: The Virtual Storyteller: Story Creation by Intelligent Agents. In *Proc. of the First International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, Fraunhofer IRB Verlag, Darmstadt (2003), 204-215.

[45] TOLKIEN, J.R.R.: *The Lord of the Rings*. Houghton Mifflin, Boston (1988).

[46] TYCHSEN, A.: Role Playing Games – Comparative Analysis Across Two Media Platforms. In *Proc. of the Third Australasian Conference on Interactive Entertainment*, Murdoch University, Perth (2006), 75-82.

[47] WALT DISNEY PICTURES: *The Lion King*. The Walt Disney Company, (1994).

[48] WALTON, K.: Fearing Fictions. *Journal of Philosophy*, Vol. 75, No. 1 (1978) 5-27.

Internet Sources

[49] ARNOLD, J.: *RPG Toolset*. http://www.rpgtoolset.net/ [2009-05-12]

[50] COSTIKYAN, G.: *Where Stories End and Games Begin*. http://www.costik.com/gamnstry.html [2009-05-12]

[51] *FengGUI*. http://www.fenggui.org/ [2009-05-12]

[52] INTERNATIONAL GAME DEVELOPERS ASSOCIATION: *Foundations of Interactive Storytelling*. http://www.igda.org/writing/InteractiveStorytelling.htm [2009-05-12]

[53] *Java Compiler Compiler (JavaCC)*. https://javacc.dev.java.net/ [2009-05-12]

[54] *jMonkeyEngine*. http://www.jmonkeyengine.com/ [2009-05-12]

[55] *JOrbis*. http://www.jcraft.com/jorbis/ [2009-05-12]

[56] KAHAN, S.: *The Power of Storytelling – An Interview with John Kotter*. http://www.sethkahan.com/Resources_LF003_2006KotterJohn.html [2009-05-12]

[57] *Lightweight Java Game Library (LWJGL)*. http://lwjgl.org/ [2009-05-12]

[58] MAHER, J.: *Toward Games that Matter: The Promise and Problems of the Storygame*. http://home.grandecom.net/~maher/if/storygames.htm [2009-05-12]

[59] MATEAS, M., STERN, A.: *Façade: An Experiment in Building a Fully-Realized Interactive Drama*. http://www.interactivestory.net/papers/MateasSternGDC03.pdf [2009-05-12]

[60] MATEAS, M., STERN, A.: *Façade*. http://www.interactivestory.net/ [2009-05-12]

[61] *MD5Loader*. http://gamedev.kproject.gr/index.php?n=Main.KProjectMD5 [2009-05-12]

[62] *Open Audio Library (OpenAL)*. http://www.openal.org/ [2009-05-12]

[63] *Open Graphics Library (OpenGL)*. http://www.opengl.org/ [2009-05-12]

[64] PRINCETON UNIVERSITY: *WordNet*. http://wordnet.princeton.edu/ [2009-05-12]

[65] RAMSEY, I.: *Origins of Storytelling*. http://falcon.jmu.edu/~ramseyil/storyorigins.htm [2009-05-12]

[66] SPECTOR, W.: *Next-Gen Storytelling Part One: What Makes a Story?* http://www.escapistmagazine.com/news/view/70852-Next-Gen-Storytelling-Part-One-What-Makes-a-Story [2009-05-12]

[67] *Storytron*. http://www.storytron.com/ [2009-05-12]

[68] UNIVERSITY OF TEESSIDE: *Research in Interactive Narratives*. http://www-scm.tees.ac.uk/users/f.charles/is_index.htm [2009-05-12]

PUBLICATIONS BY AUTHOR

1. DIVÉKY, M., JURNEČKA, P., KAJAN, R., OMELINA, Ľ.: S.M.I.L.E.: Smart Multipurpose Interactive Learning Environment. In *Proc. of IIT.SRC 2007 – 3rd Student Research Conference in Informatics and Information Technologies*, STU, Bratislava (2007), 389-390.
2. DIVÉKY, M., JURNEČKA, P., KAJAN, R., OMELINA, Ľ., BIELIKOVÁ, M.: Adaptive Educational Gameplay within Smart Multipurpose Interactive Learning Environment. In *Proc. of SMAP 2007 – 2nd International Workshop on Semantic Media Adaptation and Personalization*, IEEE Computer Society Press (2007), 165-170.
3. BIELIKOVÁ, M., DIVÉKY, M., JURNEČKA, P., KAJAN, R., OMELINA, Ľ.: Automatic Generation of Adaptive, Educational and Multimedia Computer Games. *Signal, Image and Video Processing*, Springer, Vol. 2, No. 4 (2008) 371-384.
4. BIELIKOVÁ, M., DIVÉKY, M., JURNEČKA, P., KAJAN, R., OMELINA, Ľ.: Learning with Smart Multipurpose Interactive Learning Environment. In *IFIP – International Federation for Information Processing, Vol. 281, Learning to Live in the Knowledge Society*, Springer, Boston (2008), 101-104.
5. DIVÉKY, M., BIELIKOVÁ, M.: An Approach to Interactive Storytelling and its Application to Computer Role-playing Games. In *Proc. of Znalosti 2009*, STU, Bratislava (2009), 59-70.
6. DIVÉKY, M.: Generating Interactive Stories in Computer Role-playing Games. In *Proc. of IIT.SRC 2009 – 5th Student Research Conference in Informatics and Information Technologies*, STU, Bratislava (2009), 243-250.
7. DIVÉKY, M., BIELIKOVÁ, M.: Generating Educational Interactive Stories in Computer Role-playing Games. In *Proc. of EC-TEL 2009 – 4th European Conference on Technology Enhanced Learning*, Springer Lecture Notes in Computer Science. To appear.

APPENDICES

A. Technical Documentation
    a. Fundamental Data Structures
    b. Sample Source Code
    c. Example Input Files
B. Empirical Evaluation Data
C. Znalosti 2009 Paper
D. IIT.SRC 2009 Paper
E. EC-TEL 2009 Paper
F. Electronic Medium Contents

A. TECHNICAL DOCUMENTATION

The following pages contain selected parts of the technical documentation related to the implemented interactive storytelling prototype.

Fundamental Data Structures

In order to implement the input rules and data structures (described in chapter 7) on which the proposed interactive storytelling solution operates, a class structure with a meta-level was designed and implemented.

The purpose of the meta-level is to explicitly define relationship constraints among implemented classes. Inclusion of the meta-level enables the authors to define their own custom, domain-specific rules and data structures according to which interactive stories are generated and played by players via computer role-playing games.

A subset of all classes forming the fundamental core of the implemented prototype is depicted in the following figures.

Figure A-1 shows a class diagram containing an analytical pattern that forms the described meta-level, which defines the abstraction of story elements, their type hierarchy and properties, including relationships constraints among them.

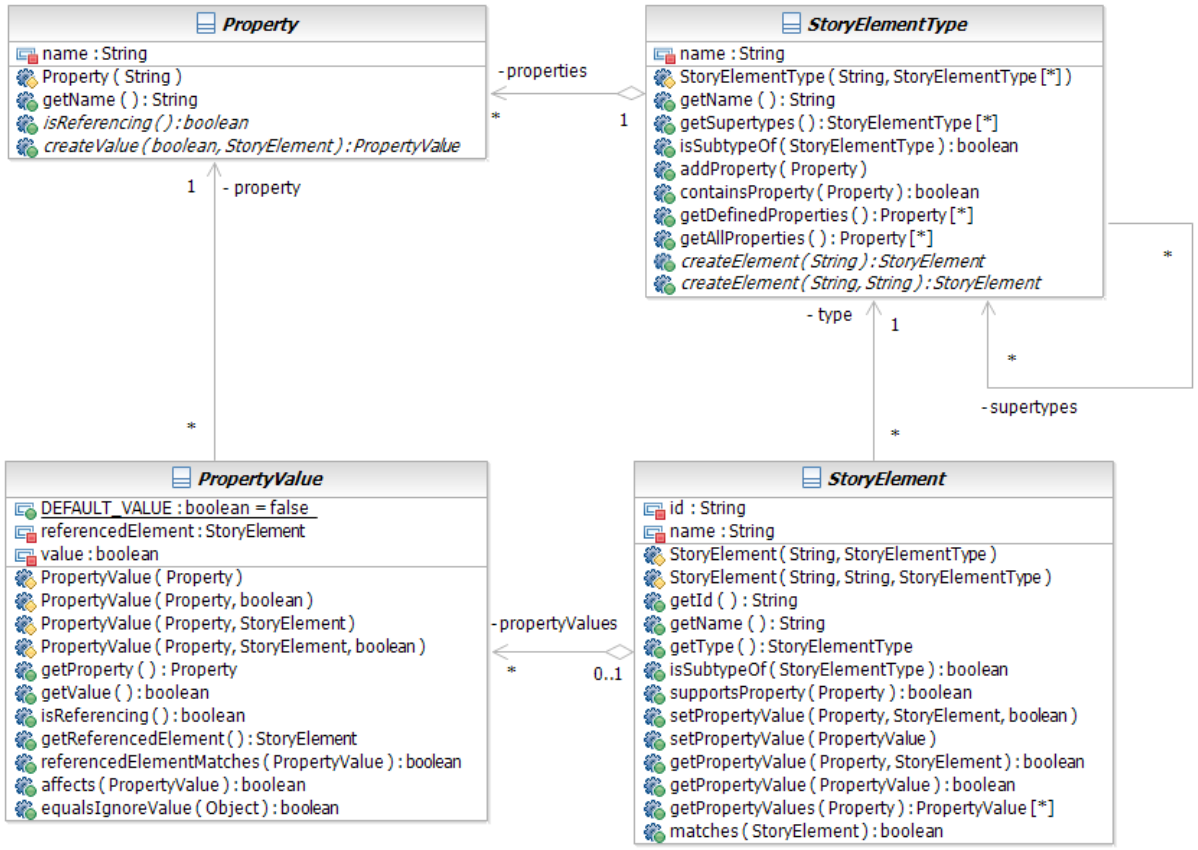


Figure A-1. Classes forming an abstraction of story elements, their type hierarchy and properties.

Relationships among classes that represent items, their subtype hierarchy (categorization), properties (tags) and binding to the meta-level classes are depicted in Figure A-2.

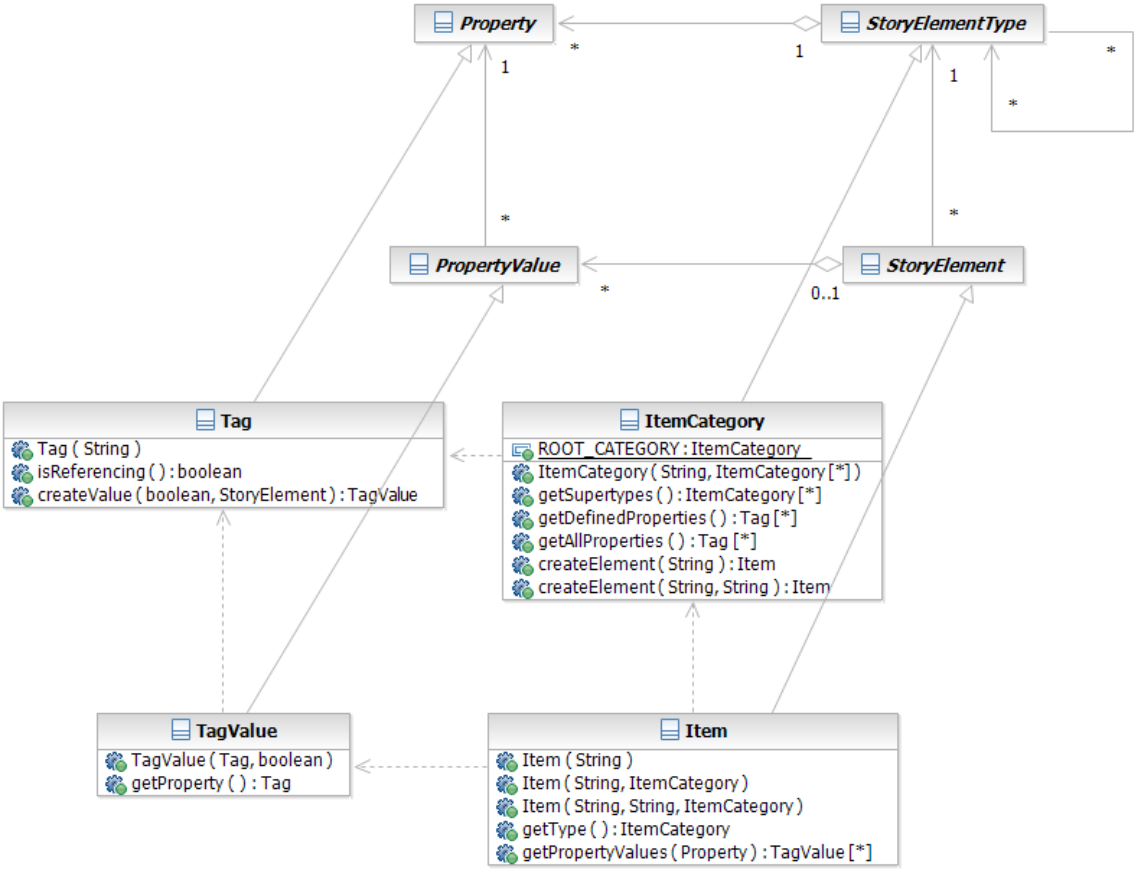


Figure A-2. Classes representing items, their subtype hierarchy and properties.

Figure A-3 contains classes representing objects, their subtype hierarchy, properties and binding to the meta-level classes.

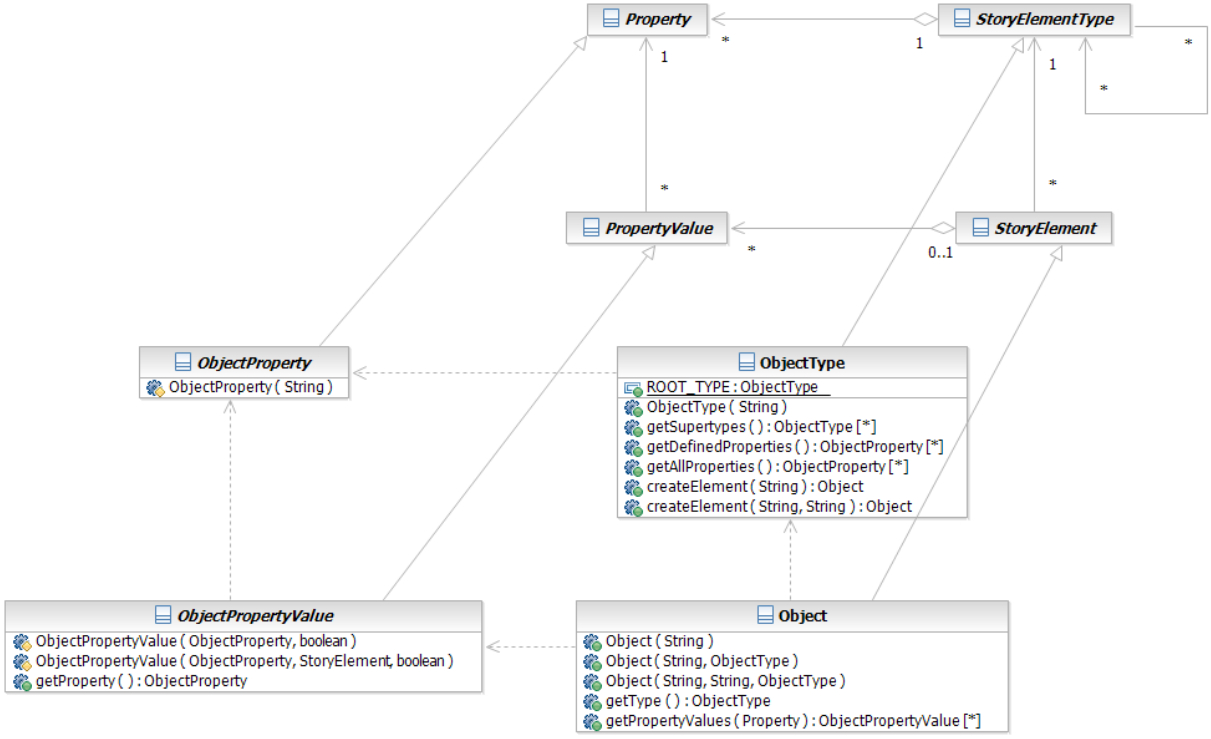


Figure A-3. Classes representing items, their subtype hierarchy and properties.

Classes representing user-definable object properties and their binding to classes describing objects are shown in Figure A-4.

