

HERIOT-WATT UNIVERSITY



Towards A Crowdsourced Solution For The Authoring Bottleneck In Interactive Narratives

Michael Kriegel

April 2015

SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE
ON COMPLETION OF RESEARCH IN THE
DEPARTMENT OF MATHEMATICS,
SCHOOL OF MATHEMATICAL AND COMPUTER SCIENCES.

The copyright in this thesis is owned by the author. Any quotation from the thesis or use of any of the information contained in it must acknowledge this thesis as the source of the quotation or information.

Abstract

Interactive Storytelling research has produced a wealth of technologies that can be employed to create personalised narrative experiences, in which the audience takes a participating rather than observing role. But so far this technology has not led to the production of large scale playable interactive story experiences that realise the ambitions of the field. One main reason for this state of affairs is the difficulty of authoring interactive stories, a task that requires describing a huge amount of story building blocks in a machine friendly fashion. This is not only technically and conceptually more challenging than traditional narrative authoring but also a scalability problem.

This thesis examines the authoring bottleneck through a case study and a literature survey and advocates a solution based on crowdsourcing. Prior work has already shown that combining a large number of example stories collected from crowd workers with a system that merges these contributions into a single interactive story can be an effective way to reduce the authorial burden. As a refinement of such an approach, this thesis introduces the novel concept of Crowd Task Adaptation. It argues that in order to maximise the usefulness of the collected stories, a system should dynamically and intelligently analyse the corpus of collected stories and based on this analysis modify the tasks handed out to crowd workers.

Two authoring systems, ENIGMA and CROSCAT, which show two radically different approaches of using the Crowd Task Adaptation paradigm have been implemented and are described in this thesis. While ENIGMA adapts tasks through a realtime dialog between crowd workers and the system that is based on what has been learned from previously collected stories, CROSCAT modifies the backstory given to crowd workers in order to optimise the distribution of branching points in the tree structure that combines all collected stories. Two experimental studies of crowdsourced authoring are also presented. They lead to guidelines on how to employ crowdsourced authoring effectively, but more importantly the results of one of the studies demonstrate the effectiveness of the Crowd Task Adaptation approach.

Acknowledgements

I am deeply grateful to my supervisors Ruth Aylett and Sandy Louchart for bearing with me for all this time, teaching me so much about this topic, always having an open door to discuss problems and provide advice and for giving me many opportunities to connect to the Interactive Storytelling research community through conference visits. A big thank you to my examiners Dr. Mariët Theune and Dr. Yun-Heh (Jessica) Chen-Burger for showing great interest in my work and providing much appreciated feedback and a very constructive and inspiring discussion during my viva. And of course also for awarding me the PhD title.

I would like to thank my colleagues and fellow students from the Computer Science Department at Heriot-Watt University for invaluable discussions and feedback. Particularly Asad, Amol, Christopher, Iain, Rob, Irene, Matthias, Shahira and Patricia, all of whom I shared an office with at some time and the Storylab guys Allan, Neil and Andy all gave me many opportunities to bounce off ideas.

During the e-Circus project I had the chance to be part of a multidisciplinary and multinational team doing groundbreaking work in educational IS applications that also motivated and spawned this PhD. Being part of this team was a fantastic experience and I want to take this opportunity to acknowledge the many talented and genuinely nice people I had the pleasure to work with. I worked especially close with Joao, Rui and Marco of the GAIPS research group at INESC-ID, Lisbon, who taught me much about the ins and outs of FAtiMA and who I was lucky enough to be able to visit several times for collaborative work.

Ivo Swartjes, formerly a PhD student and researcher at University of Twente, on his research visit to our department provided me with a wonderfully enlightening collaboration opportunity and helped in shaping my views on Interactive Storytelling that inform this work. Alasdair Clarke and Scott Watson are both much more well versed in statistics than me and deserve my thanks for having kindly answered my questions regarding experimental design and statistical analysis. I would also like to thank all the volunteers who participated in one of the several authoring experiments that this thesis describes. Without their help this research would have been impossible to complete.

The work on this PhD has accompanied me for the better part of a decade. This is

a long time and along the way life has produced many distractions, while the PhD work itself has also not been without its diversions, dead ends and desperate moments. Most importantly, I would therefore like to express my eternal gratitude to my family. This includes my parents, my sister, my two brothers-in-law and my parents-in-law who were all fantastic in encouraging me along the way. I am especially indebted to my dad who took it upon himself to proof-read this thesis and gave me much appreciated feedback. But my most heartfelt thanks have to go to my wonderful wife Mei Yui and daughter Alicia, whose support and understanding allowed me to find enough time to focus on writing this thesis and who patiently suffered with me through the final stretches of this work, when many evenings, weekends and holidays had to be sacrificed for this endeavour. Now, as I'm writing these words, that is almost done and I am looking forward to give them and our newest family member Sebastian the time as husband and father that they deserve.

ACADEMIC REGISTRY

Research Thesis Submission



Name:			
School/PGI:			
Version: <i>(i.e. First, Resubmission, Final)</i>		Degree Sought (Award and Subject area)	

Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

- 1) the thesis embodies the results of my own work and has been composed by myself
- 2) where appropriate, I have made acknowledgement of the work of others and have made reference to work carried out in collaboration with other persons
- 3) the thesis is the correct version of the thesis for submission and is the same version as any electronic versions submitted*.
- 4) my thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
- 5) I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.

* *Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.*

Signature of Candidate:		Date:	
-------------------------	--	-------	--

Submission

Submitted By <i>(name in capitals)</i> :	
Signature of Individual Submitting:	
Date Submitted:	

For Completion in the Student Service Centre (SSC)

Received in the SSC by <i>(name in capitals)</i> :			
Method of Submission <i>(Handed in to SSC; posted through internal/external mail):</i>			
E-thesis Submitted (mandatory for final theses)			
Signature:		Date:	

Contents

List of Figures	vii
1 Introduction	1
1.1 Motivation	1
1.1.1 Interactive Digital Storytelling	1
1.1.2 Story Worlds and Runtime Engines	2
1.1.3 Fully Realized IS Artefacts	3
1.1.4 The Authoring Bottleneck	4
1.2 Crowdsourced Authoring	5
1.2.1 Collecting Linear Example Stories	6
1.2.2 The Crowd Task Adaptation Hypothesis	6
1.3 Summary Of Contributions	7
1.4 Thesis structure	8
I The Authoring Bottleneck	10
2 Creating FearNot! - An Authoring Case Study	11
2.1 FearNot! Overview	12
2.1.1 FearNot Creation Stages	13
2.2 An Overview of the FAtiMA Architecture	15
2.2.1 Planning	15
2.2.2 Emotions	17
2.2.3 Reactive Behaviour	19
2.2.4 Agents and their environment	20
2.2.5 Personality	21
2.3 Integrating FAtiMA into an IS Runtime Engine	22
2.3.1 The Story Facilitator	24
2.4 Story World Contents	27
2.4.1 Character Configuration Content	27
2.4.2 Dialogue Content	28
2.4.3 Interaction Rules	29

2.4.4	Presentational Content	29
2.4.5	Story Structure Content	30
2.5	FearNot! Authoring Process	30
2.5.1	Resources	30
2.5.2	Workflow	31
2.5.3	Knowledge Representation Encoding	32
2.6	Authoring Observations	36
2.6.1	Actual Use of Planning	36
2.6.2	Decision Making	37
2.6.3	Use of Emotion Model	39
2.6.4	Interactivity	41
2.6.5	Content Reuse and Abstraction	42
2.7	Conclusion	43
3	Data Structures for Story Representation	44
3.1	Explicit Specification of Branches	45
3.2	Plot-centric Story Representation	47
3.2.1	Planning	47
3.2.2	Case-Based Reasoning	49
3.2.3	Narrative Theories	50
3.2.4	User Models	52
3.3	Character-centric Story Representation	53
3.3.1	Agents Using STRIPS-like Planning	53
3.3.2	Agents Using Hierarchical Task Networks	54
3.3.3	Agents Using Heuristic Search Planning	55
3.3.4	Decision Theoretic Agents	57
3.3.5	Other Approaches	58
3.4	Hybrid Solutions	62
3.4.1	Character Autonomy For Plot-Centric Systems	62
3.4.2	Drama Management For Character-Centric Systems	64
3.5	Conclusion	66
4	Authoring Methods And Tools	68
4.1	Defining The Authoring Process	68
4.1.1	Who are the authors?	68
4.1.2	Authoring Metaphors	69
4.2	Authoring Tools	71
4.2.1	Prototyping	72
4.2.2	Authoring Explicit Branching	73
4.2.3	Authoring Generative Data Structures	74

4.2.4	Debugging	76
4.2.5	Affecting Story Presentation	77
4.2.6	Educational Authoring	78
4.3	Data Driven Authoring	78
4.3.1	The Restaurant and Improviso	79
4.3.2	Scheherazade	81
4.3.3	Comparison	82
4.4	Conclusion	83
II	Crowdsourced Authoring	84
5	Crowd Task Adaptation	85
5.1	Defining Crowdsourcing	85
5.1.1	Crowdsourcing and other crowd-powered approaches	85
5.1.2	Storytelling Examples	87
5.2	Attacking the Authoring Bottleneck with a Crowd	90
5.2.1	Useful Data	90
5.2.2	Finding a suitable crowd-powered approach	94
5.3	Crowd Task Adaptation: A Novel Process Improvement Proposal	96
5.3.1	Definition	97
5.3.2	Crowd Task Adaptation Strategies for IS Authoring	98
5.3.3	Expected Benefits	100
5.4	Conclusion	103
6	The ENIGMA Authoring System	104
6.1	Design	104
6.1.1	Overview	104
6.1.2	Storytelling Interface	106
6.1.3	Annotations	112
6.1.4	Mixed Initiative	113
6.2	Implementation	115
6.2.1	Technology Overview	115
6.2.2	User Interface	116
6.3	From Stories to FAtiMA Agents	119
6.3.1	Formalization as a machine learning problem	120
6.3.2	The probabilistic domain model	121
6.3.3	The ENIGMA learning cycle	121
6.3.4	Updating the probabilistic domain model	123
6.4	Usability Trial	125

6.4.1	Setup	126
6.4.2	Observations	126
6.4.3	Implications	127
6.4.4	Ways Forward	128
6.5	Conclusion	128
7	The CROSCAT Authoring System	130
7.1	Overview	131
7.1.1	User Perspective	131
7.1.2	System Perspective	132
7.2	The Back Story Selection Algorithm	134
7.2.1	Heuristic Function 1: Distance From Branching Points and Leafs	135
7.2.2	Heuristic Function 2: Adding Branching Ancestor Penalty . .	138
7.2.3	Heuristic Function 3: Adding Child Branch Penalty	139
7.2.4	Other Heuristics	140
7.3	Antonym Insertion	141
7.3.1	Using Antonyms	143
7.3.2	Obtaining Antonyms	145
7.4	Algorithm Implementation	146
7.5	Scalability	147
7.5.1	Concurrent Use	148
7.5.2	Scoring Algorithm Performance	149
7.6	The CROSCAT Viewer	149
7.7	Conclusion	150
III	Authoring Experiments	151
8	Study Descriptions	152
8.1	The Point Nautilus Study	152
8.1.1	Story Setting	153
8.1.2	Execution	154
8.1.3	Data Sources For Evaluation	155
8.2	The Seagnomes Study	156
8.2.1	Participant Assignment	157
8.2.2	Story Setting	159
8.2.3	Execution	160
8.2.4	Data Sources For Evaluation	162
8.3	Data Analysis Plans	165
8.3.1	Primary Research Question	165

8.3.2	Secondary Research Questions	166
8.4	Conclusion	167
9	Experimental Results And Analysis	168
9.1	Evaluating Crowd Task Adaptation	168
9.1.1	Seagnomes Author Profiles	168
9.1.2	Seagnomes Story Graphs	169
9.1.3	Seagnomes Interactive Story Ratings	171
9.1.4	Story Analysis	174
9.1.5	End User Feedback	178
9.1.6	Choice Analysis	179
9.1.7	Author Feedback	181
9.2	Story World Properties	183
9.2.1	Results of The Point Nautilus Study	183
9.2.2	Comparison of Point Nautilus and Seagnomes stories	186
9.3	Authoring Tool Design Lessons	187
9.3.1	User Interface	187
9.3.2	Content Library	188
9.3.3	Accommodating Deliberation	189
9.3.4	Storytelling Modality	190
9.4	Conclusion	191
10	Reflection	193
10.1	Summary of Thesis	193
10.2	Practicality of Crowdsourced Authoring	195
10.3	Future Work	197
10.3.1	Specific Follow-ups	197
10.3.2	General Follow-ups	198
10.4	Concluding Remarks	199
	Glossary	200
	References	201
	Appendices	217
	A Complete List of FearNot! agent goals	218
	B The Point Nautilus Story	221
	C The Seagnomes Story	226

CONTENTS

D CROSCAT usage instructions	229
E CROSCAT Graph Scoring Implementation	231
F Supplementary Digital Materials	234

List of Figures

1.1	Spierling’s and Szilas’s definition of the boundaries of authoring . . .	2
2.1	Screenshots of FearNot!	13
2.2	Major FearNot! development stages.	14
2.3	Illustration of the example cookie planning domain.	16
2.4	Complete plan in the example cookie domain.	17
2.5	The OCC emotion taxonomy	18
2.6	High-level FAtiMA architecture diagram	20
2.7	High-level components of storytelling system used in FearNot!	22
2.8	FearNot! authoring workflow	31
2.9	Iterative interactive implementation cycle	35
3.1	Illustration of a plan in the MIMESIS system	48
3.2	Freytag’s and Campbell’s Narrative Theories	51
3.3	An example dramatic arc generated by Façade	51
3.4	Example of an HTN used by the IStorytelling System	55
3.5	Interpretation operator in the EmoEmma system	56
3.6	Screenshots of SWAT	59
3.7	High-Level Overview of Comme-il-faut story world components	60
3.8	Architectural overview of the Virtual Storyteller	65
3.9	Example sentence in DEIKTO	67
4.1	Different authoring roles suggested by Hoffmann et al.	69
4.2	Landscape view of an IS story world	69
4.3	Gardening metaphor for implicit creation of IS story worlds	71
4.4	The Scene Graph View in the U-Create tool	73
4.5	ASAPS story graph editor	74
4.6	Wide Ruled: Plot Fragment Editor	75
4.7	Story Canvas: Plot Fragment Editor	76
4.8	Thespian Goal Fitting Procedure	79
4.9	Orkin’s Restaurant and Improviso Games	80
4.10	A Scheherazade Plot Graph	81

LIST OF FIGURES

5.1	Crowdsourcing and related crowd-powered approaches	86
5.2	Screenshots of the Bar Karma Authoring Tool StoryMaker	91
5.3	Tradeoff between authoring effort and transformation effort	93
5.4	Visualization of the Crowdsourcing process	97
6.1	Enigma Basic Workflow	105
6.2	Comparison of the user’s role in a FAtiMA based IS artefact and in ENIGMA.	107
6.3	Metadata associated with an action in ENIGMA	108
6.4	Metadata associated with the GiveGift action	108
6.5	Screenshot of initial ENIGMA prototype using a 3D game engine . .	110
6.6	Example of 2 frames using ENIGMA’s comics story presentation layer	110
6.7	ENIGMA’s user interface: main window	116
6.8	ENIGMA’s user interface for action selection and definition	117
6.9	ENIGMA’s user interface for annotating changes	118
6.10	ENIGMA’s user interface for annotating goals	119
6.11	Overview of the FAtiMA domain model learning approach for ENIGMA	122
7.1	Main graphical user interface of the CROSCAT Tool.	131
7.2	Overview of the Crowd Task Adaptation via back story selection feature in the CROSCAT authoring tool.	133
7.3	Branching story tree if no back story selection would be used.	134
7.4	Laurel’s view of narrative as a flying wedge of possibilities	136
7.5	CROSCAT simulation results using heuristic function 1	137
7.6	CROSCAT simulation results using heuristic function 2	138
7.7	CROSCAT simulation results using heuristic function 3	140
7.8	Example of a story fragment containing a dramatic choice point . . .	142
7.9	CROSCAT story tree with added antonym knowledge	143
7.10	Simulation results for antonym weight	144
7.11	A choice visualized in the CROSCAT Viewer	150
8.1	Locations for the Seagnomes Story	160
8.2	Instructions for the Seagnomes Study	161
9.1	Author background questionnaire results for the “Seagnomes” study .	169
9.2	The resulting story graph for condition “graph”	169
9.3	The resulting story graph for group “anto”	170
9.4	The resulting story graph for condition “none”	170
9.5	Between-Group Interactive Story Ratings	172
9.6	Within-Subject Interactive Story Ratings	172
9.7	Story Length Depending On Backstory	174

LIST OF FIGURES

9.8	Example of author’s resistance to being steered	177
9.9	Choices made by the viewers of version “graph”	180
9.10	Choices made by the viewers of group “anto”	181
9.11	Feedback On The Authoring Experience	182
9.12	“Point Nautilus” Authors’ Familiarity With Related Activities (averages)	183

Chapter 1

Introduction

1.1 Motivation

1.1.1 Interactive Digital Storytelling

Storytelling always was and continues to be an important cultural tradition central to our identity as human beings. Advances in technology have often lead to the invention of new storytelling media, allowing us to experience narratives in new ways. The printing press enabling the novel, the camera leading to movies, or the computer being a prerequisite for video games are just three of the numerous examples of how technology has transformed our experience of narratives. Advances in artificial intelligence and virtual reality technologies are currently resulting in efforts to create another storytelling medium: “Interactive Storytelling” (IS).

Within the context of this thesis, IS refers to computer mediated dramatic narrative experiences in which the audience can influence the narrative in a meaningful way. This contextualized definition is necessary, as there is no fully agreed upon terminology within the field. In the literature the terms Interactive Narrative, Interactive Drama and Interactive Storytelling are often used interchangeably and all 3 could fit the above definition within this thesis. Similarly there are several types of narrative experience that are sometimes labelled Interactive Storytelling, that do not fit the above definition. These include non computer mediated experiences, for example roleplaying, improvisational theatre or choose your own adventure books and computer mediated experiences that offer rather linear narrative experiences (e.g. most video games).

In IS the audience takes the role of a user instead of spectator as it is the case in traditional narrative media such as literature, theatre or film. By incorporating a participating user (usually playing the role of one of the protagonists), stories need to adapt dynamically to the choices of the user. Science Fiction has already shown us the possibilities on offer if this new medium reaches maturity: In the fictional

Star-Trek universe, the holodeck is a recreational technology that allows people to step into a perfectly simulated Virtual Reality and experience stories from a first person perspective as protagonists with total freedom of choice. Murray (1998) helped fostering the holodeck as a metaphor for the aspirations of the interactive drama community, with her seminal “Hamlet on the Holodeck”.

IS not only promises to provide more immersive entertainment than traditional narrative but also opens up many potential applications in education. Right now IS technology is in its infancy, maybe comparable to the state of movies in the early 20th century. We will never build a holodeck without first mastering the general hard problems of the enabling technologies of artificial intelligence and virtual reality. However, that is not a reason for not investigating the questions surrounding IS right now and a growing multidisciplinary research community is trying to find ways of progressing on the long path towards the holodeck. This thesis adds to that growing body of work.

1.1.2 Story Worlds and Runtime Engines

Complex computer programs usually separate their implementation, logic and algorithms from the data the program processes at runtime and IS systems are no exception. In order to talk about these distinctions we borrow the terminology from Spierling and Szilas (2009), who use the terms **Runtime Engine** and **Story World** to refer to processes and data in the context of IS. A story world is a library of story elements that a Runtime Engine can assemble interactively into a story, taking into account user input. The bundle of a Runtime Engine and a specific story world in this terminology is referred to as **IS Artefact**. An IS Artefact is the playable product that is distributed to end users, for whom an interaction session with it results in an **IS Experience**. Figure 1.1 summarizes these relationships.

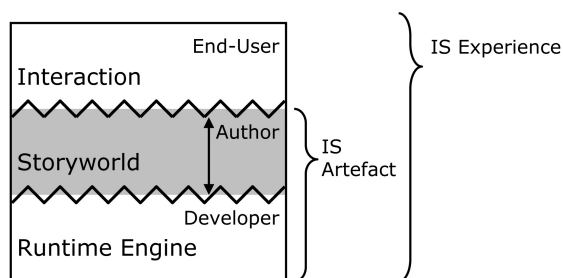


Figure 1.1: Spierling’s and Szilas’s definition of the boundaries of authoring. (from Spierling and Szilas (2009) p.52)

While IS research strives to produce IS Artefacts capable of generating vast numbers of different story experiences depending on a user’s input, there are implicit limits to the story experiences that can come out of a single IS Artefact. Within this

definition an IS Artefact always revolves around a certain theme, conflict, setting or cast of characters. For example one could imagine an artefact about the adventures of a cowboy in the wild west. No possible user interaction with this artefact would ever produce a story experience about the struggles of a football team in winning a championship. This is because no information about football would be encoded in the wild west story world that could allow the runtime engine to generate such a story. It is, however, perfectly possible to use the same runtime engine for running many different story worlds, e.g. a wild west and a football themed one.