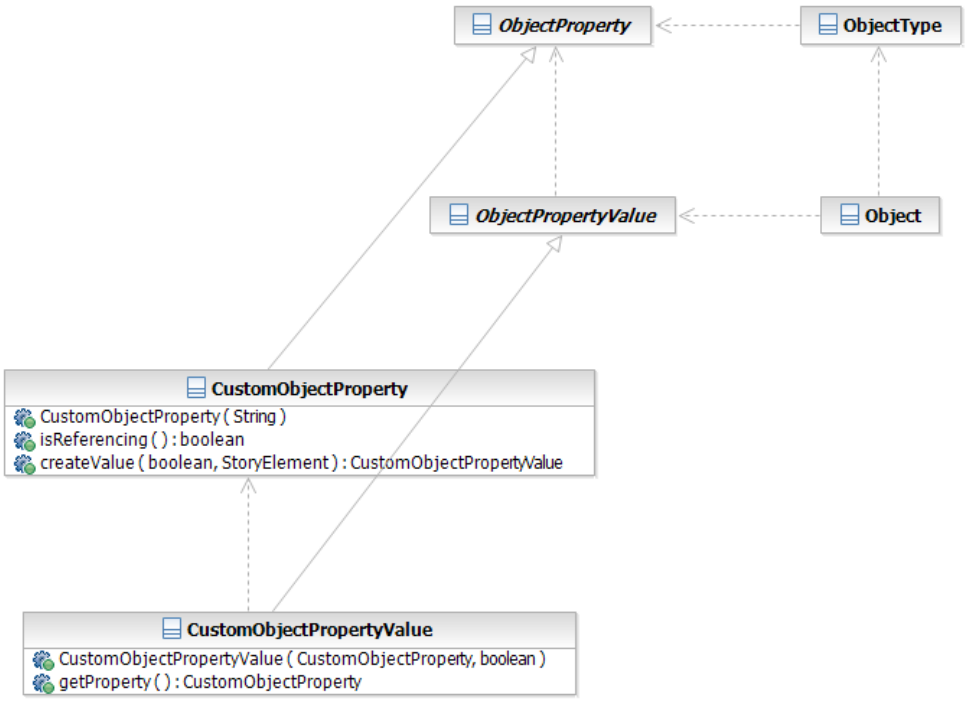


Figure A-4. Classes representing user-definable object properties.

Figure A-5 contains a class diagram with classes representing object containers (implemented as object subtypes) their hierarchy, standard property ("has") and binding to object classes.

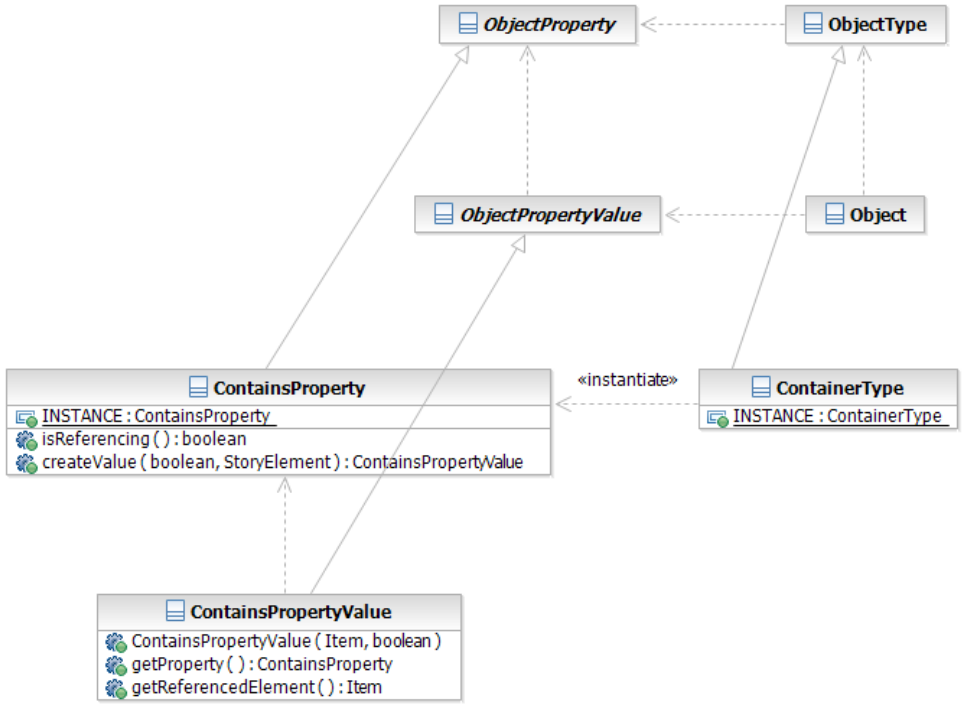


Figure A-5. Classes representing object containers, their subtype hierarchy and properties.

The class diagram in Figure A-6 shows the relationships among classes representing characters, their subtype hierarchy, properties and binding to the meta-level classes.

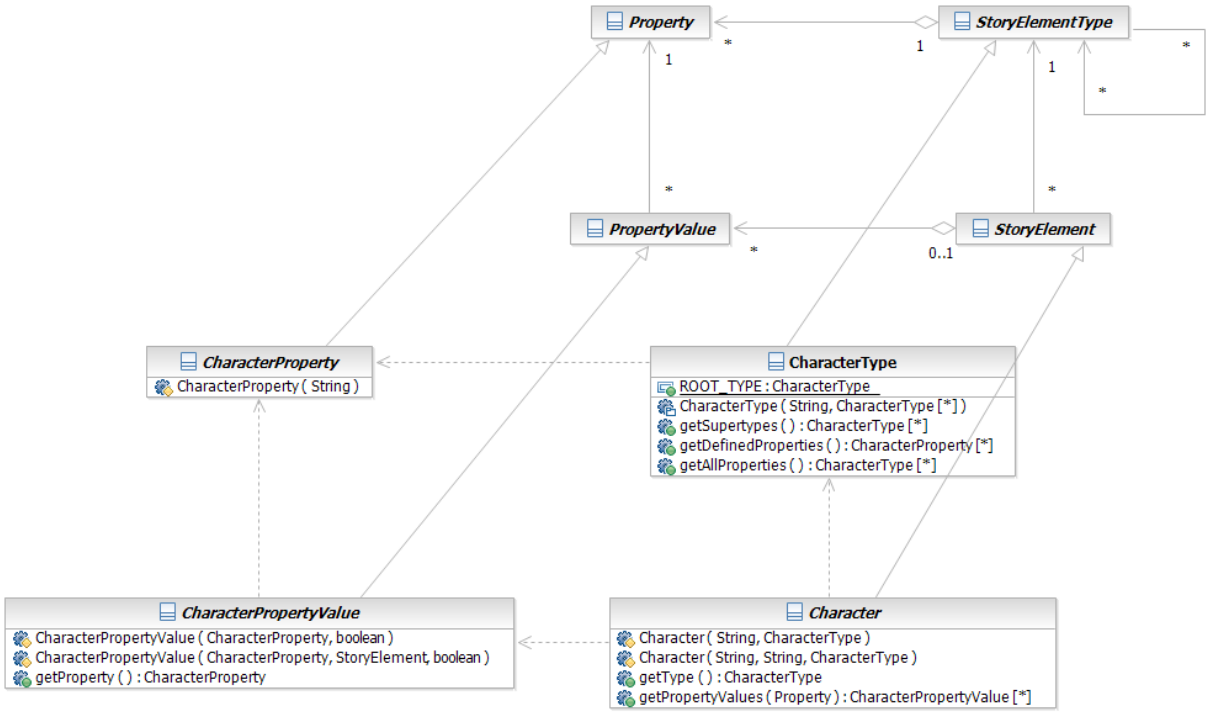


Figure A-6. Classes representing containers, their subtype hierarchy and properties.

Figure A-7 contains classes representing non-player characters and the player (implemented as character subtypes), their hierarchy, and binding to classes describing characters.

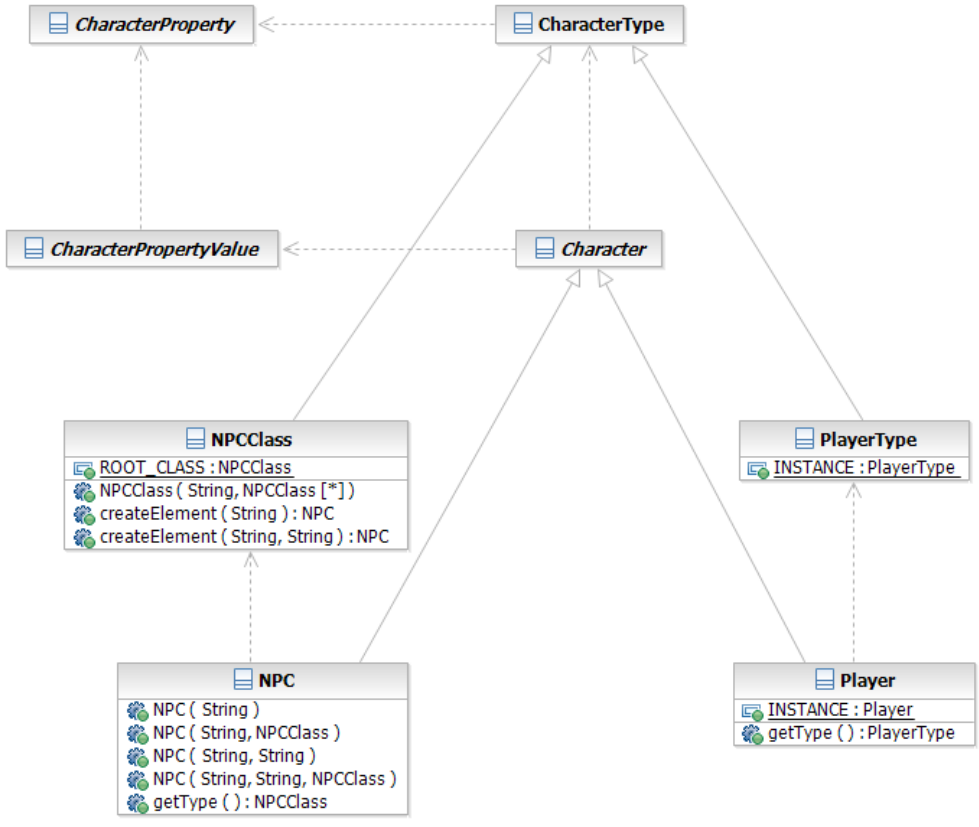


Figure A-7. Classes representing non-player characters and the player, their subtype hierarchy and properties.

Classes that represent character attributes and roles are shown in the class diagram in Figure A-8, along with their binding to character classes.

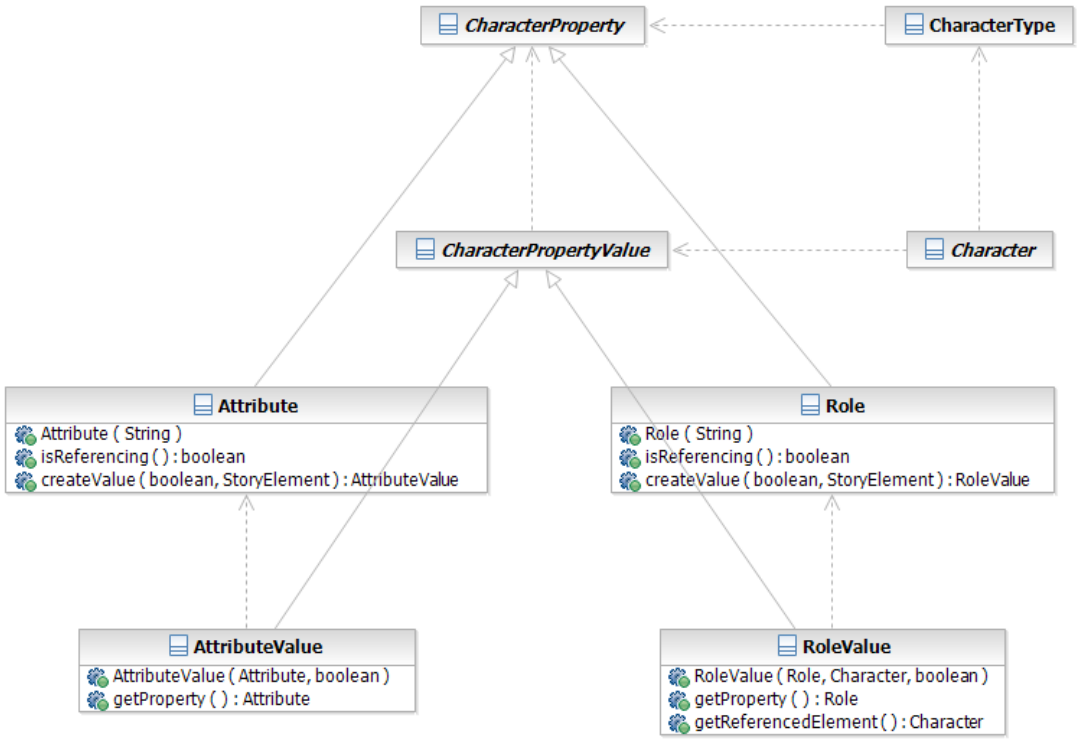


Figure A-8. Classes representing character attributes and roles.

Classes that represent character relationships and their binding to classes describing characters are contained in the class diagram in Figure A-9.

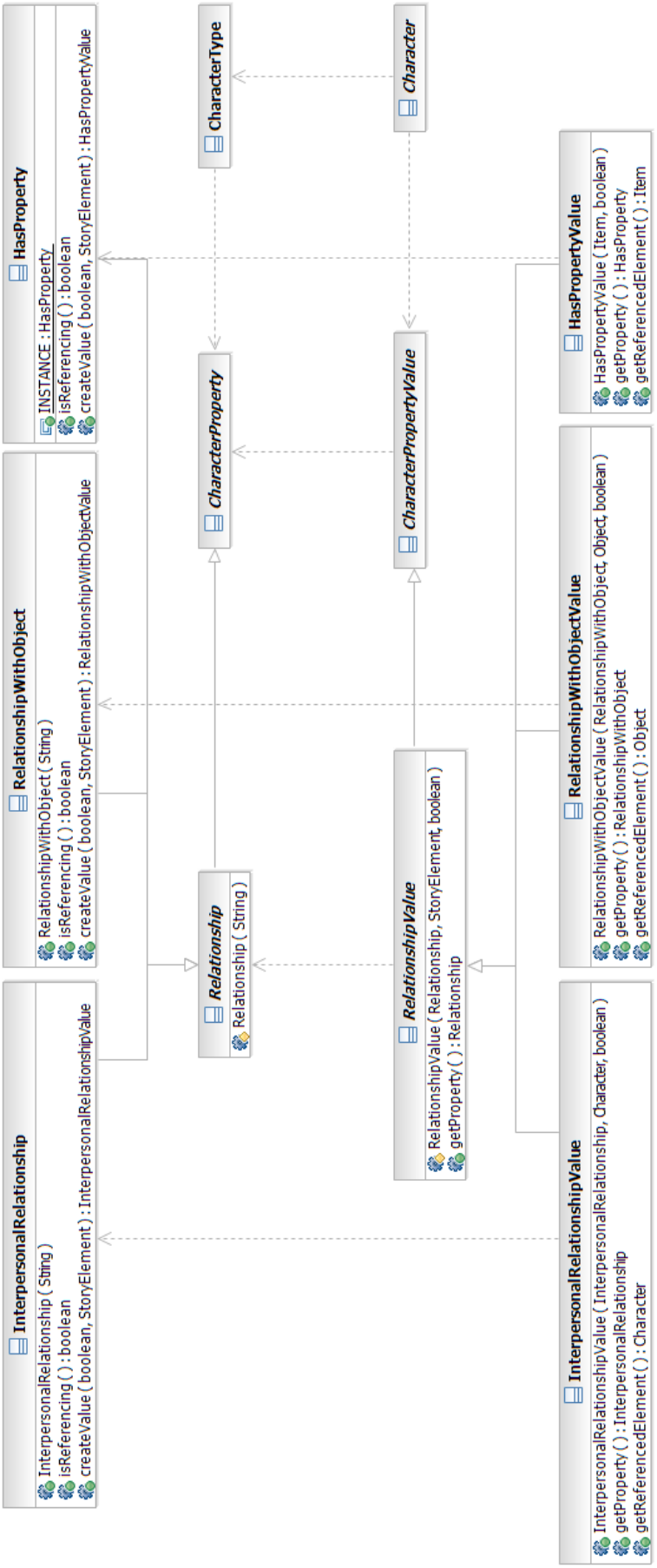


Figure A-9. Classes representing character relationships.