This distinction between runtime engine and story world also extends to the roles assigned to the creators of an IS Artefact. Most IS work stems from computer science research labs where computer scientists devise algorithms for narrative generation and/or manipulation and implement them in a runtime engine. However, as runtime engines are typically agnostic of any specific narrative content, the term **author** in IS is generally used to refer to the creator of the story world and not the runtime engine. Small story worlds are often created by the same researchers who created a runtime engine as a proof of concept and for evaluation purposes. However, most of the IS community will readily admit that these small story worlds lack the breadth and complexity to create impressive IS artefacts. The hope is that third-party authors (quite possibly with a skill set different from the computer scientists who generate the runtime engines and more rooted in creative writing) will take up a runtime engine and create story worlds for it. So far, this has not really happened.

1.1.3 Fully Realized IS Artefacts

In order to distinguish current systems from the ideal that the medium aspires to be, we use the term fully realized IS Artefact to describe the type of experience that the IS community is striving to create. Obviously this is a very subjective distinction as we are talking about a medium for artistic expression, but there is wide agreement that so far no fully realized IS artefact exists. And while no formal definition exists of what a fully realized IS artefact exactly is, the set of the following minimal criteria for it are generally agreed upon in the literature (Murray, 1998; Mateas and Stern, 2003; Cavazza et al., 2004; Crawford, 2005):

- **Agency** The user must feel like they have meaningful control over the development of the narrative and the system must provide them with enough possibilities for interacting. If the user interacts by controlling a character, most likely the protagonist of the story, they should ideally be able to make the character do whatever they chose and whenever they chose it. But agency is not so much about the quantity as it is about the quality of choices. For example in a first person shooter game we control every step of our character but have no agency

when it comes to the story. There is nothing the user can do but shoot or be shot.

- **Variety** An IS artefact is only fully realized if its possibility space cannot easily be exhausted by its users. I.e. it must be complex enough so that different users experience different personalised stories. A single user must be able to revisit/replay the IS artefact multiple times to experience alternative versions of events. Modern story based video games lack this kind of replayability. Of course video games may be replayed many times, just as books may be reread many times, but after at most a handful of play-throughs the player will no longer experience any new story content.
- **Coherence** Each story produced by the system needs to be coherent and logical. Events in the story need to be causally connected and characters need to act rationally.
- **Drama** A system that produces replayable, i.e. highly variable and coherent stories is still not enough. Each possible path a user could take through the interactive story space should ideally also have the quality of a “story” and be dramatic, i.e. contain conflict and resolution.

The real difficulty lies in addressing all of these criteria together. Focusing on a single criterion and taking measures to improve it will most likely go at the expense of one or more of the other criteria. For example a system that aims to achieve variety by randomly stringing together events will suffer from a negative impact on coherence as randomly assembled event sequences will rarely result in coherent stories. The tension between the criteria of Agency and Drama in particular has been the subject of much debate and is often called the narrative paradox (Aylett, 2000)

1.1.4 The Authoring Bottleneck

The ideal of a fully realized IS artefact, can only be achieved by a storytelling system that is backed up with a huge library of content from which it can construct a variety of alternative plots. This content could either take the form of an explicit enumeration of all the possible plots or be specified in a more implicit representation. The former is usually considered impractical, due to the combinatorial explosion of possibilities and the effort it would take to write them all out, which is why most IS systems take the latter approach. In that case the library needs to contain knowledge and facts about the world the story takes place in and the characters that inhabit this world. While this implicit approach is regarded as more scalable, it also results in a very different role for the author: their work becomes more that of a knowledge engineer

than that of traditional novelists, poets, playwrights or screen writers. Instead of detailing a plot from beginning to end, the author has to describe a world in which the possibility for numerous hypothetical plots exists. Since this description has to happen in a way understandable by the computer, computer programming might be a better analogy than writing to describe the author's activity within this framework. This process of authoring, has been recognized as one of the main bottlenecks in the creation of fully realized IS artefacts. Pizzi and Cavazza (2008) for example state that "the issue of content creation is lagging behind the early technical developments and their proof-of-concept prototypes" while Spierling and Szilas (2009) call authoring "one of the main challenges that have recently been discussed at Interactive Storytelling conferences". The authoring bottleneck does not only prevent the creation of fully realized IS artefacts but also hinders IS research progress since research topics such as story generation algorithms, user experience or the narrative paradox cannot easily be studied on a small scale. There is much work to be done in characterizing, understanding and improving the authoring process.

1.2 Crowdsourced Authoring

This thesis presents an exploration of the nature of the authoring bottleneck and suggests a crowdsourced authoring process as a potential solution. The term "Crowdsourcing", coined by Howe (2006), refers to work processes that rely on large scale division of labour. Large or tedious tasks are divided into small subtasks, which are then completed by members of a crowd, typically online. The rationale for using crowdsourcing to address the authoring bottleneck problem is straightforward and intuitive. Distributing the authorship for IS among a large enough group, should result in a large corpus of useful data, even when any one individual only makes very minor contributions.

An IS artefact created in such a way can of course not be the creative vision and product of a single mind. Critics may argue that such a method of creation is incapable of producing works of art, as it replaces the artist's determined focus with a muddled mass consensus. They may be right, but IS is too young and immature as a storytelling medium to assess what constitutes artistically valuable IS artefacts. It seems plausible that the unique IS property of providing stories that develop according to the individual reader's choices fits a distributed authorship model: As there is an almost infinite story space to fill, there is enough room for every contribution from the crowd. The existence of the authoring bottleneck seems to suggest that there is in any case no way around a collaborative creation of fully realized IS artefacts. Similar to software or film production, the scale of work necessary is beyond what a single individual can reasonably contribute. Without significant advances in Artificial

Intelligence that allow the authoring to be fully automated this is not likely to change anytime soon.

In light of this inevitability of collaboration, the exploration of a crowdsourcing approach to IS authoring seems a worthwhile endeavour that should be of interest for the wider IS community. Several interesting research questions follow from this line of enquiry, e.g. how should the authoring task be divided, what data exactly should be collected, how should it be processed, how is the collection process organised or how to assure quality. The work presented here addresses all these questions, though it does not claim to answer them definitely.

1.2.1 Collecting Linear Example Stories

Orkin and Roy (2007) demonstrated the potential of crowdsourced authoring with the “Restaurant Game”, a video game in which two players role-play the interactions between a waitress and a customer in a restaurant setting. Thousands of game play sessions were recorded and used to learn behaviour patterns for autonomous computer-controlled characters (Orkin and Roy, 2009). This work has opened up fertile ground for further exploration of the topic and more recently Li et al. (2012) have started to explore similar territory with the Scheherazade system.

Both these systems share the same approach to subdividing the authoring task: contributors from the crowd are asked to provide linear example stories, which a computer system collects and automatically merges into a story world. Asking authors to contribute on the level of linear stories provides a natural way for them to think about narrative and thus greatly reduces the learning barrier. Authors do not need to understand IS in order to make useful contributions. By relinquishing control to an automated system that merges multiple collected stories into an IS story world, one loses some confidence in what kind of experience the resulting IS artefact will be, but gains productivity and scalability. This thesis deals with the design of such authoring solutions and how they can be made more efficient.

1.2.2 The Crowd Task Adaptation Hypothesis

As mentioned above, the idea of approaching the authoring of interactive story worlds as a crowdsourced collection of linear example stories is shared with other work and cannot therefore in itself be this thesis’ claim to novelty. The central novel aspect that the work within this thesis adds to this approach is the concept of “Crowd Task Adaptation” and its applicability to the authoring of interactive story worlds.

Crowd Task Adaptation is a term we define in this thesis and refers to the simple premise that an IS system, which makes use of crowdsourcing dynamically modifies the tasks handed out to crowd workers based on the corpus of collected data to date.

This thesis' central **research hypothesis** is that by applying Crowd Task Adaptation, such a system may gain some additional value from its collected data compared to the value it would have derived from it otherwise. In other words, we hypothesize that Crowd Task Adaptation can make the collected data more valuable.

This notion of added value may sound somewhat vague but that is due to the fact that there are many different possible ways in which to implement the idea of Crowd Task Adaptation and the benefits gained are likely differing depending on the chosen implementation. Later in this thesis we discuss both different variants of Crowd Task Adaptations and the different kind of benefits we believe to be gainable from employing Crowd Task Adaptation. They include for example increased variety and homogeneity in the created story world. We also show some concrete implementations of Crowd Task Adaptation in two IS authoring systems, which were implemented in the course of this work.

1.3 Summary Of Contributions

The work presented in this thesis makes several novel contributions to the state of the art in IS research. Most importantly it introduces the theoretical concept of Crowd Task Adaptation and its application to authoring interactive stories. This is novel since previous crowdsourcing approaches to IS authoring such as the ones mentioned above have always used static tasks. This discussion does not remain theoretical but is given weight through the implementation in two concrete IS authoring systems: ENIGMA and CROSCAT. The latter system was used to obtain experimental results from crowdsourced authoring studies. These results confirm our research hypothesis. They clearly demonstrate that employing Crowd Task Adaptation has improved the quality of interactive stories authored with the CROSCAT system. In its entirety this work therefore constitutes a small further step towards a solution to the authoring bottleneck.

Furthermore this thesis makes several secondary contributions:

- A concrete IS authoring case study is presented in Chapter 2, giving previously unpublished details about the construction of one of the major IS artefacts (FearNot!). This case study sheds more light on the authoring bottleneck.
- Two algorithms for adapting crowdsourced example story collection tasks are presented in this thesis. The first is used for learning planing domains from example stories with mixed initiative authoring and is described in the context of the ENIGMA authoring system in Section 6.3. The second is an algorithm for selecting backstories from a branching story tree, which is implemented in

the CROSCAT system and discussed in Section 7.2. Since the Crowd Task Adaptation concept itself is novel these algorithms are obviously novel as well.

- The ENIGMA and CROSCAT authoring systems that were exclusively written for this PhD are available to interested parties (see Appendix F for details) and could be used in future IS research.
- During the implementation and evaluation of the systems mentioned above, generic lessons about the design of IS authoring tools were learned, which are shared in this thesis.
- As far as we are aware, out of the few published pieces of research that apply crowdsourcing to IS authoring, the work in this PhD is the only research to date that focuses on collecting dramatic character-driven stories rather than scripts of typical behaviour in certain situations. In these other pieces of work, much duplication in the collected story material is needed to statistically determine what is typical behaviour. As a result other pieces of work have framed the task in a way that does not encourage dramatic improvisation as much as this work does. Section 9.2 provides a discussion backed by experimental results about some properties of stories that our approach seems particularly suited to.
- As an outcome of the experimental studies conducted in this PhD, several actual IS artefacts were authored. The creation of new IS artefacts in itself arguably constitutes a useful contribution to the research field, as especially theoretically minded IS scholars need IS artefacts to analyse. All IS artefacts created in the course of this PhD work are available to interested parties (see Appendix F).

1.4 Thesis structure

The remainder of this thesis is structured into 3 parts and 9 chapters.

Part I “The Authoring Bottleneck” consists of 3 chapters, in which the existing problems around authoring in IS systems are explored.

First, Chapter 2 motivates the work within this thesis by presenting a case study of authoring activities performed by the eCircus project team (including the author of this thesis) when creating the educational interactive bullying drama FearNot! An analysis of the problems encountered during these activities is provided.

Next, Chapter 3 reviews a multitude of other existing IS systems, focusing on the data structures for story representation employed by these systems. This will serve to demonstrate the many facets of modern IS systems and underline the universality of the authoring bottleneck problem addressed by this thesis.

Chapter 4 is a review of literature that acknowledges the authoring bottleneck and proposed solutions in the form of authoring processes and tools.

Part II “Crowdsourced Authoring” is the central part of the thesis, where crowdsourcing as a potential solution to the authoring bottleneck is discussed and crowd task adaptation is suggested as a novel modification to this approach with the potential to increase crowdsourcing effectiveness. Chapter 5 compares crowdsourcing with other scalable online collaboration approaches, discusses the merits of basing an authoring process for IS systems on crowdsourcing in general and on crowd task adaptation in particular. This is followed by 2 chapters describing concrete prototype implementations of IS authoring systems based on the crowd task adaptation principle.

Chapter 6 centres on ENIGMA, the first prototype for a collaborative authoring system built as part of this research. A first small-scale user trial revealed that the amount of work required to make this approach usable goes beyond what is achievable within the scope of a single PhD.

In Chapter 7 a second authoring system prototype, CROSCAT, is described, which was implemented following the lessons learned from the ENIGMA user trial. This system serves as the basis for the authoring study described in Section 8.2.

Part III “Authoring Experiments” puts crowdsourced authoring and the principle of crowd task adaptation, proposed in Part II to the test through the analysis of collaborative authoring studies.

Chapter 8 describes the motivation and design of two collaborative authoring studies carried out during the course of this work. In one of these experiments hand-written stories were collected, while the other experiment employs the CROSCAT authoring system for a comparison between crowdsourced authoring with and without crowd task adaptation.

Chapter 9 presents the results of these studies and in their analysis focuses on what these results can teach us about a) crowd task adaptation and b) the properties of story worlds suitable to crowd-sourced authoring.

Finally, Chapter 10 reflects upon the work presented in this thesis and draws conclusions from it.

Part I

The Authoring Bottleneck

There is nothing to writing. All you do is sit down at a typewriter and bleed.

Ernest Hemingway

Chapter 2

Creating FearNot! - An Authoring Case Study

We start exploring the issues surrounding IS authoring with a case study examining the creation of a specific system: the anti-bullying interactive drama “FearNot!”. FearNot! as an IS artefact is based on an IS Runtime Engine, which utilises the agent architecture FAtiMA (short for FearNot AffecTive Mind Architecture) (Dias and Paiva, 2005; Aylett et al., 2006a). This chapter’s discussion of how FearNot! was created is therefore primarily focused on how to encode IS story worlds in the FAtiMA architecture. During the research project *eCIRCUS*, the author was personally involved in the creation of FearNot!, both as an author and as a system designer/programmer. The FearNot! design, implementation and authoring work in itself was a joint team effort and is not claimed to be a novel contribution by this thesis. However, this chapter’s retrospective analysis of the authoring processes that were employed in producing FearNot! and the problems encountered along the way constitutes a novel contribution. These analyses are presented in Sections 2.5 and 2.6.

These experiences in IS authoring were very important motivators for the research direction this PhD has taken and thus should not be omitted from this thesis. Besides mirroring the author’s personal research journey, this chapter also serves the purpose of providing an in-depth example of an IS runtime engine and its associated story representation. Another reason for dedicating considerable space to a single system in this chapter is that the discussion in Chapter 6 of the ENIGMA authoring system will require a basic understanding of FAtiMA.

Furthermore, readers unfamiliar with interactive storytelling research will benefit from such a detailed system description to appreciate the complexity of the IS authoring task. Those already well-versed in the field of IS research on the other hand will probably recognize FearNot! as one of the more prominent IS systems and appreciate the previously unpublished details of its creation process.

While it is somewhat unconventional to start a thesis with a case study instead of a

literature review chapter, in this case it makes both chronological and didactic sense. Literature reviews detailing a variety of alternative approaches follow in the subsequent two chapters but having first discussed a single system in detail helps ground these discussions and gives them context.

When the eCIRCUS project set out to create FearNot! there were no established processes to rely on that define how one best approaches such a task. In fact, to this date this still has not changed. The workflow descriptions in this chapter are therefore representing an experimental attempt of establishing best practises for authoring an IS story world with a medium size multidisciplinary team. Critically evaluating these practises in retrospect falls very much within the research scope of the eCIRCUS project. In other words, while producing FearNot! was undoubtedly the primary goal, the authoring journey of arriving there was in itself a worthwhile endeavour that provides lessons learned and contributes to the overall knowledge of the IS research field. In some respect, this Chapter attempts to belatedly deliver this previously unpublished process analysis.

2.1 FearNot! Overview

VICTEC (Virtual ICT with Empathic Characters) and its successor eCIRCUS (Education through Characters with emotional Intelligence and Role-playing Capabilities that Understand Social Interaction) were interdisciplinary collaborative research projects exploring the use of IS technology in social and emotional learning for young primary school children. This means learning of moral lessons and empathy rather than hard facts. The FearNot! application, which applies these ideas to anti-bullying education is the primary output of the VICTEC and eCIRCUS projects. The design goal for FearNot! was to create a game-like application that would allow children to empathise with the victims of bullying and safely experiment with different coping strategies (Aylett et al., 2006b). Ideally, with the outcome that bullies using the application would learn to empathise with the victims and act less aggressively, bystanders would more readily intervene when witnessing bullying incidents and victims would gain more confidence. Interactive stories presented in game-like virtual environments promise to be a suitable medium for conveying these messages as video games have a wide acceptance in and attraction to the target audience of primary school pupils.

FearNot! conveys the problems surrounding bullying in schools by describing the trials and tribulations of a single primary school child (henceforth referred to as the victim) who is experiencing bullying in the school environment. The gender of the victim protagonist is chosen to match the user's gender in order to make it easier for the user to identify themselves with the protagonist. One of the first and most im-



Figure 2.1: Screenshots of the FearNot! antibullying application showing a dramatic episode (left) and an interaction session (right)

portant decisions when designing an IS artefact is the role of the user. The VICTEC team in their design of FearNot! did not chose the most obvious and prevalent IS interaction style “user as participant”, where the user influences the unfolding story by directing the actions of a character in the story. Instead in FearNot!, the user takes a “Spect-Actor” role, a mix between spectator and actor. Concretely in FearNot! this means that the user is an invisible friend to the bullying victim protagonist.

The story itself unfolds in several dramatic episodes (left side of Figure 2.1). During these episodes the user just observes the dramatic interactions between the characters. Interspersed between the dramatic episodes are interaction sessions, in which the protagonist consults the user and asks for advice. This consultation takes the form of a free text entry chat (right side of Figure 2.1). During this interaction the user can advice the victim on how to best deal with their bullying problems. This influences the victim’s behaviour in the subsequent dramatic episode.

The interaction structure of FearNot! was inspired by the Forum Theatre approach, pioneered by Boal (2000). In this form of dramatic performance, there are several time-out sessions from the performance during which the actors address the audience in-character and discuss the preceding events. Forum Theatre has been found to be very effective in education and thus aligns very well with what FearNot! tries to achieve.

2.1.1 FearNot Creation Stages

Figure 2.2 gives an overview of the major stages involved in creating FearNot! The division between the VICTEC and eCIRCUS projects is shown. VICTEC laid the groundwork by defining the conceptual design for what FearNot! should be (stage 1), designing and implementing a software architecture for supporting this design (stage

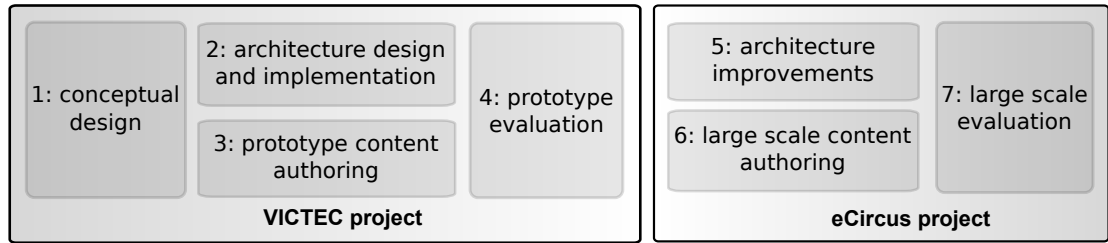


Figure 2.2: Major FearNot! development stages.

2), authoring some content for an application prototype (stage 3) and evaluating this prototype with school children (stage 4). This evaluation in which an emergent version of the FearNot! prototype was compared with a scripted one showed some promise. Specifically it was found that children found the conversations more interesting and realistic, thought the victim was more willing to take their advice and thought they were able to help the victim better in the emergent version (Aylett et al., 2005). But the evaluation also highlighted that there was much more work to be done to turn the FearNot! prototype into a useful educational tool. The prototype was simply offering too little content and too short interaction to make any discernible difference.

The eCIRCUS project’s goal was to scale up the FearNot! prototype that was available by the end of the VICTEC project and increase its robustness. These goals necessitated substantial improvements/changes to the FearNot! software architecture (stage 5). More importantly for this chapter, the increase in scale went hand in hand with the authoring of new content (stage 6) that went several times in size beyond what was created in VICTEC (stage 3). While the VICTEC prototype provided a fairly linear experience of less than 5 minutes, the goal in eCIRCUS was to allow for up to 45 minutes of interactions of a more emergent nature. The benefit of a robust large-scale application was the possibility of being able to perform a large-scale longitudinal evaluation study in actual primary schools, directly embedded in the curriculum over several weeks (stage 7). For the purpose of our discussion of authoring, the following summary of the study results shall suffice: While no statistically significant increase in coping strategy knowledge as a result of using the FearNot! software was found (Watson et al., 2010), the use of FearNot! was shown to have “a short-term effect on escaping victimization for a priori identified victims, and a short-term overall prevention effect for UK children” (Sapouna et al., 2010).