Sample Source Code

Following is the source code of the class `BehavioralPattern`, which is the part of the implemented software prototype that is responsible for identifying and matching behavioral patterns (see page 40) in storyworlds:

```
package net.metarpg.engine.patterns;

import …;

/**
 * Class representing a behavioral pattern that defines the circumstances leading to
 * a change in the behavior of characters from a narrative point of view. The set of
 * all behavioral patterns defines the conditional reasoning of all in-game characters.
 */
public class BehavioralPattern
{
   /** An optional text concisely describing the behavioral pattern. */
   private final String description;
   /** Template story elements that are part of the pattern and define its preconditions. */
   private final Set<StoryElement> templateElements = new HashSet<StoryElement>();
   /** A mapping of goals to template characters. */
   private final Map<Character, Goal> goals = new HashMap<Character, Goal>();

   /**
    * Default constructor.
    * @param desc - the pattern's description
    */
   public BehavioralPattern(String desc)
   {
      description = desc;
   }

   /**
    * Adds a template story element to the pattern.
    * @param template - story element to be added
    */
   public void addElement(StoryElement template)
   {
      templateElements.add(template);
   }

   /**
    * Adds multiple template story elements to the pattern.
    * @param templates - story elements to be added
    */
   public void addElements(StoryElement... templates)
   {
      for (StoryElement template : templates) addElement(template);
   }

   /**
    * Adds a goal for a particular template character.
    * @param ch - the template character to which the goal applies
    * @param element - the targeted template story element for which the goal applies
    * @param goalValue - the property value resembling the goal's desired effect
    * @throws NullPointerException - if the template character or targeted story element
    * are not a part of the behavioral pattern
    * @throws IllegalArgumentException - if the template character already has a different
    * goal defined
    */
   public void addGoal(Character ch, StoryElement element, PropertyValue goalValue)
   {
      if (!templateElements.contains(ch) || !templateElements.contains(element))
         throw new IllegalArgumentException("Character or element is not part of pattern");

      // If the character does not have any goal defined, create a blank one and store it
      if (!goals.containsKey(ch)) goals.put(ch, new Goal(element));

      // Set the character's goal, or throw an exception if it is already defined
      Goal goal = goals.get(ch);
      if (!goal.getSourceElement().equals(element))
         throw new IllegalArgumentException("Character already has another goal defined");
```

```
      else goal.addPropertyValue(goalValue);
   }

   /**
    * Returns the goal set for a template character.
    * @param ch - the template character
    * @return the template character's goal or <code>null</code> if it undefined
    * @throws NullPointerException - if the template character is <code>null</code>
    * @throws IllegalArgumentException - if the template character is not a part of the
    * behavioral pattern
    */
   public Goal getGoal(Character ch)
   {
      if (ch == null) throw new NullPointerException("Character cannot be null");
      else if (!templateElements.contains(ch))
         throw new IllegalArgumentException("Character or element is not part of pattern");
      else return goals.get(ch);
   }

   /**
    * Compares the behavior pattern to a storyworld and returns all found matches,
    * with duplicate matches removed.
    * @param story - the storyworld in which to find the pattern's matches
    * @return a list containing all found and distinct matches
    * @see Match
    */
   public List<Match> findMatches(Storyworld story)
   {
      // Create a map of all template elements to candidate storyworld elements and fill
      // it with all possible elements of the corresponding types
      HashMap<StoryElement, List<StoryElement>> candidateMap =
         new HashMap<StoryElement, List<StoryElement>>();
      for (StoryElement element : templateElements)candidateMap.put(element,
         new ArrayList<StoryElement>(story.getElementsOfType(element.getType())));

      // Iterate over all template elements and remove non-matching storyworld elements
      // from the list of possible candidates
      for (StoryElement template : templateElements)
      {
         for (Iterator<StoryElement> iter = candidateMap.get(template).iterator();
            iter.hasNext();)
         {
            StoryElement candidate = iter.next();
            if (!candidate.matches(template)) iter.remove();
         }
      }

      // Count the total number of found matches (a multiplication of the number of matched
      // elements to each template element)
      int count = 1;
      for (List<StoryElement> mappedElements : candidateMap.values())
         count *= mappedElements.size();

      // Create an empty list of matches
      List<Match> matches = new ArrayList<Match>();

      // Repeat for as many matches were found
      outer:
         for (int i = 0; i < count; i++)
         {
            // Create a new match
            Match match = new Match();
            // Store the match's sequence number
            int number = i;

            // Iterate over all template elements
            for (StoryElement template : templateElements)
            {
               // Get the matched storyworld elements for the present template element
               List<StoryElement> matchedElements = candidateMap.get(template);

               // Set the divider to the number of matched elements
               int divider = matchedElements.size();

               // Get the matched element at the selected position
               StoryElement mappedElement = matchedElements.get(number % divider);
```

```
                // If the element is already mapped, ignore the duplicate match and skip to
                // the next one
                if (match.hasMappedElement(mappedElement)) continue outer;

                // Otherwise add the mapping of the matched element to the present match
                match.addMapping(template, mappedElement);

                // Update the position number
                number /= divider;
            }
            // Add the newly-created match to the list
            matches.add(match);
        }

    // Return the list of found matches
    return matches;
}

/**
 * Returns the pattern's description.
 * @return description of this behavioral pattern
 */
public String getDescription()
{
    return description;
}

@Override
public boolean equals(Object obj)
{
    return this == obj;
}

@Override
public int hashCode()
{
    return super.hashCode();
}

/**
 * Class representing a match of a particular behavioral pattern in a particular
 * storyworld. A match represents a mapping of all behavioral pattern's template
 * elements to actual elements from a particular storyworld.
 */
public final class Match
{
    /** Mapping of the pattern's template elements to actual elements in the storyworld. */
    private final Map<StoryElement, StoryElement> mapping =
        new HashMap<StoryElement, StoryElement>();

    /**
     * Adds a mapping of a template element to an actual storyworld element.
     * @param template - the template element
     * @param mappedElement - the actual storyworld element
     */
    private void addMapping(StoryElement template, StoryElement mappedElement)
    {
        mapping.put(template, mappedElement);
    }

    /**
     * Returns whether a particular storyworld element is mapped in this mapping.
     * @param element - the actual storyworld element
     * @return <code>true</code> if the parameter element is mapped in this mapping,
     * <code>false</code> otherwise.
     */
    private boolean hasMappedElement(StoryElement element)
    {
        return mapping.values().contains(element);
    }

    /**
     * Returns an actual storyworld element mapped to the particular template element.
     * @param template - the template element
     * @return the actual storyworld element mapped to the template element
     * @throws IllegalArgumentException - if the template element is not part of the
```

```
         * owning behavior pattern
         */
        public StoryElement getMappedElement(StoryElement template)
        {
           if (!templateElements.contains(template))
              throw new IllegalArgumentException("Element is not part of behavioral pattern");
           else return mapping.get(template);
        }

        /**
         * Returns the goal assigned to the template character.
         * @param template - the template character
         * @return the goal set for the template character
         * @see BehavioralPattern#getGoal(Character)
         */
        public Goal getGoal(Character template)
        {
           return getOwningPattern().getGoal(template);
        }

        /**
         * Returns a mapping of all defined goals in the owning behavioral pattern to actual
         * storyworld characters.
         * @return a map of all goals to actual storyworld characters
         */
        public Map<Character, Goal> getAllGoals()
        {
           Map<Character, Goal> allGoals = new HashMap<Character, Goal>();

           for (Character templateChar : goals.keySet())
              allGoals.put((Character)getMappedElement(templateChar), getGoal(templateChar));

           return allGoals;
        }

        @Override
        public String toString()
        {
           StringBuilder sb = new StringBuilder();

           for (StoryElement template : mapping.keySet())
              sb.append(template.getName() + " -> " + mapping.get(template).getId() + "\n");

           return sb.toString();
        }

        @Override
        public boolean equals(Object obj)
        {
           if (this == obj) return true;
           if (obj == null || !(obj instanceof Match)) return false;

           Match other = (Match)obj;
           if (!getOwningPattern().equals(other.getOwningPattern())) return false;
           else return mapping.equals(other.mapping);
        }

        @Override
        public int hashCode()
        {
           final int prime = 31;

           int result = 1;
           result = prime * result + getOwningPattern().hashCode();
           result = prime * result + mapping.hashCode();

           return result;
        }

        private BehavioralPattern getOwningPattern()
        {
           return BehavioralPattern.this;
        }
     }
}
```

Example Input Files

Below are examples of XML input files containing rules and data structures (described in chapter 7) used by the example educational storyworld (described throughout chapter 8). These input files are read by the implemented prototype.

The first XML input file defines all necessary custom subtypes of story elements (see page 37), actions and behavioral patterns in the educational domain selected for prototyping:

```
<?xml version="1.0" encoding="UTF-8"?>
<domain name="History of Computing and Programming Basics">
   <types>
      <item_category id="books" name="Book"/>
      <item category id="mainframe_handbooks" name="Mainframe Handbook">
         <supertype id="books"/>
      </item_category>
      <item_category id="circuit_boards" name="Circuit Board"/>
      <object_type id="computers" name="Computer"/>
      <object type id="mainframes" name="Mainframe Computer">
         <supertype id="computers"/>
         <objectProperty name="malfunctioning"/>
         <objectProperty name="self-aware"/>
      </object_type>
      <all npcs>
         <relationship name="dislikes" ref="character"/>
         <relationship name="owns" ref="object"/>
      </all npcs>
      <player>
         <attribute name="trustworthy"/>
      </player>
   </types>
   <actions>
      <simple_action name="REPROGRAM" source_type="player" target_type="mainframes"
       base_action="SOLVE_PROGRAM_CODE">
         <preconditions>
            <target>
               <property name="malfunctioning" value="true"/>
            </target>
         </preconditions>
         <effects>
            <target>
               <property name="malfunctioning" value="false"/>
            </target>
         </effects>
      </simple_action>
      <simple_action name="REPAIR" source_type="player" target_type="mainframes"
       parameter type="circuit_boards" base_action="USE">
         <preconditions>
            <target>
               <property name="malfunctioning" value="true"/>
            </target>
         </preconditions>
         <effects>
            <source>
               <property name="has" ref="parameter" value="false"/>
            </source>
            <target>
               <property name="malfunctioning" value="false"/>
            </target>
         </effects>
      </simple_action>
      <simple_action name="STEAL" source_type="player" target_type="all_npcs"
       parameter_type="all_items" base_action="TAKE">
         <effects>
            <target>
               <property name="dislikes" ref="source" value="true"/>
            </target>
         </effects>
      </simple_action>
      <simple action name="GIVE CIRCUIT BOARD" source type="all_npcs" target_type="player"
       parameter type="circuit_boards" base_action="GIVE">
         <preconditions>
```

```
            <target>
               <property name="trustworthy" value="true"/>
            </target>
         </preconditions>
      </simple action>
      <simple_action name="BECOME_TRUSTWORTHY" source_type="player" target_type="all_npcs"
       base_action="ANSWER_QUESTION">
         <effects>
            <source>
               <property name="trustworthy" value="true"/>
            </source>
         </effects>
      </simple_action>
      <simple action name="READ" source_type="player" target_type="mainframe_handbooks"
       base action="EQUIP">
         <effects>
            <source>
               <property name="trustworthy" value="true"/>
            </source>
         </effects>
      </simple_action>
   </actions>
   <behavioral_patterns>
      <behavioral_pattern>
         <description>A malfunctioning mainframe's owner wants to have the mainframe
          fixed.</description>
         <elements>
            <object id="old_mainframe" type="mainframes">
               <property name="malfunctioning" value="true"/>
            </object>
            <npc id="scientist">
               <property name="owns" ref="old_mainframe" value="true"/>
            </npc>
         </elements>
         <goals>
            <goal>
               <character ref="scientist"/>
               <target ref="old mainframe">
                  <property name="malfunctioning" value="false"/>
               </target>
            </goal>
         </goals>
      </behavioral pattern>
   </behavioral_patterns>
</domain>
```

The second example XML input file contains the actual definition of the prototyped story-world, utilizing custom domain-specific data structures defined in the previous input file:

```
<?xml version="1.0" encoding="UTF-8"?>
<storyworld name="Prototype Storyworld">
   <items>
      <item id="circuit_board1" type="circuit_boards"/>
      <item id="mainframe_handbook1" type="mainframe_handbooks"/>
   </items>
   <objects>
      <object id="mainframe1" name="Tim's Mainframe" type="mainframes">
         <property name="malfunctioning" value="true"/>
      </object>
      <object id="wooden_chest1" type="chests">
         <property name="contains" ref="mainframe_handbook1" value="true"/>
      </object>
   </objects>
   <npcs>
      <npc id="tim" name="Tim">
         <property name="owns" ref="mainframe1" value="true"/>
      </npc>
      <npc id="john" name="John">
         <property name="has" ref="circuit_board1" value="true"/>
      </npc>
   </npcs>
</storyworld>
```

B. EMPIRICAL EVALUATION DATA

The implemented interactive storytelling prototype was evaluated empirically in the form of questionnaires filled out by test players after having played through the prototype storyworld. Figure B-1 shows the questionnaire handed out to test players with 14 questions according to which all players rated their experience in various dimensions (described in section 10.3).

The number of times I have played the game is:						
	1	2	3	4	5+	
I perceived the story's environment as being:						
Realistic	1	2	3	4	5	Made-up
I found the story's setting to be:						
Believable	1	2	3	4	5	Unreal
I felt that my actions influenced the story:						
Considerably	1	2	3	4	5	Not at all
The number of different narrative choices that I had were:						
Too few	1	2	3	4	5	Too many
The plot was:						
Predictable	1	2	3	4	5	Surprising
The number of paths the story went were:						
Too few	1	2	3	4	5	Too many
I consider the different story paths:						
Repetitive	1	2	3	4	5	Varied
While playing the game, I have learned:						
Nothing	1	2	3	4	5	A lot
What I have learned was:						
Interesting	1	2	3	4	5	Uninteresting
I can imagine the game set in a different (not necessarily educational) domain:						
Not at all	1	2	3	4	5	Definitely
As an educational medium, I consider interactive storytelling:						
Perfect	1	2	3	4	5	Inadequate
Overall, the game was:						
Interesting	1	2	3	4	5	Boring
I am interested in playing the game again:						
Not at all	1	2	3	4	5	Definitely

Figure B-1. Questionnaire filled out by test players.

The answers of all 14 test players are contained in Table B-1, where rows represent questions from the questionnaire and columns represent answers from test players.

Table B-1. Matrix with answers from all test players.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14
Q1	5+	2	2	3	5+	5+	5+	3	3	2	3	4	5+	3
Q2	4	2	4	4	3	2	2	5	4	5	4	3	2	3
Q3	2	2	3	3	2	2	2	4	5	3	2	2	3	4
Q4	2	1	2	2	1	1	2	1	1	3	2	2	4	1
Q5	2	3	2	2	2	2	2	2	3	2	2	3	2	4
Q6	3	3	3	3	2	2	4	1	2	1	3	4	2	5
Q7	2	2	2	3	3	2	2	3	1	2	1	3	2	3
Q8	4	3	3	2	2	–	5	4	2	1	4	2	2	4
Q9	3	2	2	2	3	2	1	2	2	2	2	2	3	3
Q10	2	2	3	4	2	4	–	4	2	3	2	2	2	4
Q11	5	4	4	4	5	4	3	4	5	5	4	4	5	3
Q12	1	1	1	1	1	3	2	1	1	2	1	2	1	2
Q13	2	2	2	2	3	3	2	2	2	4	2	3	3	2
Q14	2	4	3	5	2	2	4	2	2	4	4	3	4	3

Graphs showing an overview of all test players' answers per each question from the questionnaire are shown in Figure B-2 to Figure B-7.

The horizontal axis of all graphs contains the possible answers and the vertical axis contains the number of test players (out of a total of 14) who chose the particular answers in their questionnaires.

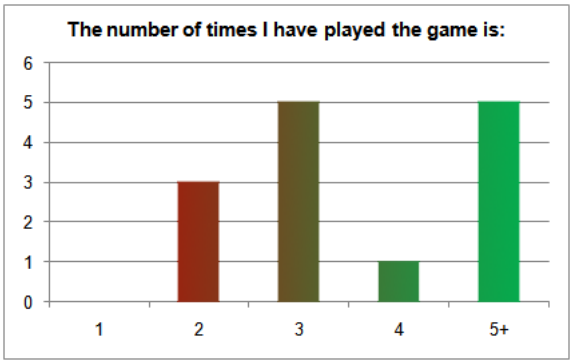


Figure B-2. Graph with answers to the first question.

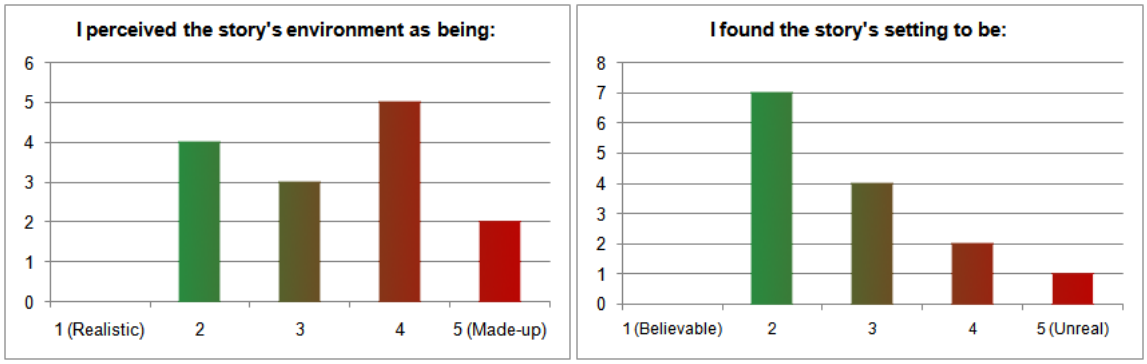


Figure B-3. Graphs with answers related to the dimension of immersion.

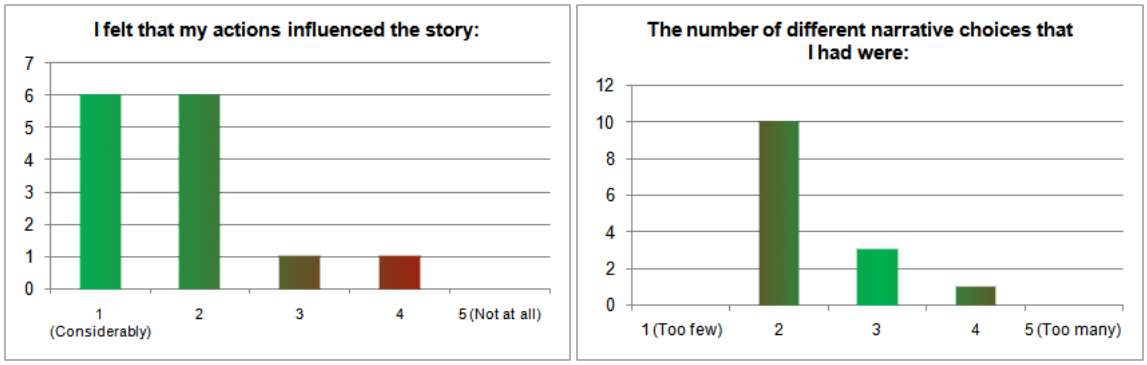


Figure B-4. Graphs with answers related to the dimension of agency.

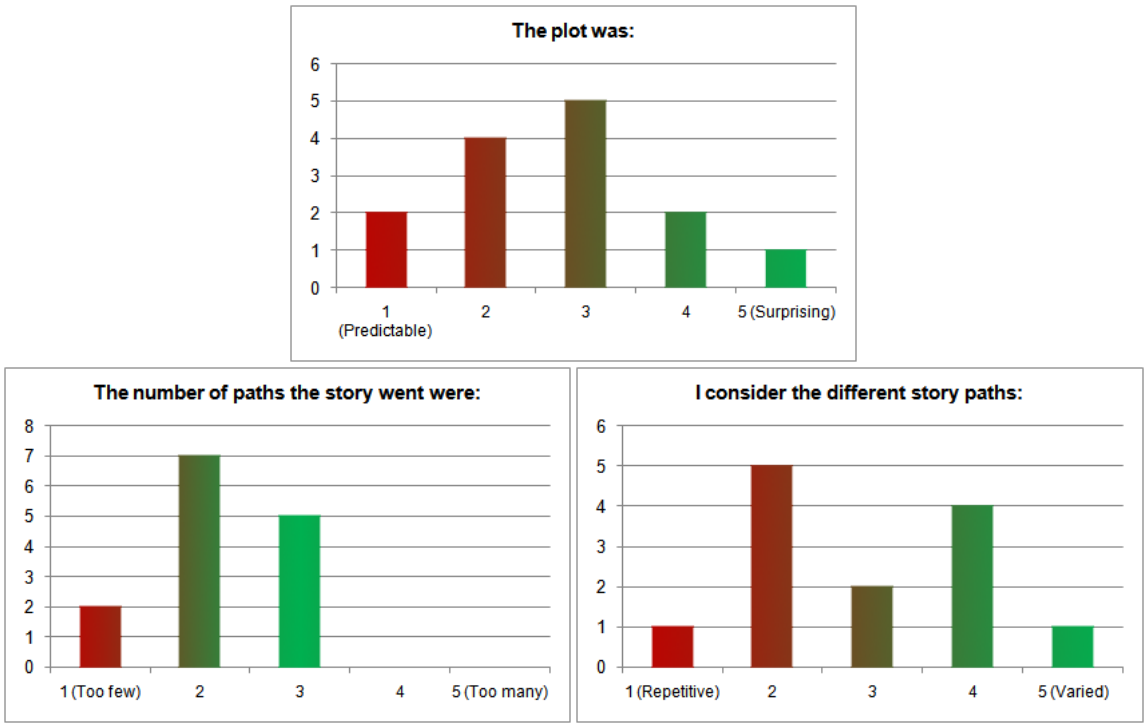


Figure B-5. Graphs with answers related to the dimension of transformation.

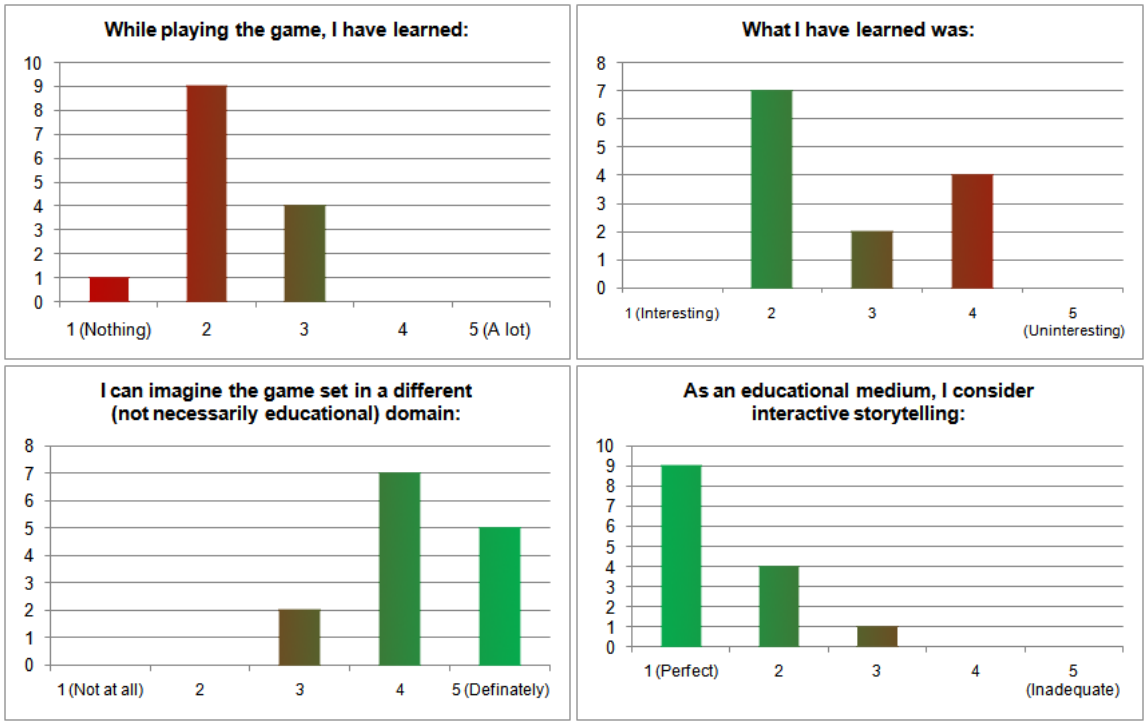


Figure B-6. Graphs with answers related to the educational potential of the generated stories and interactive storytelling as a whole.

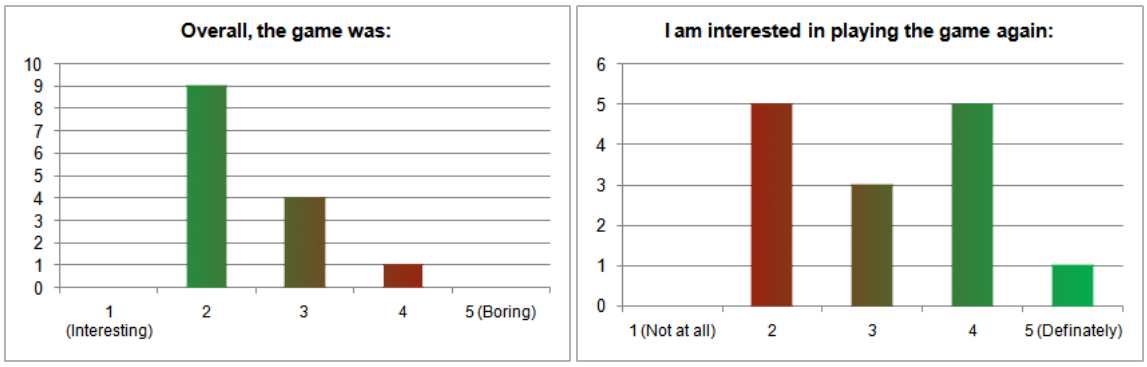


Figure B-7. Graphs with answers related to the overall impression of the generated stories.

C. ZNALOSTI 2009 PAPER

This appendix contains a paper that has been submitted to, accepted and presented at the Znalosti 2009 conference, which took place from February 4 until February 6, 2009 at the Faculty of Information Technology, Brno University of Technology, Czech Republic.

An Approach to Interactive Storytelling and its Application to Computer Role-playing Games

Marko Divéky, Mária Bieliková

Institution of Informatics and Software Engineering, Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, 842 16 Bratislava, Slovakia
markod@nextra.sk, bielik@fiit.stuba.sk

Abstract. Interactive storytelling is a rather young field of research that focuses on combining conventional stories with interactivity, resulting in immersing the reader inside the story by letting him shape the storyline in any desired direction through committing narrative actions. Despite the large amount of work that has already been done in this field, there have been only a few working solutions that found practical use other than being a proof-of-concept demonstrations. Today's popular computer games are an ideal medium for interactive storytelling systems, given their practically unlimited degree of interactivity and visual attractiveness. In this paper, we describe an innovative concept that aims to reduce the border between the field of interactive storytelling and modern computer games by making it possible to programmatically generate interactive stories in visually appealing computer role-playing games.

1 Introduction and Related Work

It is an undoubtable fact that stories and storytelling have been part of the human experience from the earliest days of language. The educational potential of stories and storytelling is widely utilized in business [8], where the term "business narrative" is often used as its synonym [7]. Some of the top thinkers and knowledge management leaders all over the world consider storytelling spread knowledge amongst knowledge workers in order to help them become much more productive and collaborate with one another [11].

Many aspects of stories have changed since the very first records of storytelling in ancient times. However, even in today's modern world, the fundamental idea behind conventional stories remains unaltered. Let us suppose that the storyteller seeks to communicate some truth or information (principle) to a desired audience. Instead of just telling the principle, the storyteller translates it into an instantiation (a story), then communicates the story to the audience, which in turn translates the instantiation back into the principle [6], as depicted in Figure 1.

Fig. 1. The process of telling a conventional story.

Interactive storytelling is best described as "a narrative genre on computer where the user is one main character in the story and the other characters and events are automated through a program written by an author. Being a character implies choosing all narrative actions for this character" [16].

Interactive storytelling differs from conventional stories in a way that the process of transforming the principle into the instance (story) is delegated to the computer. The computer transforms the principle into an interactive story (often called a *storyworld* [6]), which operates on rules rather than predefines events.

A single playing of a storyworld generates a single story. In other words, when a player goes thought a storyworld, he produces a linear sequence of events that form a story. Different playing of the storyworld can yield many different stories, but all of them share one common principle [6] – see Figure 2.

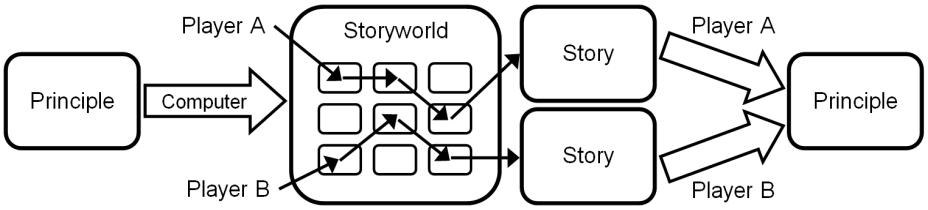


Fig. 2. The process of playing through an interactive story (storyworld).

Computer games were created as an art form based on the fundamental human activity of play, in contrast to storytelling, which originated from the need to communicate experience and knowledge among humans. Consequently, there are many differences between computer games, stories and movies as medium for storytelling that are related to the context of this paper:

- Stories and movies are about *empathy*, since the reader *identifies himself to a character* and can only "see" the consequences of the character's actions [12].
- Computer games are focused on *immersion*, because the player *embodies a character* and can "feel" the consequences of his actions throughout the world of the game [5].

Since computer games and game design have their purpose and structure different from stories and storytelling, researchers in the field of interactive storytelling often disregard computer games as a suitable storytelling medium [6]. Still, there are some that see them as the ideal form for storytelling, such as Janet Murray, who writes that computer games are "a new medium of expression [that] allows us to tell stories we could not tell before, to retell the age-old stories in new ways" [13].

The storytelling potential of computer games differs from one genre to the next. Throughout the many diverse genres of computer games available today, role-playing games (RPGs) and adventure games[1] have the most detailed and involving storyline, thus being the most appropriate genre for storytelling [15].

[1] Role-playing games were chosen in favor of adventure games mainly due to the increasing popularity of their online multiplayer versions, Massively Multiplayer Online Role-playing Games (MMORPGs) [2].

Up until now, there have been many different, more or less successful, interactive storytelling systems created, with the most notable systems being mentioned below.

Chris Crawford, a former computer game developer, has been working on his interactive storytelling system called Storytron [6] (formerly called Erasmatron) since 1991. The system uses a drama manager agent to control the plot of all created stories, which, despite being robust and flexible, are all text-based with limited visualization.

Marc Cavazza and Fred Charles have developed their interactive storytelling system with the use of Hierarchical Task Network (HTN) planning and Heuristic Search Planning (HSP) [4]. Moreover, the authors also take characters' emotions and feelings into account when generating stories [14]. However, users have only very limited ways of interacting with such stories, since they directly control no characters.

Other existing interactive storytelling systems use either artificial intelligence formalisms aimed at controlling agents in multi-agent systems, or custom heuristic algorithms. Both approaches, however, require complex knowledge in order to define storyworlds for such solutions. Moreover, almost all of the existing interactive storytelling systems fail either to visualize the generated stories or to make their course notably changeable by the player, resulting in the stories not being truly interactive.

2 Proposed Concept of Generating Interactive Stories in RPGs

The aim of the hereby-described concept is to programmatically generate dynamic and interactive stories with computer role-playing games (RPGs) as their medium, therefore combining the dynamic and enthralling storyworlds created by interactive storytelling with the visual appearance, gameplay and popularity of modern computer role-playing games. The presented concept can be broken into three logical layers, as depicted in Figure 3.

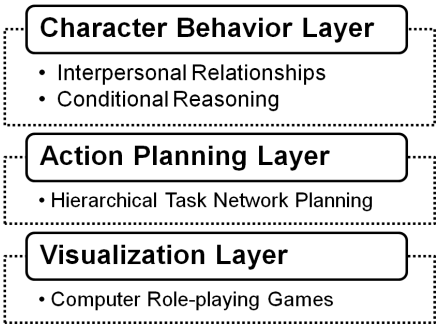


Fig. 3. Logical structure of the proposed concept.

All generated interactive stories initiate in the topmost logical layer named the *Character Behavior Layer*, since all stories are about people, despite the fact that references to them are often indirect or symbolic [6]. The behavior of characters results in creating plans and planning actions that move the story forward. The middle layer, called the *Action Planning Layer*, handles all the planning and replanning. All created plans eventually break down into actions that are visualized by the bottom-most layer called the *Visualization Layer*.

1.1 Visualization Layer

The lowest logical layer provides graphical visualization of the generated interactive stories to the players. This layer uses the concept of computer role-playing games as the visualization and storytelling medium. Below is a throughout analysis of aspects regarding computer role-playing games that are important for the scope of this work.

Themes. Games based on the role-playing genre are most often set in a fictional fantasy world closely related to classic mythology, or in a science fiction world set somewhere in the future. On the other hand, there are a number of role-playing games set in historical or modern settings [2].

Avatars. In role-playing games, a player controls one in-game character called the *avatar*, and uses him as an instrument for interacting with the world that the game takes place in. Some games also enable the player to control a group of characters in addition to the player's avatar [17].

Character Development. Besides having the option to fully customize the appearance of his in-game character, the player is allowed to choose various attributes, skills, traits and special abilities that his avatar will possess. These are given to players as rewards for overcoming challenges and achieving goals, most commonly for completing *quests* (see below). Character development plays, together with stories, a key role in today's computer role-playing games [2].

Quests. A *quest* in role-playing games can be defined as a journey across the game world in which the player collects *items* and talks to *non-player characters* (see below) "in order to overcome challenges and achieve a meaningful goal" [9]. Quests often require the player to find specific items that he needs to correctly use or combine in order to solve a particular task, and/or require the player to choose a correct answer from a number of given answers to a certain question [3]. Upon solving a quest, the player is often presented with a reward, which can have many forms – i.e. ranging from a valuable item to a new skill, trait or ability for the player's avatar.

Many quests are optional, allowing for freedom of choice in defining the player's goals and intentions. Moreover, a set of quests may be mutually exclusive with another set, therefore forcing the player to choose which set of quests he will solve, having in mind the possible long-term effects these quests will have on the game world. Some quests can be solved in more than just one way, thus bringing non-linearity to the game. Quests can also be linked together to form *quest chains*, i.e. groups of quests that the player can only complete in sequence [9].

What is noteworthy and important to realize in regards to the focus of this work is the fact that *quests are a fundamental structure by which the player moves the storyline forward in computer role-playing games*. In other words, a quest is a conceptual bridge between the open structure of role-playing games and the closed structure of stories [10], thus being an ideal vehicle for interactive storytelling in computer role-playing games. Consequently, *it is possible to use computer role-playing games as a medium for interactive storytelling by dynamically generating non-linear quests*.

Non-player Characters. Role-playing game worlds are populated by *non-player characters* (NPCs) that cannot be controlled by the player. Instead, their behavior is scripted by the game designers and executed by the game engine. Players interact with non-player characters through dialogue. Most role-playing games feature branching dialogue (or *dialogue trees*). As a result, when talking to a non-player character,

the player may choose between a list of dialogue options where each choice often results in a different reaction. Such choices may affect the player's course of the game, as well as latter conversations with non-player characters. In other words, key non-player characters "remember" witnessed actions and dialogue responses previously committed and said by the player, and shape their relationship with him based on their memories [2].

Items, Containers and Objects. Throughout playing role-playing games, quests require the player to find, collect and properly use various *items* scattered throughout the game world. Special items may be equipped on the player's avatar, improving his abilities, skills or other attributes. All items can be either created from other items, or obtained from *containers*, i.e. objects that can hold or carry items. The player himself is an example of a container, since he can carry items in his *inventory*. Other typical examples of containers include non-player characters and *objects* representing treasure chests.

From an interactive storytelling point of view, computer role-playing games can be formalized into a set of *atomic actions* having narrative impact that a player (or a non-player character) is able to commit inside the game world. Each atomic action has the following properties:

- **Name:** A string concisely describing the atomic action.
- **Source:** The type of an in-game element that can commit the action, e.g., the player, or a non-player character.
- **Target:** The type of an in-game element that this action is committed upon, e.g., a container.
- **Parameters:** A list of in-game elements that the action operates on. Each item in the list consists of a unique name identifying the parameter and the type of the in-game element specified by the parameter.
- **Preconditions:** A list of statements regarding the specified *source*, *target* and *parameters* defining the circumstances under which the action can be committable in the game world.
- **Effects:** A list of changes to the game world that are a consequence of the action having been committed.