This chapter will first focus on describing the software architecture that FearNot! is based on, especially the agent architecture FATiMA. As it has little bearing on the discussions here, we will not make a distinction between features implemented in stages 2 and 5. The later sections of this chapter will then follow to describe the authoring activities during stage 6.

2.2 An Overview of the FAtiMA Architecture

Before describing the process of creating FAtiMA based IS artefacts in general and FearNot! in particular, it is necessary to give the reader an overview of the FAtiMA system itself. It is important to note that FAtiMA is not a complete IS runtime engine, but an agent architecture following the Beliefs, Desires, Intentions (BDI) paradigm (Rao et al., 1995). A FAtiMA instance simulates the thought processes of a single autonomous agent, incorporating psychological theories of the human mind, such as the OCC model of emotion (Ortony et al., 1988) or the theory of appraisal and coping by Lazarus and Folkman (1984). In order to produce interactive stories using this technology one can populate a simulation environment with several such agents, each one acting autonomously. As these autonomous agents are influenced by each other and the environment, their interactions can be interpreted by a spectator as a narrative. If the system provides means for the spectator to influence the agents, e.g. through an avatar present within the simulated environment, the spectator becomes an interactor and the simulation as a whole can be considered an IS artefact. This distributed bottom-up approach to Interactive Storytelling has been termed “Emergent Narrative” by Aylett (1999). We now describe the basic components and functionality of a single FAtiMA agent before providing more details on how several such agents and a user interact in an emergent narrative application framework like the one used in FearNot!.

2.2.1 Planning

At the heart of a FAtiMA agent resides a STRIPS (Fikes and Nilsson, 1971)-like planning system that builds plans (intentions) based on the agent’s knowledge of the current world state (beliefs) in order to achieve goals (desires). Plans are ordered sequences of actions that the agent must execute in order to achieve a goal state. To illustrate how this works in FAtiMA, consider the example of an agent that is placed in an environment with a jar containing a cookie. For the sake of simplicity we assume the agent has only a single goal *GetCookie* and can only perform 3 actions (also called planning operators): *MoveTo*(*[target]*), *Open*(*[container]*) and *Pick-From*(*[object]*,*[container]*). Note that the actions are parameterised, i.e. they take variable arguments and can be performed on different objects. FAtiMA uses square brackets to denote variables. This is one aspect in which FAtiMA’s planning domain representation differs from the STRIPS planning language standard, which is strictly propositional, i.e. does not allow any variable bindings. In FAtiMA, the world state is described through a set of logical predicates referring to entities (objects and agents). It is these entities that the variables in planning operators can be bound to. In our example the environment contains the 3 entities *agent*, *jar* and *cookie*. Both the goal

state and the initial world state are given through a set of predicates. Additionally all actions define pre-conditions (requirements in order to be able to execute the action) and effects (world state changes as a result of executing the action). Figure 2.3 shows the logical descriptions of the initial and goal world state and lists the preconditions (on top) and effects (at the bottom) for each action.

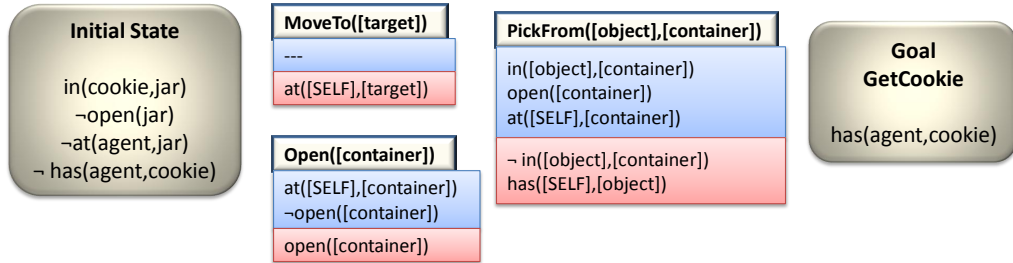


Figure 2.3: Illustration of the example cookie planning domain.

As should become evident from the examples, FATiMA uses the open world assumption, which allows it to ignore all aspects of the world state that are not made explicit. For example the goal state only contains the statement that the agent has the cookie but no other predicates, meaning that their truth value is irrelevant for this goal.

The agent's process of AI planning (Russell and Norvig, 2003) is equivalent to a search task over a state space defined by all possible actions. The planner uses backward search and first checks if the goal is already achieved. As this is not the case it proceeds to search for an operator that can achieve the goal state, i.e. it examines the effects of each operator and checks if they contain the predicate *has(agent,cookie)* or a predicate containing variables that can be substituted to achieve *has(agent,cookie)*. The only action that qualifies is *PickFrom* so an instance of this action is inserted as the final plan step with the variable *[object]* bound to the value *cookie*. The variable *[SELF]* is automatically bound to the executing agent, it merely exists, so that the executors name need not be hard-coded, so action definitions can be shared amongst several agents. Having a final plan step, the planner now checks the preconditions of the *PickFrom* action, which leads to a binding of the variable *[container]* to *jar*, as *in(cookie,jar)* is a current world state. However 2 preconditions of the action are not met, so the planner has to search the action space to find actions that can meet those preconditions. Figure 2.4 shows the shortest complete resolved plan. Note however that it is not the only valid plan. Any sequence of actions that achieves the goal state is eligible, as long as the preconditions of each action in the plan are met along the way. One example would be the sequence *MoveTo(cookie)*, *MoveTo(jar)*, *Open(jar)*, *PickFrom(jar,cookie)*.

The example also shows that FATiMA does not employ a type system for its

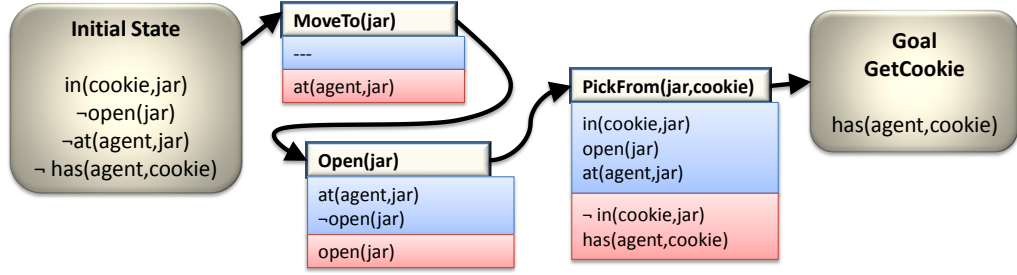


Figure 2.4: Complete plan in the example cookie domain.

variables. Any entity can be bound to any variable. A type system can however be emulated by the author through additional predicates, e.g. *isContainer*([*container*]). The actions and goals available to an agent and the predicates describing them are defined at design time and immutable at runtime. As will be shown later, defining and describing these action and goal sets constitutes a major part of the author's duty when authoring IS storyworlds using FAtiMA. Typically an agent has more than one goal. In order to help contextualize which goal should be pursued when, preconditions for goals can be defined. Only goals for which all preconditions are met will be considered by the planner. If there is more than one goal at a given time with all preconditions met, the planner will select one goal at a time to focus on. The goal that the planner focuses on and currently plans for is called the active intention. Like actions, goals can be parameterised, i.e. contain variables in their preconditions. For example, a generalized version of the goal in the cookie example could be expressed as *GetFood*([*food*]). A precondition like *isFood*([*food*]) will bind a specific entity to the variable. The variable would then be referred to again in the success conditions, which could be restated as *has*(*agent*,[*food*]).

A final point that should be made about the FAtiMA planner is that it also supports interest goals. These are different kind of goals that express constraints that the planner should attempt to work around, i.e. world states that the agent wishes to preserve. A very common interest goal for most agents would be for example self-preservation. Adding an interest goal that protects a predicate like *healthy*(*agent*) can prevent the agent from performing actions that contain an effect like *¬healthy*, by causing the planner to prefer other alternative plans.

2.2.2 Emotions

FAtiMA sets itself apart from other planning systems through its tight integration with a model of emotion. The model employed is based on the OCC taxonomy of emotions by Ortony et al. (1988). The OCC model describes emotions as valenced reactions to events experienced by the agent and helps determine which emotion a certain type of event should generate. Figure 2.5 shows the basic structure of emotions

in the OCC model.

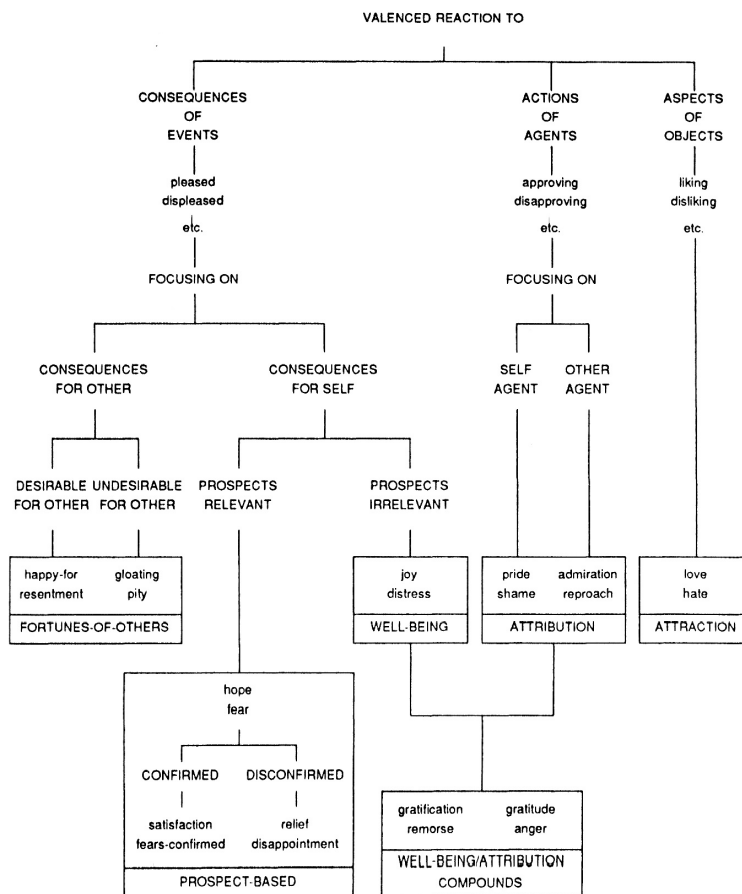


Figure 2.5: The OCC emotion taxonomy (from Ortony et al. (1988))

An emotion is represented through a label, a numeric intensity value and supporting information such as the target of the emotion and the inciting event. An agent can experience several emotions at the same time and each emotion decays over time until it eventually vanishes. The compound of all emotions felt by an agent at any given time is called its emotional state and a numeric value called mood is assigned to this too, comprised of the sum of all intensities, with negative emotions such as anger or shame contributing negative scores. That means in FAtiMA it is for example possible for an agent to have a neutral mood with a value of 0, but feel 2 strong emotions of opposing polarities, for example anger and pride.