For example, the second atomic action mentioned in Table 1 is defined as follows:

- **Name:** `TAKE`
- **Source:** Character
- **Target:** Container
- **Parameters:** Item i
- **Preconditions:** Target holds item i
- **Effects:**
  - ~~Target holds item i~~ (negation of the action's precondition)
  - Source holds item i

The typical set of atomic actions that describes a common computer role-playing game is shown in Table 1.

Table 1. A typical computer role-playing game described by atomic actions.

Source	Atomic Action[2]	Target	Description
Character	`GIVE (i : Item)`	Container	The character specified as the source gives the item `i` to a container set as the action's target.
Character	`TAKE (i : Item)`	Container	The source character takes the item `i` from the target container.
Character	`USE (i : Item)`	Element	The action's source character uses item `i` on the element specified as the target.
Character	`EQUIP ()`	Item	The character specified as the source equips the targeted item.
Character	`WALK_TO ()`	Object	The source character walks near the targeted object.
Character	`TALK_TO (d : Dialogue)`	Character	The action's source initiates the dialogue tree `d` with the targeted character.

1.2 Action Planning Layer

From a top-down perspective, the role of the middle logical layer is to transform character goals set by the *Character Behavior Layer* into atomic and simple actions that are to be executed and visualized by the lowest logical layer – the *Visualization Layer*. From a bottom-up perspective, the *Action Planning Layer* is responsible for processing atomic and simple actions committed by the player and reported back from the *Visualization Layer*.

Since the described process is done on-the-fly in parallel while the player is playing the created computer role-playing game, the hereby described layer creates new quests based on the previous actions committed by the player and seamlessly integrates them into the game, thus dynamically creating an interactive story, as perceived by the player.

The *Action Planning Layer* operates on *simple actions*, *complex actions* and *action bindings*. A *simple action* refines the usage of exactly one atomic or simple action by specializing its source, target, parameters or by adding additional preconditions or effects. Unlike atomic actions, simple actions can alter interpersonal relationships – a component of the *Character Behavior Layer*. Every simple action has the following structure:

- **Name:** A string concisely describing the simple action.
- **Base action:** The atomic or simple action that this extending action refines.
- **Source:** The base action's source, its subtype or a concrete entity.
- **Target:** The base action's target, its subtype or a concrete entity.

[2] The atomic actions are written in a shortened textual form with the following structure: `NAME` (*list of parameters written as* "`name : type`" *pairs*).

- **Parameters:** A list of in-game elements that the simple action operates on. Each item in the list is equal to, a subtype of, or a concrete entity of the corresponding item in the base action's list of parameters.
- **Preconditions:** A list of the base action's preconditions plus additional preconditions regarding this simple action.
- **Effects:** A list of the base action's effects plus additional effects regarding this simple action.

Below is an example of a simple action that enables the player to steal items from non-player characters by refining the atomic action `TAKE` (defined in section 2.1):

- **Name:** `STEAL`
- **Base action:** `TAKE`
- **Source:** Player (a subtype of Character)
- **Target:** Non-player character (a subtype of Container)
- **Parameters:** Stealable i (a category of Item, see below)
- **Preconditions[3]:** [Target holds item i]
- **Effects[3]:**
  - ~~[Target holds item i]~~
  - [Source holds item i]
  - i is stolen
  - Target dislikes player
  - Player's karma decreased

The last two effects alter the player's attributes and interpersonal relationships between the target NPC and the player – see section 2.3 for more details.

The third effect tags the parameter `i` as "stolen," since `i` is an Item of category Stealable, which has the "stolen" tag defined. An Item instance can belong to multiple item categories, each having defined custom tags that can be set to be present or absent on the Item instance – see Figure 4.

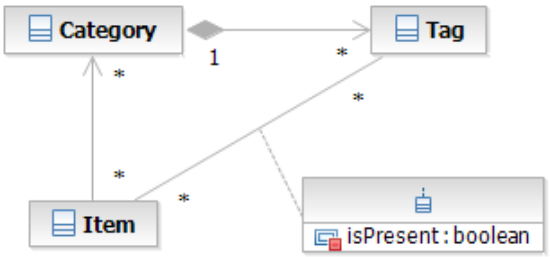


Fig. 4. A class diagram showing the principle of item categorization and tagging.

A *complex action* encloses multiple actions in a list with atomic, simple or complex actions as its elements. Every complex action has the following structure:

[3] The preconditions and effects inherited from the base action are enclosed in [square brackets].

- **Name:** A string concisely describing the complex action.
- **Source:** The type of an in-game element, or a concrete entity that is the source of all enclosed actions.
- **Targets:** The types of in-game elements, or concrete entities that are targets of the enclosed actions.
- **Parameters:** A list of in-game elements that the enclosed actions operate on.
- **Enclosed actions:** A list containing atomic, simple or complex actions that this complex action encloses. These are executed sequentially and can be interrupted.
- **Preconditions:** A filtered[4] list containing preconditions of the enclosed actions plus additional preconditions regarding this complex action.
- **Effects:** A filtered[4] list containing effects of the enclosed actions plus additional effects regarding this complex action.

Following is an example of a complex action, with which a non-player character unlocks a locked container for the player if the particular non-player character likes him:

- **Name:** `OPEN_LOCKED_CONTAINER_FOR_PLAYER`
- **Source:** Non-player character
- **Targets:**
  - Player
  - Lockable i
- **Parameters:** Lockable i (a category of Item)
- **Enclosed actions[5]:**
  - `Source : TAKE (i) → Player`
  - `Source : UNLOCK() → i`
  - `Source : GIVE (i) → Player`
- **Preconditions:**
  - [Target holds i]
  - [i is locked]
  - Source likes target
- **Effects:**
  - [Target holds i]
  - ~~[i is locked]~~

The purpose of *action bindings* is to bind actions to actual in-game entities, thus defining what can be done with all defined items and what exactly can the player and all existing non-player characters do.

An example given below denotes that the player can take or steal the item representing blueprints of the ENIAC computer from non-player characters representing its designers, John Mauchly and John Eckert.

Actual in-game entities:

- Item “ENIAC Blueprints”

[4] The lists do not contain preconditions or effects that are created and at the same time annulled during the sequential execution of the enclosed actions.

[5] Each action is written in the form: `Source : NAME (`*parameters*`) → Target`.

– NPC “John Mauchly”
– NPC “John Eckert”

Action bindings[5]:

```
– Player : TAKE  (“ENIAC Blueprints”) → “John Mauchly”
– Player : STEAL (“ENIAC Blueprints”) → “John Mauchly”
– Player : TAKE  (“ENIAC Blueprints”) → “John Eckert”
– Player : STEAL (“ENIAC Blueprints”) → “John Eckert”
```

The *Action Planning Layer* utilizes Hierarchical Task Network planning in a similar way as described in [4]. The algorithm searches for appropriate actions based on recursive matching of their effects with required preconditions. In other words, HTN planning finds all actions resulting in the required preconditions. The process of planning and replanning of actions is not described in this paper due to length constraints.

1.3 Character Behavior Layer

The *Character Behavioral Layer* is responsible for generating goals of all in-game characters, i.e. the player and all non-player characters. These goals are created according to interpersonal relationships among the player and non-player characters by matching their existing values[6] to a set of *behavior patterns* (see below) and picking the resulting goals from the best matching patterns. All generated characters’ goals are afterwards translated to plans by the *Action Planning Layer*.

A *behavioral pattern* defines the circumstances that lead to a change in the behavior of characters from a narrative point of view. The set of all behavioral patterns defines the conditional reasoning of all in-game characters. Each pattern has the following structure:

– **Description:** An optional text concisely describing the behavioral pattern.
– **Preconditions:** A set of in-game elements, including their attributes and relationships among them. The pattern’s preconditions also include the preconditions of goals represented by actions. Undesirable ones may be excluded from the list.
– **Goals:** A list containing goals describing the change in characters’ behavior as a consequence to the situation portrayed by the preconditions. The list of goals can contain actions of any type and desired changes in relationships or attributes of in-game elements.
– **Effects:** A list containing effects of all identified goals. Undesirable effects of goals represented by actions may be explicitly excluded from the list.

Below is an example behavioral pattern that describes the situation in which John, a husband of Mary with a respiratory infection, cures his wife with thyme syrup that the player had given him:

[6] The user sets the initial values of interpersonal relationships, if necessary. Otherwise all relationships are initialized to their default values.

- **Preconditions:**
  - Non-player characters "John," "Mary"
  - "John" is husband of "Mary", "Mary" is wife of "John"
  - "Mary" is sick_with_respiratory_infection
  - […preconditions of the two actions that form goals…]
- **Goals:**
  - `Player : GIVE ("Thyme syrup") → John`
  - `John : USE ("Thyme syrup") → Mary`
- **Effects:**
  - […effects of the two actions that form goals…]
  - ~~"Mary" is sick_with_respiratory_infection~~

This pattern contains examples of two actual interpersonal relationships, namely "is husband of," "is wife of" and an attribute "is sick_with_respiratory_infection." Following is a detailed description of all types of interpersonal relationships and attributes that the *Character Behavior Layer* operates on.

NPC → Character Relationships

Interpersonal relationships oriented from a non-player character[7] to any type of in-game character (i.e. the player or another non-player character) are identified with their unique label.

Under normal circumstances, NPC → Character relationships are either *absent* (the default initial value) or *present*[8]. However, there are cases when two relationships are mutually exclusive (meaning that the presence of one disallows the presence of the other and vice-versa). Such pairs of relationships are merged into relationships with two complementary values. These interpersonal relationships can have only one of their two values present at a time, e.g., if the precondition "`Tom likes Player`" is true at a given time, then the precondition "`Tom dislikes Player`" is false at the same time.

Character Roles

Another type of relationships between in-game characters are *character roles*. Unlike NPC → Character relationships, character roles can be oriented from the player, as well. Moreover, character roles can be bilateral.

Similarly to NPC → Character relationships, character roles are either *absent* (the initial default value) or *present*. If a bilateral character role is present, then preconditions testing either end evaluate to true, e.g., preconditions "`John is husband of Mary`" and "`Mary is wife of John`" are either both true, or both false in a given time.

---

[7] Relationships oriented from the player are not considered, because monitoring such relationships and limiting the number of actions the player can commit based on their presence would degrade the player's experience.

[8] Certain types of NPC → Character relationships can be set to have a numerical value instead of being either present or absent. Such mathematical relationships require additional apparatus for expressing preconditions that is not described in this paper.

Character Attributes
Every in-game character, whether being the player or a non-player character can have various attributes defined. Such attributes are either *unnumbered* or *numbered*.

Unnumbered character attributes do not have any numerical value and are either *absent* (the default initial value) or *present*. An example of an unnumbered character attribute is "sick_with_respiratory_infection" referenced in the aforementioned behavioral pattern example.

Numbered character attributes are present in every in-game character (i.e. the player and all non-player characters) and store a numerical value (individually for every character and from a custom-defined interval), unlike unnumbered attributes. An example of numbered attributes found in the majority of computer role-playing games is located in the left column of Table 2.

In addition, these attributes can be easily made domain-specific, e.g., a role-playing game created in the domain of teaching programming languages can define the numbered character attributes located in the right column of Table 2.

Table 2. Examples of various numbered character attributes.

Common Attributes	Domain-specific Attributes
– Agility	– C++ proficiency
– Endurance	– Java proficiency
– Intelligence	– Lisp proficiency
– Strength	
– Wisdom	

3 Conclusions and Future Work

We have described an innovative concept in the field of interactive storytelling that uses techniques common in this field in such a unique way that, in contrast to all existing solutions, enables to programmatically generate interactive stories in computer role-playing games. This is achieved by formally describing computer role-playing games in terms of atomic actions into which all narrative actions are eventually translated from character goals.

Our future work consists of implementing a prototype of an interactive storytelling system built upon the principles proposed in this paper and evaluating the results.

The prototype will first read from an input file a set of rules and data structures defining a storyworld (i.e. simple and complex actions, character relationships and attributes as defined in section 2) including the player's main goal, according to which an interactive story will be generated and presented to the player. The story will progress on-the-fly by reacting to narrative actions that the player had committed according to rules stored in the input file and based on the player's personality [1].

The resulting prototype will be validated by generating interactive stories in the domain of teaching the basic principles of software design and engineering. The results are to be evaluated by analyzing the correlation between the number of inputted rules and the number of possible interactive story variations that may be generated

according to these rules. Moreover, the prototype will be assessed by test players who will fill out questionnaires regarding various aspects of the generated stories.

Acknowledgement. This work was partially supported by the Cultural and Educational Grant Agency of the Slovak Republic, grant No. KEGA 3/5187/07.

References

1. ANDREJKO, A., BARLA, M. BIELIKOVÁ, M.: Ontology-based User Modeling for Web-based Information Systems. In: *Knapp. G. et al. (Eds.): Advances in Information Systems Development: New Methods and Practice for the Networked Society*, Vol. 2, Springer, Budapest, (2007), 457-468.
2. BARTON, M.: *Dungeons and Desktops: The History of Computer Role-playing Games*. A K Peters Ltd, Wellesley (2008).
3. BIELIKOVÁ, M., DIVÉKY, M., JURNEČKA, P., KAJAN, R., OMELINA, Ľ.: Automatic generation of adaptive, educational and multimedia computer games. *Signal, Image and Video Processing*, Vol. 2, No. 4 (2008) 371-384.
4. CAVAZZA, M., CHARLES, F., MEAD, S. J.: Planning Characters' Behaviour in Interactive Storytelling. *Journal of Visualization and Computer Animation*, Vol. 13, No. 2 (2002) 121-131.
5. CHENG, K., CAIRNS, P. A.: Behaviour, Realism and Immersion in Games. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, ACM, New York, (2005), 1272-1275.
6. CRAWFORD, C.: *Chris Crawford on Interactive Storytelling*. New Riders Games, (2004).
7. DENNING, S.: *The Leader's Guide to Storytelling: Mastering the Art and Discipline of Business Narrative*. Jossey-Bass, (2005).
8. DENNING, S.: *The Springboard: How Storytelling Ignites Action in Knowledge-Era Organizations*. Butterworth-Heinemann, (2000).
9. HOWARD, J.: *Quests: Design, Theory, and History in Games and Narratives*. A K Peters Ltd, Wellesley (2008).
10. JUUL, J.: *Half-Real: Video Games between Real Rules and Fictional Worlds*. MIT Press, Cambridge (2005).
11. KOTTER, J. P.: *A Sense of Urgency*. Harvard Business School Press, Boston (2008).
12. MCKEE, R.: *Story: Substance, Structure, Style and the Principles of Screenwriting*. Methuen Publishing Ltd, London (1999).
13. MURRAY, J.: From Game-Story to Cyberdrama. In: *FirstPerson: New Media as Story, Performance, and Game*, MIT Press, Cambridge (2004), 2-11.
14. PIZZI, D., CHARLES, F., LUGRIN J-L. et al.: Interactive Storytelling with Literary Feelings. In *Proc. of the 2nd Int. Conf. on Affective Computing and Intelligent Interaction*, Springer, Berlin (2007).
15. ROLLINGS, A., ADAMS, E.: *Andrew Rollings and Ernest Adams on Game Design*. New Riders Games, (2003).
16. SZILAS, N.: The Future of Interactive Drama. In *Proc. of the 2nd Australasian Conf. on Interactive Entertainment*, Creativity & Cognition Studios Press, Sydney (2005), 193-199.
17. TYCHSEN, A.: Role Playing Games – Comparative Analysis Across Two Media Platforms. In *Proc. of the 3rd Australasian Conf. on Interactive Entertainment*, Murdoch University, Perth (2006), 75-82.

D. IIT.SRC 2009 PAPER

This appendix includes a paper that has been submitted to, accepted and presented at the IIT.SRC 2009 student research conference, which took place on April 29, 2009 at the Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Slovakia.

Generating Interactive Stories in Computer Role-playing Games

Marko DIVÉKY*

Slovak University of Technology
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
markod@nextra.sk

Abstract. Interactive storytelling is a new area of research in the field of artificial intelligence. Its aim is to tell stories with the use of computers in a new and interactive way, which immerses the reader inside the story as the protagonist and enables him to drive its course in any desired direction. Interactive storytelling thus transforms conventional stories from static structures to dynamic and adaptive storyworlds. In this paper, we describe an innovative approach to interactive storytelling that utilizes computer role-playing games, today's most popular genre of computer games, as the storytelling medium, and focus on explaining the story generation cycle in detail.

1 Introduction

Stories and the art of storytelling have played an inevitable role in our lives ever since the earliest days of language. Historians found first records of storytelling in ancient cultures and their languages, in which stories served as the vehicle by which cultural knowledge was communicated from one generation to the next [2].

Even in today's modern times, such utilitarian use of stories has endured and the educational potential of storytelling is widely utilized in many different areas, e.g., in business [4] to spread knowledge amongst employees in order to help them become much more productive and collaborate with one another. Stories are also used in many other ways: in policy, in process, in pedagogy, in critique and as a foundation and as a catalyst for change [10].

The reason why storytelling proves to be an effective medium for educating is that the human mind is programmed much more for stories than for abstract facts [11].

---

* Master degree study programme in field: Software Engineering
Supervisor: Professor Mária Bieliková, Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies STU in Bratislava

IIT.SRC 2009, Bratislava, April 29, 2009, pp. 1–8.

1.1 Conventional Stories

Many aspects of stories have changed since the very first records of storytelling in ancient times. However, even in today's modern world, the fundamental idea behind conventional stories remains unaltered. Let us suppose that the storyteller seeks to communicate some truth or information (principle) to a desired audience. Instead of just telling the principle, the storyteller translates it into an instantiation (a story), then communicates the story to the audience, which in turn translates the instantiation back into the principle [2], as depicted in Figure 1.