A FAtiMA agent generates emotions in three different ways: The prospect-based emotions of the OCC model emerge naturally as a by-product of the agent's planning processes. Whenever an agent adopts a goal, hope (to achieve the goal) and fear (of failing) are generated. The ratio between hope and fear is mainly determined by the agent's assessment of the probability of achieving the goal. Once a goal succeeds, hope and fear are transformed into satisfaction and relief, whereas if it fails on the other hand they turn into disappointment and fears-confirmed. The other non prospect-

based emotions are generated as reactions to external events using a set of hard coded event appraisal rules (termed emotional reactions in FAtiMA). An emotional reaction maps an event to up to 3 values: desirability (how much does the appraising agent desire this event), desirability for other (how much does the other agent involved desire this event) and praiseworthiness (how is this event morally judged by the appraising agent). Each of these values can take negative or positive values and their combination leads to the instantiation of specific emotions in the fortune of others, attribution and well-being categories as defined by the OCC model.

Of course letting an agent possess emotions is only useful if the emotions are being used somehow by the system. One major use for emotions is showing them to the user. When a FAtiMA instance drives a graphical character in a simulated virtual world, emotions can be utilised in the visualisation process by driving expressive behaviour such as facial expressions. But while this can greatly improve character believability to a human observer, this is a purely cosmetic use of emotions. There are a number of other ways in which the emotional state also directly affects the agent's decision making processes:

- The intensity of the overall emotional state, i.e. the agent's mood influences the further generation of emotions and relationships to other agents. A positive mood, i.e. a positive emotional state will amplify new positive emotions and lessen the effect of new negative emotions, while a negative mood has the opposite effect.
- Emotions can be directly queried and used in boolean statements within for example goal pre-conditions. For example a goal `TakeRevenge([enemy])` could have a precondition that the agent's anger value towards the person to take revenge on needs to be above a certain threshold, e.g. *Greater(Anger([SELF],[enemy]),5)*.
- Prospect-based emotions influence the goal selection process. Negative emotions associated with a certain goal for example decrease the likelihood of that goal being chosen again.
- All emotions can also act as a trigger for reactive behaviour (see next section).

2.2.3 Reactive Behaviour

Not all behaviour is triggered through careful deliberation. In certain situations our impulses make us do things that we would not choose to do if we were thinking about them rationally. For example one might burst into tears in certain situations without having chosen or wanting to do so. FAtiMA provides a facility to simulate this type of behaviour as an alternative to the deliberative behaviour (Planning) described so far. The reactive behaviour of an agent is defined through a set of simple triggers

called *action tendencies*. The trigger part of an action tendency is described by an emotion and/or triggering event. The action that this trigger should cause is also explicitly stated. As with goals and actions, variables may be used to allow more generic statements. Figure 2.6 summarizes the integration of deliberative and reactive behaviour in FAtiMA and also highlights how the concepts discussed so far are plausible in terms of psychology and align with the theory of appraisal and coping (Lazarus and Folkman, 1984).

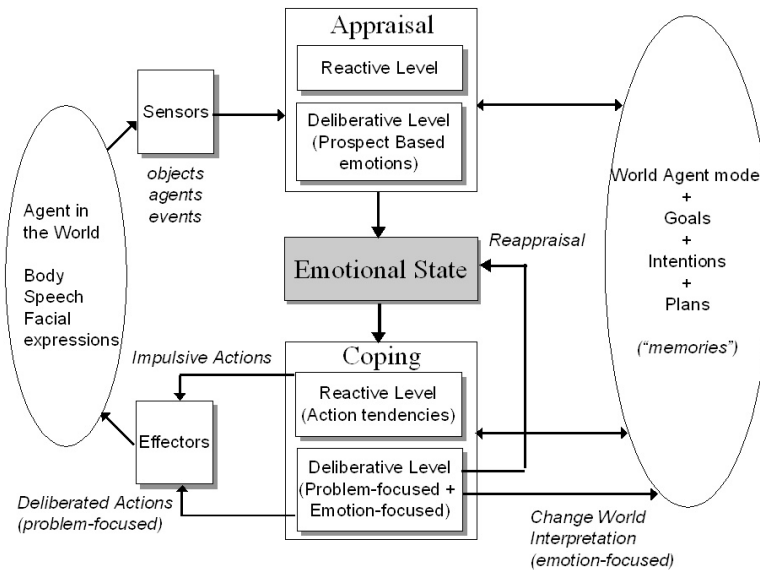


Figure 2.6: High-level FAtiMA architecture diagram (from Aylett et al. (2006a))

2.2.4 Agents and their environment

FAtiMA agents follow a sense-think-act cycle. They sense by exposing a set of well defined methods through a communication interface. When embedded in a compatible agent execution environment, it is the environment’s task to route messages regarding events in the environment to the agent. The agent processes these perceptions and updates its knowledge base. Changes in the knowledge base can trigger action tendencies, cause certain goals to become activated and others to become deactivated. Furthermore they might make the current plan invalid or alternative plans more viable. Acting for FAtiMA agents means sending the actions in the current plan to the execution environment. As the system is designed for controlling the behaviour of virtual agents in game like virtual environments, the execution of actions in the world is typically not instantaneous. Instead the duration of actions is dependent on the specific realisation of the action in the execution environment. For example FAtiMA as the agent’s “mind” might issue the command *walk to door*. By issuing this command the mind delegates control to the “body”, i.e. the part of the exe-

cution environment responsible for controlling the agent’s behaviour. In the case of *walk to door*, the body module would perform tasks such as path planning, animating and moving the character and performing obstacle avoidance. During all this time FAtiMA waits patiently for a feedback message regarding the action’s success (or failure). Only once it has received this message will it proceed with the next step in the plan. Note that the execution environment is also responsible for applying the effects of actions. While FAtiMA necessarily has internal representations of action effects these serve merely as heuristics for the agent’s planning. The agent needs to perceive the effect through its sensory inputs in order to accept it. Consider for example the action *Open([container])*. The effect *open([container])* might not be applied by the environment if the container is locked. Because effects are not guaranteed, an optional probability for each effect can be specified. This does not need to match the actual probability of the effect happening but should rather indicate the agent’s certainty that the effect will occur. The probability values are taken into account by the planner to compute overall probabilities of plans, which in turn are used to compare alternative plans.

With the realtime execution of actions and realtime decay of emotions, time represents a major source of non-determinism in any simulation that involves FAtiMA agents. This non-determinism is further amplified by the fact that usually multiple FAtiMA agents are running in parallel as concurrent, independent processes. From an author’s perspective this makes it harder to predict that a desired outcome will be produced and “debug” an agent but on the other hand it can lead to the kind of emergent behaviour that is desired from an Interactive Story.

2.2.5 Personality

One could encode all of an agent’s personality explicitly into its goals and actions. For the sake of authoring efficiency it is however often useful to formulate goals and actions in an objective way so that multiple agents that inhabit the same environment can share a single library of action and goal definitions. In order to be able to do this without creating “clones”, FAtiMA provides the facility of personality profiles. These contain:

- The already mentioned action tendencies and emotional reactions.
- The names of goals (not their definition) that are enabled for the agent and 2 numeric values per goal indicating the importance of success and failure for this goal.
- A decay and threshold parameter per OCC emotion. The decay parameter unsurprisingly controls the speed of realtime decay of the emotion. The threshold

value filters out all emotions below a certain intensity. For example if the emotional system generates a fear emotion with the intensity 4, an agent with a fear threshold of 5 would not experience this emotion, whereas an agent with a fear threshold of 3 would. In other words the fear threshold value controls how fearful a character is.

2.3 Integrating FAtiMA into an IS Runtime Engine

A FAtiMA agent is an abstract configurable real-time decision making machine inspired by human psychology principles, which needs to be integrated into a storytelling system in order to deliver an IS experience. There are several possibilities of how a storytelling system could be built around FAtiMA agents. However, for the purpose of this chapter it suffices to describe the actual storytelling system that was built in the eCIRCUS project for delivering the FearNot! IS artefact. We will refer to this system, which FAtiMA is a vital subcomponent of, as the “FearNot! software architecture”. Figure 2.7 gives an overview of its components.

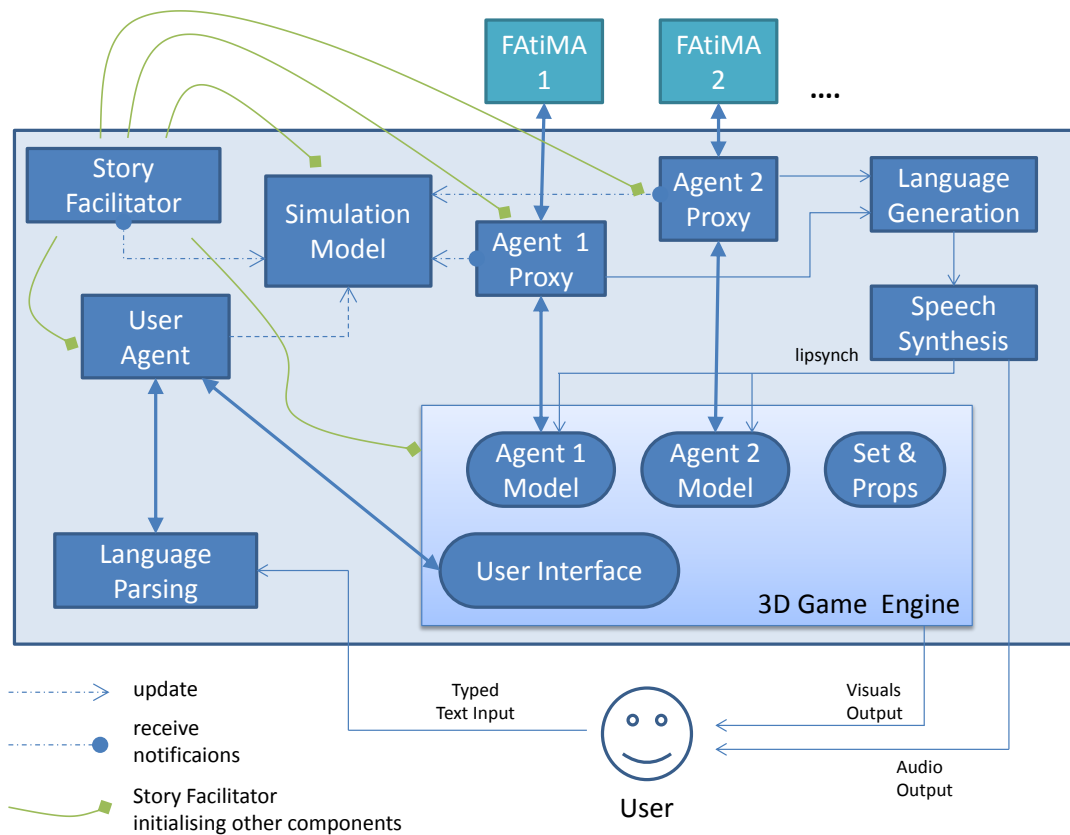


Figure 2.7: High-level components of storytelling system used in FearNot!