Figure 1. The process of telling a conventional story.

1.2 Interactive Storytelling

Interactive storytelling is best described as "a narrative genre on computer where the user is one main character in the story and the other characters and events are automated through a program written by an author. Being a character implies choosing all narrative actions for this character" [12].

Interactive storytelling differs from conventional stories in a way that the process of transforming the principle into the instance (story) is delegated to the computer. Formally speaking, the computer transforms the principle into a *storyworld*, which operates on rules rather than on predefines events – see Figure 2. A single playing of a storyworld generates a single story. In other words, when a player goes thought a storyworld, he produces a linear sequence of events that makes a story. Different playing of the storyworld can yield many different stories, but all of them share one common principle [2], as depicted in Figure 2.

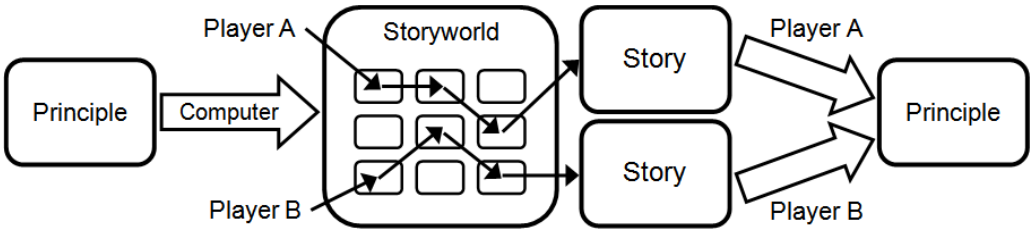


Figure 2. The process of playing through an interactive storyworld.

1.3 Computer Role-playing Games

Researchers in the field of interactive storytelling often disregard computer games as a suitable storytelling medium [2]. Still, some see them as the ideal form for storytelling given their practically unlimited degree of interactivity and visual attractiveness. Janet Murray writes that computer games are "a new medium of expression [that] allows us

to tell stories we could not tell before, to retell the age-old stories in new ways" [8]. To support her opinion, Murray states that computer games are "a medium that includes still images, moving images, text, audio, three-dimensional, navigable space – more of the building blocks of storytelling than any single medium has ever offered us" [8].

The storytelling potential of computer games differs from one genre to the next. Throughout the many diverse genres of computer games available today, role-playing games (RPGs) have the most detailed and involving storyline, thus being the most appropriate genre for storytelling [6, 9].

2 Related Work

Despite the large amount of work that has already been done in the field of interactive storytelling, there have been only a few working solutions that found practical use other than being proof-of-concept demonstrations. However, it is possible to divide the most notable solutions into the following three categories:

1. *Systems that generate complex, but only text-based interactive stories*, e.g., Storytron (formerly Erasmatron) developed by Chris Crawford since 1991 [3]. Storytron utilizes a drama manager agent to direct the generated stories, which are, however, very cumbersome to define and have only text-based visualization.
2. *Systems that generate visually attractive, but not interactive stories*, e.g., I-Storytelling developed by Marc Cavazza and Fred Charles at the University of Teesside [1]. Their solution utilizes Hierarchical Task Network (HTN) planning and Heuristic Search Planning (HSP) to generate stories via autonomous behavior of character agents. Users, however, have very limited or no means of interacting with and directly influencing the generated stories.
3. *Systems that generate interactive stories, but only with a single dramatic conflict*, e.g., Façade developed by Michael Mateas and Andrew Stern [7]. Façade uses Natural Language Processing (NLP) to parse user-inputted actions and visualizes stories with a custom proprietary 3D graphics engine. Façade solves almost all problems that relate to interactive storytelling; however, it is able to generate only a single dramatic scene.

Consequently, our goal is to devise a new approach to interactive storytelling that not only builds on present-day techniques and formalisms in a way that eliminates the above-mentioned drawbacks of existing interactive storytelling solutions, but also enables to generate interactive stories in computer role-playing games.

3 Overview of the Proposed Approach

The aim of the hereby-described approach is to programmatically generate dynamic and interactive stories with computer role-playing games as their medium, therefore combining the dynamic and enthralling storyworlds created by interactive storytelling with the visual appearance, gameplay and popularity of computer role-playing games. The presented concept can be broken into three logical layers, as depicted in Figure 3.

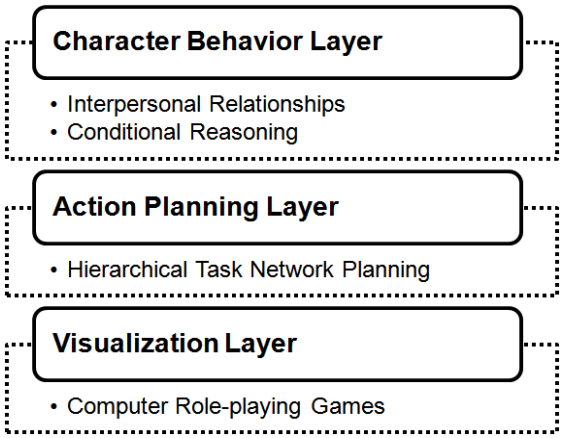


Figure 3. Logical structure of the proposed concept.

All generated interactive stories initiate in the topmost logical layer, named the *Character Behavior Layer*, which is responsible for generating goals of all in-game characters, i.e. the player and all non-player characters (NPCs). These goals are created according to interpersonal relationships among the player and non-player characters by conditionally matching their existing values to a set of *behavior patterns* and picking the resulting goals from the best matching patterns.

From a top-down perspective, the role of the *Action Planning Layer* is to transform character goals set by the *Character Behavior Layer* into plans consisting of atomic actions that are to be visualized by the *Visualization Layer*. From a bottom-up perspective, the *Action Planning Layer* is responsible for processing and evaluating actions previously committed by the player and all non-player characters, as reported back by the *Visualization Layer*.

The lowest logical layer called the *Visualization Layer* uses the concept of computer role-playing games as the visualization and storytelling medium, formalized into a set of *atomic actions* that a player (or a non-player character) is able to commit inside the game world. The *Visualization Layer* therefore provides graphical visualization of the generated interactive stories by executing atomic actions planned for non-player characters and letting the player interact with the generated stories by letting him commit narrative actions, which are reported back to the *Action Planning Layer* immediately upon being committed.

4 The Story Generation Cycle

The three layers described above are cyclically used to generate interactive stories, as depicted in Figure 4. The whole process is described below in more detail.

Before any interactive stories can be generated, the author of the stories, i.e. the storyteller creates his domain-specific rules and input data based on which the proposed storytelling system operates. In other words, the storyteller defines the necessary *simple* and *complex actions*, *action bindings*, types of possible character properties

(*attributes*, *relationships* and *roles* between characters) and *behavioral patterns*. All of these data structures are described in more detail in [5].

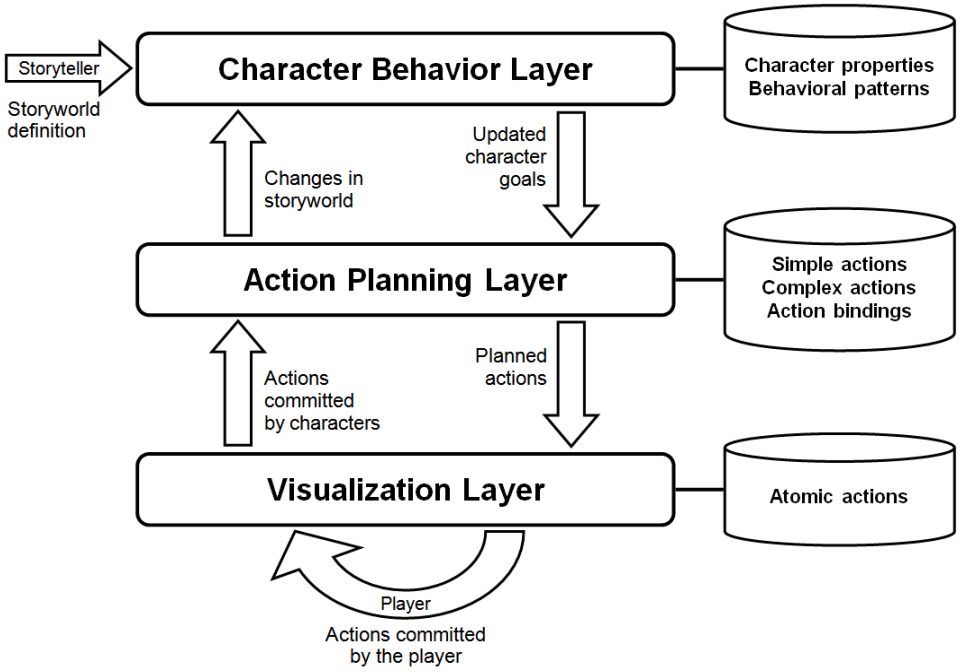


Figure 4. The story generation cycle.

After having defined all necessary custom domain-specific rules comes the initialization phase, in which the storyteller defines the storyworld that he wants the generated stories to take place in. He therefore creates the desired non-player characters along with their initial attributes, roles and relationships between them. The storyteller can also scatter objects and items throughout the storyworlds, as desired.

After the storyteller had defined the initial state of the storyworld, the *Character Behavior Layer* analyzes the storyworld and chooses one or more goals that best match its current state, i.e. goals having all of their preconditions met.

Such goals are afterwards processed by the *Action Planning Layer*, which translates all player's and non-player characters' active goals to plans. The planning process is based on the Hierarchical Task Network (HTN) planning formalism that recursively finds appropriate actions, effects of which meet the required conditions.

The planning process starts out with the effects of each active goal and finds any complex, simple or atomic actions (in this order) that have their effects match the goal's effects. The whole planning process afterwards runs recursively to search for actions that have their effects match the preconditions[1] of the newly found actions.

[1] Since the planner matches the required preconditions with effects of candidate actions, all actions depicted in Figure 5 have preconditions placed below them and effects above.

As shown in Figure 5, the resulting plan for every active goal is a tree-like graph that contains multiple paths of complex, simple or atomic actions, which result in accomplishing the particular goal when followed in the storyworld.

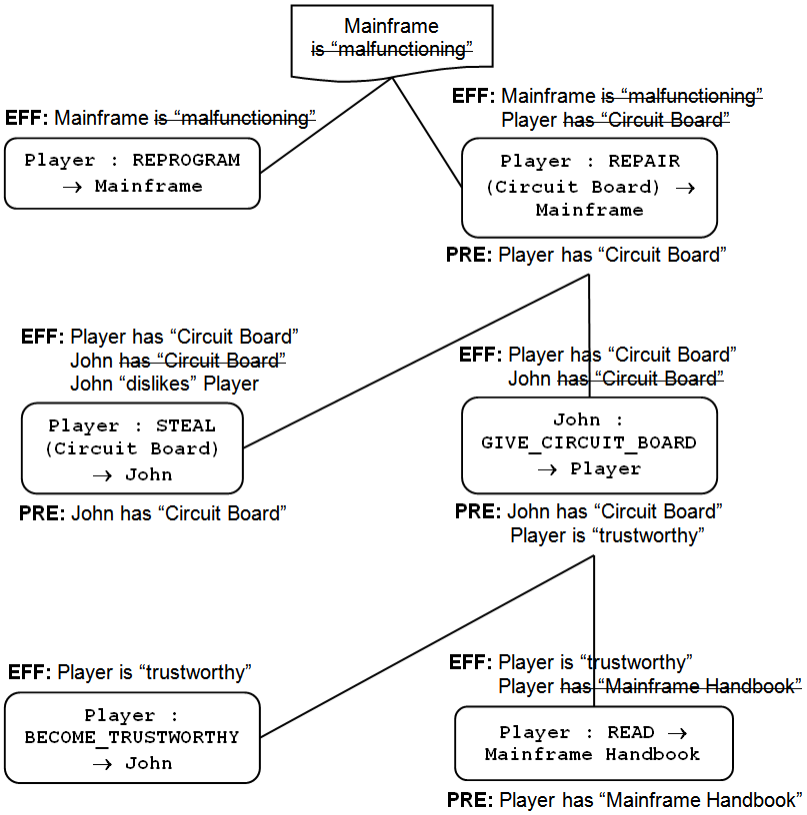


Figure 5. An example plan[2] consisting of six simple actions that are interconnected by common preconditions (PRE) and effects (EFF), with the goal being the plan's root. The plan is constructed from top to bottom, but carried out from bottom to top.

After plans for the player and all non-player characters are constructed, a subset of all available player's planned paths is chosen and delegated to the *Visualization Layer*, along with planned actions that each non-player character should commit, if any.

The subset of the player's plan is chosen according to the player's user model, which contains records of all actions that he had previously successfully committed in other storyworlds. Thanks to these records, it is possible to filter out and ignore

[2] The plan is from a storyworld in the domain of teaching history of computing and programming basics, in which the player is to repair a malfunctioning mainframe computer, either by reprogramming it or repairing it with a spare circuit board, which he can steal or get from John, a non-player character, upon becoming trustworthy by successfully answering a question or by reading a mainframe handbook.

planned paths containing similar actions that the player previously successfully committed (*negative personalization*), or actions he had not yet committed or failed to commit successfully (*positive personalization*). If the player's user model contains no records (i.e. the player is playing through his first storyworld) then the subset of player's planned paths is selected randomly.

In the example plan shown in Figure 5, presuming positive personalization and that the player's user model states that the player had previously committed the simple action `REPROGRAM`, then the path with the simple action `REPROGRAM` is chosen and delegated to the *Visualization Layer*, as shown in Figure 6.

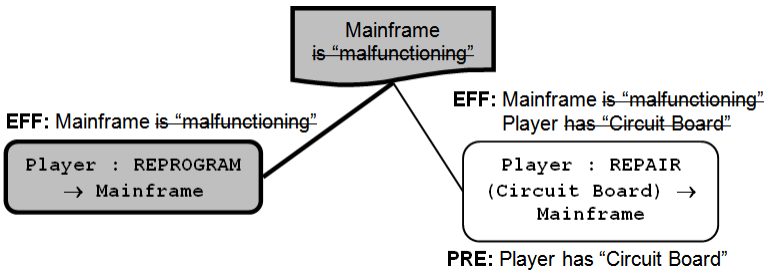


Figure 6. An example of a chosen path, colored grey (the rest of the player's plan is omitted).

The *Visualization Layer* receives planned actions for both the player and non-player characters (NPCs) from the *Action Planning Layer* and the actions destined for NPCs are committed by the particular NPCs, whereas the actions planned for the player are presented to him as multiple options via the game's graphical interface.

The player commits, either successfully or unsuccessfully, his desired action (that may not have been planned, but must have its preconditions met), what is afterwards signaled back to the *Action Planning Layer*, along with actions that were, either successfully or unsuccessfully, committed by any non-player characters.

All committed actions, whether successfully and unsuccessfully, are processed by the *Action Planning Layer* that matches all committed actions to player's and non-players' existing plans, which remain either unaffected, or are replanned according to the new state of the storyworld, i.e. new planned paths are again chosen based on the player's user model, as described in the preceding paragraphs.

Any alterations to the state of the storyworld, such as changes in characters' attributes, roles or relationships are signaled to the *Character Behavior Layer*, which collects all changes to the storyworld from the *Action Planning Layer* and determines whether any of the existing goals are affected (in terms of having been accomplished or not being accomplishable anymore), or whether preconditions for any new goals are met. The current set of both the player's and non-player characters' goals is updated and any changes are delegated back to the *Action Planning Layer*.

The hereby-described process is repeated in a cyclic manner until the player has successfully accomplished all of his goals, in which case the story ends. An ending also occurs if the player is unable to accomplish all of his existing goals and there are no more possible paths left to do so.

5 Conclusions and Future Work

We have described the story generation cycle of an innovative approach in the field of interactive storytelling, which uses common formalisms in such a unique way that, in contrast to all existing solutions, operates on simple-to-define rules, and enables to programmatically generate interactive stories in computer role-playing games.

Our future work consists of evaluating a prototype system built upon the hereby-described concept both formally and empirically on an example storyworld in the domain of teaching the history of computing and programming basics. The prototype is to be assessed by players who will fill out questionnaires regarding the generated stories.

Acknowledgement: This work was partially supported by the Cultural and Educational Grant Agency of the Slovak Republic, grant No. KEGA 3/5187/07.

References

[1] Cavazza, M., Charles, F., Mead, S.J.: Sex, Lies, and Video Games: An Interactive Storytelling Prototype. In: *Proceedings of the 2002 AAAI Spring Symposium*, AAAI Press, Menlo Park (2002), pp. 13–17.

[2] Crawford, C.: *Chris Crawford on Interactive Storytelling*. New Riders Games, (2004).

[3] Crawford, C. et al.: *Storytron*. [Online; accessed February 25, 2009]. Available at: http://www.storytron.com/

[4] Denning, S.: *The Springboard: How Storytelling Ignites Action in Knowledge-Era Organizations*. Butterworth-Heinemann, (2000).

[5] Divéky, M., Bieliková M.: An Approach to Interactive Storytelling and its Application to Computer Role-playing Games. In Návrat, P., Chudá, D., eds.: *Znalosti 2009: Proceedings of the 8th annual conference*, STU, Bratislava (2009), pp. 59–70.

[6] Mallon, B., Webb, B.: Stand Up and Take Your Place: Identifying Narrative Elements in Narrative Adventure and Role-play Games. *Computers in Entertainment*, Vol. 3, No. 1 (2005) pp. 6–6.