The system’s front-end is a Game Engine in which 3D character models can be

visualized and animated within a virtual environment. Every character displayed in the virtual environment is controlled by a separate instance of FAtiMA. Agent proxy components handle the communication with FAtiMA by implementing the sensor and actuator interfaces dictated by FAtiMA. Each agent proxy acts as a place holder for a FAtiMA agent instance in the simulation model and turns actions issued by the connected FAtiMA instance into behaviour that the virtual body of the agent can perform. Behaviours usually correspond either to a series of animations to be performed by the body or a piece of dialogue that the agent should utter. In the latter case a language generation system (a customized version of the semantic parser SPIN (Engel, 2006)) is invoked that turns the symbolic representation of the action into an actual line of dialogue. For example it might turn *Speak(Greet, Ollie)* into “Hi Ollie, how are you?”. The language engine uses pre-authored templates and has access to several context variables (e.g. gender and name of the speaker) in order to modify utterances sensibly. The generated utterance is then passed to a speech synthesizer which generates and plays back the character’s voice and links up with the 3d model of the character for lip-syncing. The user agent component maps user input events into symbolic world events. In the case of FearNot! the user interface consists of typed natural language input and thus the user agent enlists the help of a language parser that essentially performs the reverse task of the language generator described above. In fact language generator and parser use the same system in this case.

A simulation model stores the current symbolic state of the world. This consists of a set of predicates describing the properties of simulated entities plus the history of events that have occurred in the world. For each simulated entity (character or object) the world model has one container. Each container stores the predicates associated with the entity as a map of key / value pairs. For example the fact that the character John is hurt could be stored in the world model as the value of the *hurt* property inside the John container, i.e. *John.hurt = true*. The event memory of the world model is an ordered list of events that have occurred in the past with each event stored as a tuple of event type, subject, action and parameters. To give a concrete example, the event of Luke having successfully stolen John’s book could be expressed as an event with type *ACTION_FINISHED*, action *steal*, subject *Luke* and the parameters *John*, *book*. Parameters are not broken down into more specific fields (e.g. object, item, etc) at this level in order to be flexible in supporting arbitrary events initiated by FAtiMA agents. Other event types include *ACTION_STARTED* and *ACTION_FAILED*.

Agent proxies update the simulation model when they start or finish performing an action. Equally the simulation model gets updated by user actions. At the same time agent proxies are also observers of the simulation model. Every agent proxy gets notified when any changes to the model occur and passes these changes on to its

connected FAtiMA instance as perceptions. If for example FAtiMA 1 asks to perform action x , its proxy will pass this event on to the simulation model, which is being observed by Agent 2 Proxy, which in turn sends a perception event to FAtiMA 2. The simulation model does not provide or need to provide any automation by itself. It is essentially an observable black board for other components to exchange information. All actions in the system are initiated by either the agents, the user or the Story Facilitator (see next Section).

2.3.1 The Story Facilitator

The story facilitator (Figueiredo et al., 2008) serves the purpose of structuring the narrative into episodes. It essentially breaks the emergent simulation up into small manageable pieces with clearly defined boundaries. At design time the author provides the story facilitator with a story profile and several episode profiles. The story profile defines the overall properties of the story world, e.g. the characters that are part of the story, the possible locations and props, an initial world state and the set of episode profiles linked to the story. Each episode profile contains the following:

- **pre-conditions:** These are expressions of world states that need to be fulfilled in order to be able to select this episode. For example an episode dealing with the funeral of the hero would probably include a pre-condition that the hero needs to be dead in order to prevent this episode from appearing in the wrong context.
- **set, characters & props:** Where is this episode situated and which characters (and possibly objects) are initially present. This information is reminiscent of the scene introductions usually found in dramatic texts written for the stage.
- **character goals:** The episode also defines the subset of goals (out of the overall pool of goals available to each FAtiMA agent) that each character can adopt during this episode. Considering the lack of complex hierarchical goal management in FAtiMA this was considered a reasonable compromise that allows the coexistence of many goals in an agent while still allowing contextually highly relevant goals to be chosen in every situation. Of course the price one pays for this is a restriction of emergence as the agent is deprived of a large part of its possible behaviour repertoire by only having a selective subset of goals.
- **triggers:** Each episode may define trigger events (reactive rules that fire when certain world states occur). A set of special narrative actions can be invoked through those trigger (e.g. entrance and exit events of characters or external/chance events such as weather.). The intention of the triggers is to provide

the opportunity to subtly steer the story into a desired direction without interfering with the character's autonomy. For example, if the hero gets cold feet and attempts to leave the villain's lair without having confronted him, the story facilitator might use the damsel in distress trope and spawn a princess that calls out for help. This is in line with how a game master in pen & paper role playing games operates (Aylett et al., 2008).

- **finish conditions:** These define conditions in which the story facilitator considers the episode to be over. Multiple finish conditions can be defined so that multiple outcomes are supported. The default is to end the episode after a configurable amount of inactivity. This relieves the author from predicting in advance what the emergent outcomes of an episode might be. The downside to using this facility is that each episode ends with an awkward silence.

The episode selection process is straightforward. The next episode is selected randomly from the set of all eligible episodes (with all pre-conditions met) each time an episode ends. If the set is empty the story is over. In the software architecture depicted in Figure 2.7, the story facilitator is responsible for loading the respective FAtiMA agents and the user agent, instructs the game engine to load new sets and models and initializes the simulation model whenever it starts a new episode. It does a similar clean up after an episode ends. The contents of the simulation model and the states of FAtiMA agents persist in between episodes so that changes to the world and characters can be propagated across episode boundaries.

Example Scenario

The following example illustrates how the Story Facilitator (SF) works. Given a story profile with three available episodes as shown in Table 2.1 the SF would start by examining the pre-conditions of all episodes. With a blank slate at the beginning of a story, only Episode 1 has its pre-conditions fulfilled. At the beginning of the story no events have happened yet and Episodes 2 and 3 have pre-conditions checking for the presence of a prior event, whereas the pre-condition of Episode 1 checks for the absence of a prior event. As it is the only eligible one, Episode 1 is loaded and initialised. This involves instructing the 3D game engine to load the matching environment (School) and character models (Luke and John) and the startup of the FAtiMA agents for Luke and John and their proxies. The FAtiMA agents are also constrained to only have the goals specified in the episode. In this case Luke has the goal of bullying John and John has the goal of escaping from Luke. After this initialisation is complete, the SF relinquishes control to the agents and monitors the ongoing changes to the simulation model.

Episode 1	Episode 2	Episode 3
Description		
John encounters an aggressive Luke on the street	John confronts Luke and demands him to stop the bullying	John asks the user for advice on how to deal with Luke's bullying
Characters		
John, Luke	John, Luke	John, User
Setting		
School	Playground	John's Room
Pre-conditions		
!EVENT: (Luke, Bully, John)	EVENT: (Luke, Bully, John) John(courageous)	EVENT: (*, Bully, John)
Character Goals		
John: Escape(Luke) Luke: Bully(John)	John: Confront(Luke) Luke: Apologise(John) Luke: MakeFunOf(John)	John: AskForHelp(User)
Finish-conditions		
timeout	timeout	EVENT: (User, Advise, John, *)

Table 2.1: A simple example of Story Facilitator episodes

The FATiMA agents for Luke and John will now likely spring into action, probably with Luke pursuing his goal of bullying John. This may or may not succeed and John may or may not succeed in running away from Luke. The SF is not concerned with which events happen, it simply waits until the agents stop acting, since the finish-condition for the episode is a timeout. When that timeout occurs, the SF ends the current episode and looks for the next eligible episode. Assuming a bullying event has happened this makes Episode 3 eligible. Episode 2 is still not eligible as it has an additional unfulfilled pre-condition (John being courageous). The SF now initialises Episode 3 by switching the setting to John's room and activating the user and John and giving John the goal of asking the user for help. The John FATiMA agent should pursue its goal of asking for help by initiating a dialogue with the user. The user can type in different pieces of advice in response. Eventually the event (User, Advise, John, *) will have occurred, which is the episode's finish-condition and which will result in the SF ending the episode.

Depending on what advice the user has given (notice the * place holder in the Advise event), John may or may not have become courageous. In the case of him not becoming courageous, there are no more eligible episodes left to select for the SF and the story ends. If John had become courageous on the other hand, the SF would have found Episode 2 to be eligible and selected it. In the latter case the SF would initialise and monitor Episode 2, waiting once again for a timeout, after which in this simple

example the story would end as all episodes have been used up.

2.4 Story World Contents

In the conceptual division of an IS Experience shown in Figure 1.1, the FearNot! software architecture (composed of multiple instances of FAtiMA agents, a supporting agent framework and a simulated 3D virtual environment) constitutes an IS runtime engine. In order to create an IS artefact like FearNot!, a story world (i.e. content for the runtime engine to process) needs to be authored. The following types of content are required by the FearNot! software architecture:

- **Character Configuration Content:** Configuration data for FAtiMA that define an agent's personality and its repertoire of behaviour.
- **Dialogue Content:** Actual lines of dialogues, that are uttered by characters or a narrator.
- **Interaction Rules:** Rules for how the user can interact with the story world.
- **Presentational Content:** All graphical and audio material for presenting the unfolding story to the user.
- **Story Structure Content:** Configuration data for the Story Facilitator describing the constraints for story progression across episodes.