[7] Mateas, M., Stern, A.: *Façade*. [Online; accessed February 25, 2009]. Available at: http://www.interactivestory.net/

[8] Murray, H.J.: From Game-Story to Cyberdrama. *FirstPerson: New Media as Story, Performance and Game*, MIT Press, Cambridge (2004), pp. 2–11.

[9] Rollings, A., Adams, E.: *Andrew Rollings and Ernest Adams on Game Design*. New Riders Games, (2003).

[10] Sandercock, L.: Out of the Closet: The Importance of Stories and Storytelling in Planning Practice. *Planning Theory & Practice*, Vol. 4, No. 1 (2003), pp. 11–28.

[11] Schank, R.: *Tell Me a Story: Narrative and Intelligence*. Northwestern University Press, Evanston (1995).

[12] Szilas, N.: The Future of Interactive Drama. In: *Proceedings of the Second Australasian Conference on Interactive Entertainment*, Creativity & Cognition Studios Press, Sydney (2005), pp. 193–199.

E. EC-TEL 2009 PAPER

This appendix contains a paper that has been submitted and accepted[1] as a full paper to the EC-TEL 2009 conference, which will take place from September 29 until October 2, 2009 in Nice, France.

[1] Only 35 (out of a total of 158) submissions were accepted as full papers.

Generating Educational Interactive Stories in Computer Role-playing Games

Marko Divéky, Mária Bieliková

Institution of Informatics and Software Engineering,
Faculty of Informatics and Information Technologies,
Slovak University of Technology, Ilkovičova 3, 842 16 Bratislava, Slovakia
markod@nextra.sk, bielik@fiit.stuba.sk

Abstract. The aim of interactive storytelling is to tell stories with the use of computers in a new and interactive way, which immerses the reader inside the story as the protagonist and enables him to drive its course in any desired direction. Interactive storytelling thus transforms conventional stories from static structures to dynamic and adaptive storyworlds. In this paper, we describe an innovative approach to interactive storytelling that utilizes computer role-playing games, today's most popular genre of computer games, as the storytelling medium in order to procedurally generate educational interactive stories.

Keywords: Interactive storytelling, story generation, educational stories, computer games, role-playing games

1 Introduction

Stories and the art of storytelling have played an inevitable role in our lives ever since the earliest days of language. Historians found first records of storytelling in ancient cultures and their languages, in which stories served as the vehicle by which cultural knowledge was communicated from one generation to the next [1].

Even in today's modern times, such utilitarian use of stories has endured and their educational potential is utilized in many different areas, e.g., in business to spread knowledge amongst employees in order to help them become more productive and to collaborate with one another [2]. Stories are also used in many other ways: in policy, in process, in pedagogy, in critique and as a foundation and as a catalyst for change [3]. The reason why storytelling proves to be an effective medium for educating is that our minds are programmed much more for stories than for abstract facts [4].

Many aspects of stories have changed since the very first records of storytelling in ancient times. However, even in today's modern world, the fundamental idea behind *conventional stories* remains unaltered. Let us suppose that the storyteller seeks to communicate some truth or information (principle) to a desired audience. Instead of just telling the principle, the storyteller translates it into an instantiation (a story), then communicates the story to the audience, which in turn translates the instantiation back into the principle [1], as depicted in Fig. 1.

Fig. 1. The process of telling a conventional story.

Interactive storytelling differs from conventional stories in a way that the process of transforming the principle into the instance (story) is delegated to the computer. It is best described as "a narrative genre on computer where the user is one main character in the story and the other characters and events are automated through a program written by an author. Being a character implies choosing all narrative actions for this character" [5].

In interactive storytelling, the computer transforms the principle into a *storyworld*, which operates on rules rather than on predefines events (see Fig. 2). A single playing of a storyworld generates a single story. In other words, when a player goes through a storyworld, he produces a linear sequence of events that makes a story. Different playing of the storyworld can yield many different stories, but all of them share one common principle [1], as depicted in Fig. 2.

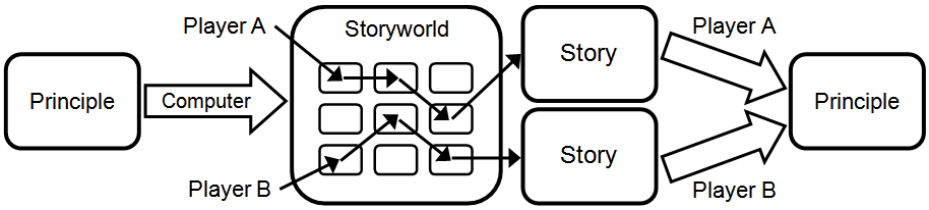


Fig. 2. The process of playing through an interactive storyworld.

In this paper, we propose an approach that enables to generate educational interactive stories in *computer role-playing games*. Researchers in the field of interactive storytelling often disregard computer games as a suitable storytelling medium [1]. Still, some researchers see them as the ideal form for storytelling given their practically unlimited degree of interactivity and visual attractiveness [6].

Throughout the many diverse genres of computer games available today, role-playing games (RPGs) have the most detailed and involving storyline, thus being the most appropriate genre for storytelling [7]. Moreover, computer role-playing games have originally derived from tabletop role-playing games and are considered as being one of today's most popular genres of computer games [7].

2 Related Work

Despite the large amount of work that has already been done in the field of interactive storytelling, there have been only a few working solutions that found practical use

other than being proof-of-concept demonstrations. However, it is possible to divide the most notable solutions into three categories described below:

- *Systems that generate complex, but only text-based interactive stories*, e.g., Storytron [8] (formerly Erasmatron) developed by Chris Crawford since 1991. Storytron utilizes a drama manager agent to direct the generated stories, which are, however, very cumbersome to define and have only text-based visualization.
- *Systems that generate visually attractive, but not interactive stories*, e.g., I-Storytelling [9] developed by Marc Cavazza and Fred Charles at the University of Teesside. Their solution utilizes Hierarchical Task Network (HTN) planning and Heuristic Search Planning (HSP) to generate stories via autonomous behavior of character agents. Users, however, have very limited or no means of interacting with and directly influencing the generated stories.
- *Systems that generate interactive stories, but only with a single dramatic conflict*, e.g., Façade [10] developed by Michael Mateas and Andrew Stern. Façade uses Natural Language Processing (NLP) to parse user-inputted actions and visualizes stories with a custom proprietary 3D graphics engine. Façade solves almost all problems that relate to interactive storytelling; however, it is able to generate only a single dramatic scene.

Consequently, our goal is to devise a new approach to interactive storytelling that not only builds on present-day techniques and formalisms in a way that eliminates the above-mentioned drawbacks of existing interactive storytelling solutions, but also enables to generate educational interactive stories in computer role-playing games.

3 Overview of the Proposed Concept

The aim of the hereby-described concept is to programmatically generate educational interactive stories with computer role-playing games as their medium, therefore combining the dynamic and enthralling storyworlds created by interactive storytelling with the visual appearance, gameplay and popularity of computer role-playing games. The presented concept can be broken into the following three logical layers:

- *Character Behavior Layer* describes interpersonal relationships and conditional reasoning of characters,
- *Action Planning Layer* realizes planning and replanning of narrative actions,
- *Visualization Layer* utilizes computer role-playing games for story visualization.

All generated interactive stories initiate in the topmost logical layer named the *Character Behavior Layer*, since all stories are about people, despite the fact that references to them are often indirect or symbolic [1]. The behavior of characters results in creating plans and planning actions that move the story forward. The middle layer, called the *Action Planning Layer*, handles all the planning and replanning. All created plans eventually break down into actions that are visualized by the lowest logical layer called the *Visualization Layer*.

All three logical layers operate on easy-to-define rules and data structures, which are described in detail in the following sections.

3.1 Visualization Layer

The *Visualization Layer* provides graphical visualization of the generated interactive stories to the players. It uses the concept of computer role-playing games as the visualization and storytelling medium. The most important aspects of computer role-playing games that are important for the scope of this paper are described below.

Themes. Games based on the role-playing genre are most often set in a fictional fantasy world closely related to classic mythology, or in a science-fiction world set somewhere in the future. Historical and modern themes are also common [11].

Avatars. In role-playing games, a player controls one in-game character called the *avatar*, and uses him as an instrument for interacting with the game world [12].

Character Development. Besides having the option to fully customize the appearance of his in-game character, the player is allowed to choose various attributes, skills, traits and special abilities that his avatar will possess. These are given to players as rewards for overcoming challenges and achieving goals, most commonly for completing *quests*. Character development plays, together with stories, a key role in today's computer role-playing games.

Quests. A *quest* in role-playing games can be defined as a journey across the game world in which the player collects *items* and talks to *non-player characters* "in order to overcome challenges and achieve a meaningful goal" [13]. Quests often require the player to find specific items that he needs to correctly use or combine in order to solve a particular task, and/or require the player to choose a correct answer from a number of given answers to a certain question [14]. Upon solving a quest, the player is often presented with a reward, which can have many forms – i.e. ranging from a valuable item to a new skill, trait or ability for the player's avatar.

Many quests are optional, allowing for freedom of choice in defining the player's goals. Moreover, a set of quests may be mutually exclusive with another set, therefore forcing the player to choose which set of quests he will solve, having in mind the possible long-term effects these quests will have on the game world. Some quests can be solved in more than just one way and thus bring non-linearity into the game.

What is noteworthy and important to realize with regard to the focus of this work is the fact that quests are a fundamental structure by which the player moves the storyline forward in computer role-playing games. In other words, a quest is a conceptual bridge between the open structure of role-playing games and the closed structure of stories, thus being an ideal vehicle for interactive storytelling in computer role-playing games. Consequently, it is possible to use computer role-playing games as a medium for interactive storytelling by dynamically generating non-linear quests.

Non-player Characters. Role-playing game worlds are populated by *non-player characters* (NPCs) that cannot be controlled by the player. Instead, their behavior is scripted by the game designers and executed by the game engine. Players interact with non-player characters through dialogue. Most role-playing games feature branching dialogue (or *dialogue trees*). As a result, when talking to a non-player character, the player may choose from a list of dialogue options where each choice often results in a different reaction. Such choices may affect the player's course of the game, as well as latter conversations with non-player characters.

Items, Containers and Objects. Throughout playing role-playing games, quests require the player to find, collect and properly use various *items* scattered throughout the game world. Special items may be equipped on the player's avatar, improving his abilities, skills or other attributes. All items can be either created from other items, or obtained from *containers*, i.e. *objects* that can hold or carry items. The player himself is an example of a container, since he can carry items in his inventory. Non-player characters and objects representing treasure chests are also examples of containers.

Atomic Actions. We have formalized computer role-playing games from an interactive storytelling point of view into a set of *atomic actions* having narrative impact that a player (or a non-player character) is able to commit inside the game world. Each atomic action has the following structure:

- *Name:* A string concisely describing the atomic action.
- *Source:* The type of an in-game element that can commit the atomic action, e.g., the player, or a non-player character.
- *Target:* The type of an in-game element that this atomic action is committed upon, e.g., an object or a container.
- *Parameter:* The type of an in-game element that the atomic action operates on. The presence of the parameter is optional.
- *Preconditions:* A set of statements regarding the specified source, target and parameter defining the circumstances under which the atomic action can be committed in the game world.
- *Effects:* A set of changes to the game world that are a consequence of the atomic action having been committed.

The typical set of atomic actions that describes a common computer role-playing game is shown in Table 1.

Table 1. Atomic actions that describe a typical computer role-playing game.

Source	Atomic Action[1]	Target	Description
Character	`GIVE (Item)`	Container	The character specified as the source gives the item to a targeted container.
Character	`TAKE (Item)`	Container	The source character takes the item from the target container.
Character	`USE (Item)`	Element	The atomic action's source character uses the item on the targeted element.
Character	`EQUIP ()`	Item	The character specified as the source equips the targeted item.
Character	`WALK_TO ()`	Element	The source character walks near the targeted element.
Character	`TALK_TO ()`	Character	The atomic action's source initiates a dialogue with the targeted character.

[1] The atomic actions are written in a shortened textual form with the following structure: `NAME (parameter).`

As an example, we elaborate the second atomic action mentioned in Table 1, which is defined as follows:

Name:	`TAKE`
Source:	Character
Target:	Container
Parameter:	Item
Preconditions:	Target has parameter
Effects:	Target ~~has parameter~~ (negation of the action's precondition) Source has parameter

3.2 Action Planning Layer

From a top-down perspective, the role of the middle logical layer is to transform character goals set by the *Character Behavior Layer* into plans consisting of actions that are to be executed and visualized by the bottommost logical layer – the *Visualization Layer*. From a bottom-up perspective, the *Action Planning Layer* is responsible for processing actions committed by the player and all non-player characters that were reported back from the *Visualization Layer*.

Since the described process is done on-the-fly while the player is playing a computer role-playing game, the hereby described layer creates new quests based on the previous actions committed by the player and seamlessly integrates them into the game, thus dynamically creating an interactive story perceived by the player.

The *Action Planning Layer* utilizes Hierarchical Task Network planning in a similar way as described in [15]. Our algorithm searches for appropriate actions based on recursive matching of their effects with required preconditions. In other words, the planner finds all actions resulting in the required preconditions (see section 3.4).

The *Action Planning Layer* operates on *simple actions*, *complex actions* and *action bindings*, all of which are described below.

Simple Actions. A *simple action* refines the usage of exactly one atomic or simple action by specializing its source, target, parameter or by adding additional preconditions or effects. Unlike atomic actions, simple actions can alter *attributes* of characters and *interpersonal relationships* – features of the *Character Behavior Layer*. Every simple action consists of the following structure:

- *Name:* A string concisely describing the simple action.
- *Base action:* An atomic or simple action that this simple action refines.
- *Source:* The base action's source type or its subtype (see below).
- *Target:* The base action's target type or its subtype.
- *Parameter:* The type of an in-game element that this simple action operates on. The parameter is equal to or a subtype of the base action's parameter.
- *Preconditions:* A set containing preconditions inherited from the base action plus additional preconditions related to this simple action.

— *Effects:* A set containing effects inherited from the base action plus additional effects related to this simple action.

Below is an example of a simple action named `REPAIR` that refines the atomic action `USE` (mentioned in Table 1) and enables the player to repair a malfunctioning computer with a spare circuit board:

Name:	`REPAIR`
Base action:	`USE`
Source:	Player (a character subtype)
Target:	Computer (an object subtype, see below)
Parameter:	Circuit board (an item subtype)
Preconditions[2]*:*	[Source has parameter] Target is "malfunctioning"
Effects[2]*:*	[Source has parameter] Target is "malfunctioning"

The second effect marks the target computer as "not malfunctioning," since it is defined as being an object subtype, which has a "malfunctioning" property defined. An object instance can belong to multiple object subtypes, each having defined custom properties that can be set on the object instance.

Likewise, subtypes of all other core in-game elements (described in section 3.1), i.e. items, containers and characters are also supported, with each subtype having its own custom properties defined.

Complex Actions. A *complex action* encloses multiple actions in a sequence with atomic, simple or complex actions being its elements. Every complex action has the following structure:

— *Name:* A string concisely describing the complex action.
— *Source:* Type of an in-game element that is the source of all enclosed actions.
— *Targets:* Types of in-game elements that are targets of the enclosed actions.
— *Parameters:* A set of in-game elements that the enclosed actions operate on. Each parameter is identified by a unique name distinguishing it from the others.
— *Enclosed actions:* A set containing atomic, simple or other complex actions that this complex action encloses.
— *Preconditions:* A filtered[3] set containing preconditions of all enclosed actions plus additional preconditions related to this complex action.
— *Effects:* A filtered set containing effects of all enclosed actions plus additional effects related to this complex action.

[2] The preconditions and effects inherited from the base action are enclosed in [square brackets].
[3] The sets do not contain preconditions nor effects that are created and at the same time annulled during the sequential execution of actions enclosed by the complex action.

Below is an example of a complex action, with which a non-player character unlocks a locked item for the player if the particular non-player character likes the player:

Name:	`UNLOCK_LOCKED_ITEM_FOR_PLAYER`
Source:	Non-player character
Targets:	Player Lockable item `i` (an item subtype)
Parameters:	Lockable item `i`

Enclosed actions[4]*:*	`Source : TAKE (i) → Player`
	`Source : UNLOCK() → i`
	`Source : GIVE (i) → Player`

Preconditions:	[Player has `i`] [`i` is "locked"] Source "likes" player
Effects:	[Player has `i`] [`i` ~~is "locked"~~]

Action Bindings. The purpose of optional *action bindings* is to explicitly permit or disallow bindings of atomic, simple, and complex actions to actual in-game entities. Action bindings therefore define what can or cannot be done with all defined story elements, and what exactly can or cannot the player and all non-player characters do.

For example, the action bindings given below[4] denote that the player can steal the item "thyme syrup" from the non-player character representing an evil doctor and cannot steal it from a non-player character representing a good doctor:

- `[CAN] Player : STEAL ("Thyme syrup") → "Evil Doctor"`
- `[CAN'T]Player : STEAL ("Thyme syrup") → "Good Doctor"`

3.3 Character Behavior Layer

The *Character Behavioral Layer* is responsible for generating goals of all in-game characters, i.e. the player and all non-player characters. These goals are created according to *interpersonal relationships* among the player and non-player characters by matching their existing values[5] to a set of *behavior patterns* (see below) and picking the resulting goals from the best matching patterns. All generated characters' goals are afterwards translated to plans by the *Action Planning Layer*.

Behavioral Patterns. A *behavioral pattern* defines the circumstances that lead to a change in the behavior of characters from a narrative point of view. The set of all

[4] Each action is written in the form: `Source : NAME (`*parameter(s)*`) → Target`.

[5] The storyteller sets the initial values of interpersonal relationships, if necessary. Otherwise, all relationships are initialized to their default values.

behavioral patterns defines the conditional reasoning of all in-game characters. Each behavioral pattern has the following structure:

- *Description:* An optional text concisely describing the behavioral pattern.
- *Preconditions:* A set of in-game elements, including their properties and relationships among them. Also included are the preconditions of actions that represent goals. Undesirable preconditions may be explicitly excluded.
- *Goals:* A set containing goals describing the change in characters' behavior as a consequence to the situation portrayed by the pattern's preconditions. Goals can be represented by actions of any type, or by changes in properties of in-game elements and in relationships among them. Each goal is bound to a character.
- *Effects:* A set containing effects of all identified goals. Undesirable effects may be explicitly excluded from the set.

Below is an example behavioral pattern that describes the situation in which John, a husband of Mary with a respiratory infection, cures his wife with thyme syrup that the player had given him:

Preconditions:	Non-player characters "John" and "Mary" John is "husband" to Mary, Mary is "wife" to John Mary is "sick with respiratory infection" […preconditions of the two actions that represent goals…]
Goals:	`Player : GIVE ("Thyme syrup") → John` `John : USE ("Thyme syrup") → Mary`
Effects:	[…effects of the two actions that represent goals…] Mary ~~is "sick with respiratory infection"~~

This behavioral pattern contains examples of two actual *interpersonal relationships*, namely "is husband to," "is wife to" and an *attribute* "sick with respiratory infection." Following is a detailed description of all types of interpersonal relationships and attributes that the *Character Behavior Layer* operates on.

NPC → Character Relationships. *Interpersonal relationships* oriented from a non-player character[6] to any type of in-game character (i.e. the player or another non-player character) are identified with their unique label.

Under normal circumstances, NPC → Character relationships are either absent (the default initial value) or present[7]. However, there are cases when two relationships are mutually exclusive (meaning that the presence of one disallows the presence of the other and vice-versa). Such pairs of relationships are merged into *relationships with*

[6] Relationships oriented from the player are not considered, because monitoring such relationships and limiting the number of narrative actions the player can commit based on their presence would degrade the player's experience of the generated stories.

[7] Certain types of NPC → Character relationships can be set to have a numerical value instead of being either present or absent. Such mathematical relationships require additional apparatus for expressing preconditions that is not described in this paper.

two complementary values. These interpersonal relationships can have only one of their two values present at a time, e.g., if the precondition "Tom likes player" is true at a given time, then the precondition "Tom dislikes player" is false at the same time.

Character Roles. Another type of relationships between in-game characters are *character roles*. Unlike NPC → Character relationships, character roles can be oriented from the player, as well. Moreover, character roles can be bilateral.

Similarly to NPC → Character relationships, character roles are either absent (the default initial value) or present. If a *bilateral character role* is present, then preconditions testing either end evaluate to true, e.g., "John is husband to Mary" and "Mary is wife to John" are either both true, or both false in a given time.

Character Attributes. Every in-game character, whether being the player or a non-player character can have various *attributes* defined. Such attributes are either unnumbered or numbered.

Unnumbered character attributes do not have any numerical value and are either absent (the default initial value) or present. An example of an unnumbered character attribute is "sick with respiratory infection" referenced in the aforementioned behavioral pattern example.

Numbered character attributes are present in every in-game character (i.e. the player and all non-player characters) and store a numerical value (individually for every character and from a custom-defined interval), unlike unnumbered attributes. An example of numbered attributes found in the majority of computer role-playing games is located in the left column of Table 2.

Table 2. Examples of various numbered character attributes.

Common Attributes	Domain-specific Attributes
– Agility	– C++ proficiency
– Endurance	– Java proficiency
– Intelligence	– Lisp proficiency
– Strength	

In addition, attributes of characters can be easily made domain-specific, e.g., an interactive storyworld created in the domain of teaching programming languages can have defined numbered character attributes located in the right column of Table 2.

3.4 The Story Generation Cycle

The three logical layers described in the preceding sections are cyclically used to generate interactive stories, as depicted in Fig. 3.

Preparation Phase. Before any interactive stories can be generated, the author of the stories, i.e. the storyteller creates his domain-specific rules and input data based on which the proposed storytelling system operates. In other words, the storyteller defines the necessary simple and complex actions, action bindings, types of possible character properties (attributes, relationships and roles between characters) and behavioral patterns – as described in the previous sections.

Initialization Phase. After having defined all necessary custom domain-specific rules comes the initialization phase, in which the storyteller defines the storyworld that he wants the generated stories to take place in. He therefore creates the desired non-player characters along with their initial attributes, roles and relationships between them. The storyteller can also optionally scatter objects, containers and items throughout the storyworld as desired.

Story Generation Phase. After the storyteller had defined the initial state of the storyworld, the *Character Behavior Layer* analyzes the storyworld and chooses one or more goals that best match its current state, i.e. goals having all preconditions met.

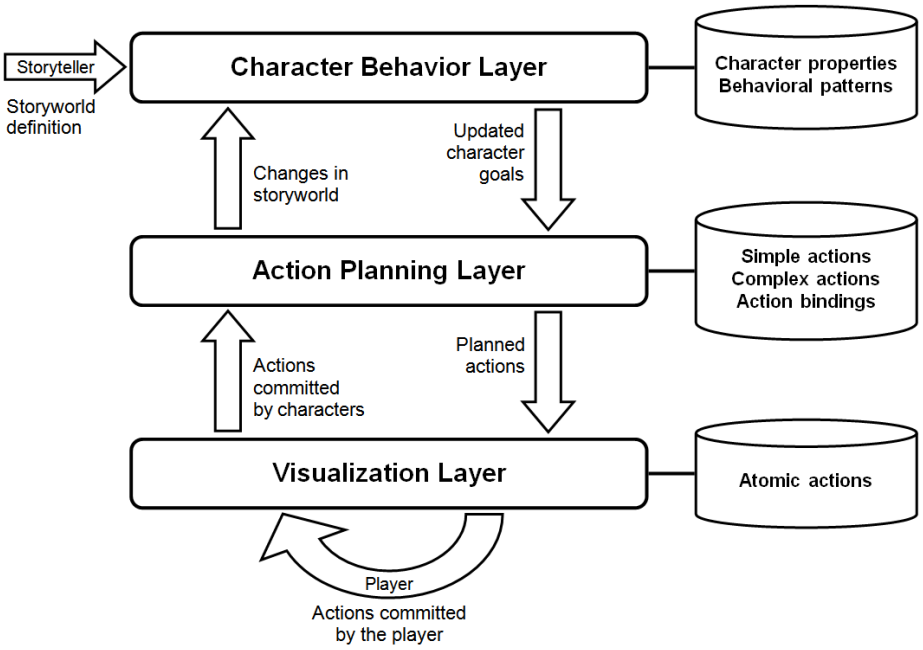


Fig. 3. The story generation cycle.

Such goals are afterwards processed by the *Action Planning Layer*, which translates all player's and non-player characters' active goals to plans. The planning process is based on the Hierarchical Task Network (HTN) planning formalism that recursively finds appropriate actions, effects of which meet the required conditions.

The planning process starts out with the effects of each active goal and finds any complex, simple or atomic actions (in this order) that have their effects match the goal's effects. The whole planning process afterwards runs recursively to search for actions that have their effects match the preconditions[8] of the newly found actions.

[8] Since the planner matches the required preconditions with effects of candidate actions, all actions depicted in Fig. 4 have preconditions placed below them and effects above.

As shown in Fig. 4, the resulting plan for every active goal is a tree-like graph that contains multiple paths of complex, simple or atomic actions, which result in accomplishing the particular goal when followed in the storyworld.

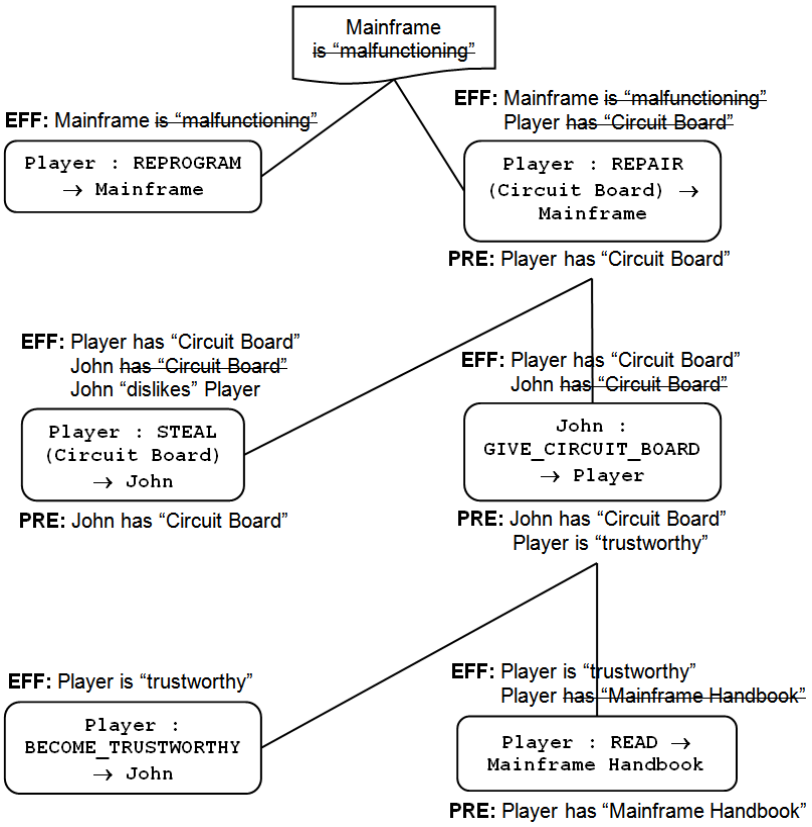


Fig. 4. An example plan[9] consisting of six simple actions that are interconnected by common preconditions (PRE) and effects (EFF), with the goal being the plan's root. The plan is constructed from top to bottom, but carried out from bottom to top.

After plans for the player and all non-player characters are constructed, a subset of all available player's planned paths is chosen and delegated to the *Visualization Layer*, along with any planned actions that each non-player character should commit.

The subset of the player's plan is chosen according to the player's user model, which contains records of all actions that he had previously successfully committed in

[9] The plan is from a storyworld in an educational domain related to the history of computing and programming basics, in which the player is to repair a malfunctioning mainframe computer, either by reprogramming it (see Fig. 6) or by repairing it with a spare circuit board, which he can either steal or get from John, a non-player character, upon becoming trustworthy by successfully answering a mainframe-related question or by reading a mainframe handbook.

other storyworlds. Thanks to these records, it is possible to filter out and ignore planned paths containing similar actions that the player previously successfully committed (*positive personalization*), or actions he had not yet committed or failed to commit successfully (*negative personalization*). If the player's user model contains no records, then the subset of player's planned paths is selected randomly.

In the example plan shown in Fig. 4, presuming positive personalization and that the player's user model states that the player had previously committed the simple action `REPROGRAM`, then the path with the simple action `REPROGRAM` is chosen and delegated to the *Visualization Layer*, as shown in Fig. 5.

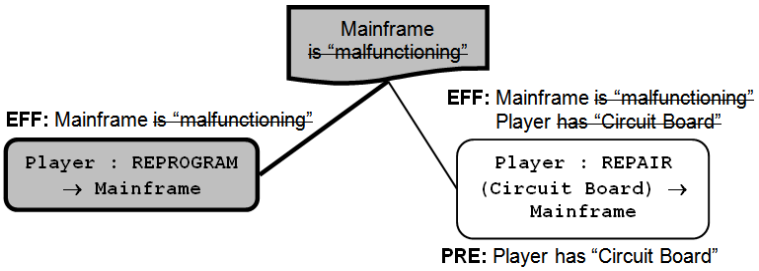


Fig. 5. An example of a chosen path, colored grey (the rest of the player's plan is omitted).

The *Visualization Layer* receives planned actions for both the player and non-player characters (NPCs) from the *Action Planning Layer* and the actions destined for NPCs are committed by the particular NPCs, whereas the actions planned for the player are presented to him as multiple options via the game's graphical interface.

The player commits, either successfully or unsuccessfully, his desired action (that may not have been planned, but must have its preconditions met), what is afterwards signaled back to the *Action Planning Layer*, along with actions that were, either successfully or unsuccessfully, committed by any non-player characters.

All committed actions are processed by the *Action Planning Layer* that matches all committed actions to player's and non-players' existing plans, which remain either unaffected, or are replanned according to the new state of the storyworld, i.e. new planned paths are again chosen based on the player's user model, as described in the preceding paragraphs.

Any alterations to the state of the storyworld, such as changes in characters' attributes, roles or relationships are signaled to the *Character Behavior Layer*, which collects all changes to the storyworld from the *Action Planning Layer* and determines whether any of the existing goals are affected (in terms of having been accomplished or not being accomplishable anymore), or whether preconditions for any new goals are met. The current set of both the player's and non-player characters' goals is updated and any changes are delegated back to the *Action Planning Layer*.

The hereby-described process is repeated in a cyclic manner until the player has successfully accomplished all of his goals, in which case the story ends. An ending also occurs if the player is unable to accomplish all of his existing goals and there are no more possible paths left to do so.

4 Conclusions and Future Work

We have described an innovative approach to interactive storytelling that uses techniques common in this field in such a unique way that, in contrast to all existing solutions, operates on simple-to-define rules, and enables to programmatically generate interactive stories in computer role-playing games.

We have implemented a prototype system based on the approach described in this paper. The domain that we have selected for prototyping is education related to the history of computing and programming basics.

The prototype system reads from an input file a set of rules and data structures (described in detail in [16]) defining a storyworld from which an educational interactive story is generated and presented to the player through a computer role-playing game (see Fig. 6). The story progresses on-the-fly by reacting to narrative actions that the player had committed.

Fig. 6. Screenshots from an educational interactive story generated by the prototype system. The top-left picture shows the player while reprogramming a mainframe computer by constructing valid program code from multiple choices of possible commands. The bottom-right picture depicts an example environment in which the generated interactive stories take place.

Our future work consists of evaluating the prototype system empirically by players who will fill out questionnaires regarding various narrative and educational aspects of the generated interactive stories.

Acknowledgments. This work was partially supported by the Cultural and Educational Grant Agency of the Slovak Republic, grant No. KEGA 3/5187/07 and by the Scientific Grant Agency of Slovak Republic, grant No. VG1/0508/09.

References

1. Crawford, C.: Chris Crawford on Interactive Storytelling. New Riders Games, (2004)
2. Denning, S.: The Springboard: How Storytelling Ignites Action in Knowledge-Era Organizations. Butterworth-Heinemann, (2000)
3. Sandercock, L.: Out of the Closet: The Importance of Stories and Storytelling in Planning Practice. Planning Theory & Practice, vol. 4, no. 1, 11–28 (2003)
4. Schank, R.: Tell Me a Story: Narrative and Intelligence. Northwestern University Press, Evanston (1995)
5. Szilas, N.: The Future of Interactive Drama. In: Proceedings of the Second Australasian Conference on Interactive Entertainment, pp. 193–199. Creativity & Cognition Studios Press, Sydney (2005)
6. Murray, H.J.: From Game-Story to Cyberdrama. FirstPerson: New Media as Story, Performance and Game, pp. 2–11. MIT Press, Cambridge (2004)
7. Rollings, A., Adams, E.: Andrew Rollings and Ernest Adams on Game Design. New Riders Games, (2003)
8. Storytron, http://www.storytron.com/
9. Cavazza, M., Charles, F., Mead, S.J.: Sex, Lies, and Video Games: An Interactive Storytelling Prototype. In: Proceedings of the 2002 AAAI Spring Symposium, pp. 13–17. AAAI Press, Menlo Park (2002)
10. Façade. http://www.interactivestory.net/
11. Barton, M.: Dungeons and Desktops: The History of Computer Role-playing Games. A K Peters Ltd, Wellesley (2008)
12. Tychsen, A.: Role Playing Games – Comparative Analysis Across Two Media Platforms. In Proceedings of the Third Australasian Conference on Interactive Entertainment, pp. 75–82. Murdoch University, Perth (2006)
13. Howard, J.: Quests: Design, Theory, and History in Games and Narratives. A K Peters Ltd, Wellesley (2008)
14. Bieliková, M., Divéky, M., Jurnečka, P., Kajan, R., Omelina, Ľ.: Automatic Generation of Adaptive, Educational and Multimedia Computer Games. Signal, Image and Video Processing, vol. 2, no. 4, 371–384 (2008)
15. Cavazza, M., Charles, F., Mead, S.J.: Planning Characters' Behaviour in Interactive Storytelling. Journal of Visualization and Computer Animation, vol. 13, no. 2, 121–131 (2002)
16. Divéky, M., Bieliková M.: An Approach to Interactive Storytelling and its Application to Computer Role-playing Games. In Návrat, P., Chudá, D., (eds.) Znalosti 2009: Proceedings of the eighth annual conference, pp. 59–70. STU, Bratislava (2009)

F. ELECTRONIC MEDIUM CONTENTS

Directory	Contents
/Documents	*Documents and papers…*
/DP1	Electronic version of the Diploma Project I report
/DP2	Electronic version of the Diploma Project II report
/DP3	Electronic version of this document
/EC-TEL 2009	Paper submitted to the EC-TEL 2009 conference
/IIT.SRC 2009	Paper and poster presented at the IIT.SRC 2009 conference
/Znalosti 2009	Paper presented at the Znalosti 2009 conference
/Prototype	*Source and binary code of the implemented prototype…*
/Bin	Binary (executable) form of the prototype
/Source	Source code of the prototype