2.4.1 Character Configuration Content

Each FAtiMA instance loads at startup some XML configuration files that encode the agent's planning domain (action and goal libraries), expressed in a STRIPS like language. The planning domain represents the general knowledge of the kind of behaviour that is available in a given story world. As this knowledge usually tends to be available to all characters, the configuration files for the planning domain can be shared between all agents. Listing 2.1 gives an impression of the XML syntax used to configure FAtiMA agents, in this case to describe an action. The action illustrated is taken from the cookie planning domain (see Figure 2.3).

Listing 2.1: Example of XML syntax for describing a FAtiMA action

```
<Action name="Open([container])">
  <PreConditions>
    <Property name="at([AGENT],[container])" operator="=" value="True" />
    <Property name="[container](isOpen)" operator="=" value="False" />
  </PreConditions>
  <Effects>
```

```
<Property name="[container](isOpen)" operator="=" value="True" />
</Effects>
</Action>
```

Furthermore each FAtiMA instance is also initialized with an XML agent personality profile (see Section 2.2.5).

2.4.2 Dialogue Content

Assuming the characters in the story world are talking with each other, the system needs to be provided with the language generation templates that map FAtiMA speech acts into actual natural language dialogue lines. More than one mapping for any given speech act may be provided, in which case one of the eligible templates is chosen at random. Listing 2.2 demonstrates the syntax of specifying language templates based on two examples of different complexity.

Listing 2.2: Two exemplary language engine rules from FearNot!

```
# giving in to a threat
Type(Value:threattalktopositiveanswer)
-> Utterance(Value: "Ok, ok. Just leave me alone, ok?")

# victim replying that he/she needs help
Type(Value:helpquestionpositiveanswer) sex(Value:$S) bully(Value:$B)
-> Utterance(Value: ("Erm, yes, actually ", $B, " bullies me. ",
                    Lex(SemCat:ppn_3, Number:sg, Gender:$S, Case:nom),
                    " makes my life a misery."))
```

The value of the mandatory parameter “Type” provides the speech act symbol. The first example, shows the most simple type of template where a specific dialogue line is directly provided. The second example on the other hand makes use of parameters. One parameter in this case is a name that is inserted into the dialogue, while the other parameter is a variable indicating gender, which is passed into a lexicon lookup function to retrieve the grammatically correct personal pronoun. Similarly any world state, including the emotional state of characters may be passed to a language rule and used in a template. Obviously templates of the first type are easier to author but the flexibility provided by the use of parameterisation as demonstrated in the second example can improve the reusability of dialogue content. On the other hand authors disinclined to use these more complex features can achieve more reusability by paying more attention to how utterances are worded. For example instead of “He/she makes my life a misery.” one might write “I feel so miserable”, which in this context conveys roughly the same meaning.

2.4.3 Interaction Rules

The FearNot! software architecture supports only one type of user interaction with the story, namely typed natural language input. The same language engine that generates agent dialogues is also used for parsing user inputs. The templates are specified using the same syntax as the generation rules. Listing 2.3 shows a parsing rule used in FearNot!. In this particular example the parser looks for one of a number of verbs, followed at some point by the noun teacher. If this pattern is encountered it is matched to the speech act “suggestcopingstrategy”. This example also shows how a variable (\$C) is used to store (a portion of) the state of the dialogue: in this case which coping strategy the user has suggested in the dialogue.

Listing 2.3: An example of a user language parsing rule

```
or(go,talk,tell,ask,speak,say,explain) teacher %$C=copingstrategy()  
-> Type(value:suggestcopingstrategy) copingstrategy(Value:tellteacher)
```

2.4.4 Presentational Content

In the FearNot! software architecture the story is presented to the user in the form of a 3D virtual environment. This requires the availability of audiovisual resources such as textured 3d models of environments, character and object models, animations for the characters, voices and sound effects, etc. Many of these resources furthermore require some meta data annotation. For example 3D environments need the definition of interaction spots, walkable areas, etc. Character models similarly have interaction spots attached to their geometry and also need higher level behaviours that string together multiple animations and can be mapped to character actions initiated by a FATiMA instance. The creation of this category of content is a well understood work step in the production of video games. Well established tools (e.g. modelling software and level editors) are available to efficiently perform these tasks. The only type of the above resources that is decidedly different in FearNot! from standard video game techniques is the character voices. Rather than having a script containing all possible sentences and recording all of these, the speech synthesis approach employed by the FearNot! software architecture uses a unit selection technique that only requires the recording of some dialogue samples to capture a given voice and make it say anything. However, in contrast to universal speech synthesis, in a unit-selection system as employed in FearNot!, the quality of the generated speech improves if the recorded samples represent the overall domain well (Weiss et al., 2007). While speech synthesis cannot match the quality of a professional voice-over, there are definite advantages to its use in the production cycle of an IS artefact. Furthermore, the use of speech synthesis is necessitated by the template based language generation and the resulting

unpredictability of generated utterances.

2.4.5 Story Structure Content

FearNot! story structure content consists of configuration files for the story facilitator. Using XML syntax these files describe the properties of individual episodes (such as characters present, finishing conditions, etc) and ordering constraints between episodes. The above Section 2.3.1 on the Story Facilitator explains the details.

2.5 FearNot! Authoring Process

In the above sections we have discussed how the FearNot! software architecture works and what kinds of content is needed to create an IS experience based on this architecture. We will now focus on how this content was authored in the case of FearNot! These observations are based on a number of sources: personal experience of being part of the team that carried out this work, analysis of the created content and retrospective analysis of archival information such as project documents, email correspondence, etc. With regards to the creation stages outlined in Figure 2.2 this section will focus exclusively on Stage 6, i.e. the authoring activities during the eCIRCUS project.

2.5.1 Resources

The box representing stage 6 in Figure 2.2 is innocuously labelled “large scale content authoring” but it represents a variety of very different activities that all fall into the category of “IS Authoring”. While there are no exact man-hour breakdowns of work that can be attributed to this process we can make a reasonably well informed estimate: The e-Circus project ran for exactly 3 years employed about 15 full-time researchers from several disciplines and dedicated about half of its resources to improving the FearNot! prototype¹. Taking into account Figure 2.2, we can see that the FearNot! work carried out during eCIRCUS covered 3 main stages: technology development, authoring and evaluation. Assuming that these stages are roughly equal in required effort, we can conclude that authoring FearNot! consumed about a third of half the project resources (a sixth), which amounts to 6 months of the full project resources. Given the roughly 15 project employees this amounts to an impressive 90 man-months. While undeniably being a very rough estimate, this number nevertheless conveys the scale / order of magnitude of the work involved. And while some

¹The other half was spent developing a new IS artefact called ORIENT (Kriegel et al., 2008) that employs the same principles and base technologies as FearNOT! but for promoting inter-cultural empathy in teenagers.

team members were more responsible for authoring than others, every team member contributed in some way to this task: Authoring turned out to be a truly multidisciplinary endeavour and a focal point of integration where every team member could contribute in some way with their area of expertise. This is for example in contrast to the technology development and evaluation phases that were very much the exclusive domain of the computer scientists and psychologists.

2.5.2 Workflow

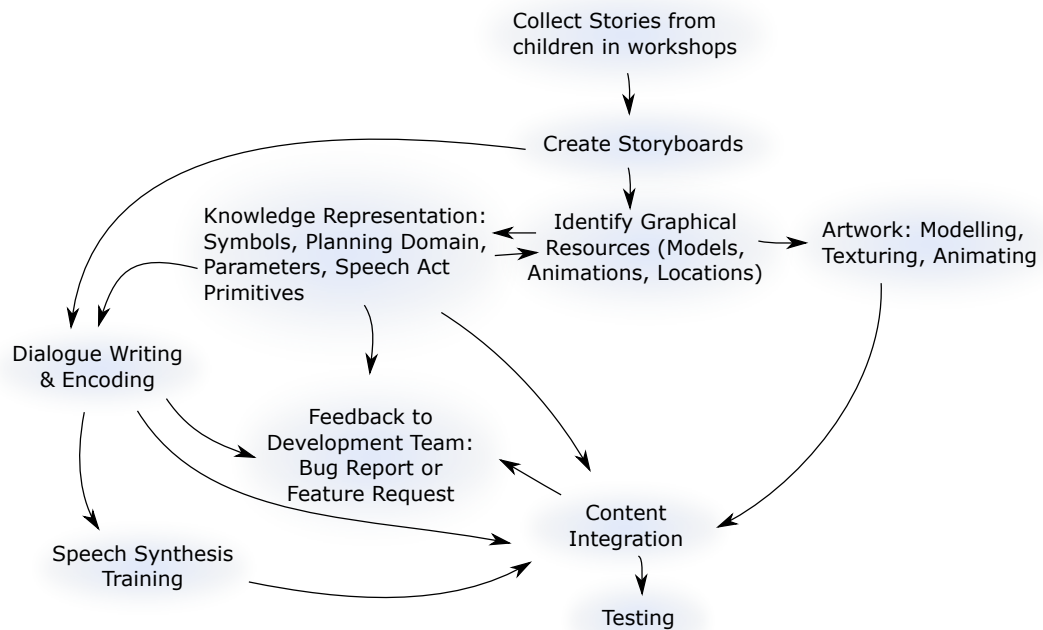


Figure 2.8: FearNot! authoring workflow

So how was the authoring work distributed between the project team members? Figure 2.8 illustrates the workflow employed by the e-Circus team. With the exception of speech synthesis training all these workflow tasks were performed manually. The process started with organizing a series of workshops in which school children were invited to write stories dealing with bullying. This kind of involvement of children in the authoring process was deemed necessary if the end result was to be a realistic portrayal of bullying that children could relate to. This stage of the process exhibits some similarities to the crowdsourcing approach to authoring that the original work presented in this thesis takes. However the children’s input stories were not used directly but rather utilised as a source of inspiration and a tool to probe children’s attitudes towards bullying. Psychologist and educational experts in the team condensed the themes emerging from the collected workshop materials into around

40 storyboards (20 for boys and 20 for girls stories). Each of these storyboards was meant to represent the blueprint of a FearNot! episode.

At this stage the team collectively analysed the available storyboards and identified a pool of required graphical resources. This included 3D locations, character and item models and animations. This step allowed the parallelisation of knowledge representation encoding (essentially configuring the parameters and planning domains for the FAtiMA agents, more on this in the next section) and artwork production. Our artists did not have to wait for primitive actions to be defined by the KR encoding team, but could immediately start producing content. The KR encoding team was still able to add graphical resources to the list as their work of translating the storyboards into autonomous agent behaviour progressed. As part of the knowledge representation encoding, a list of speech act symbols was produced. In a separate stage, language templates (see Section 2.4.2) for mapping these symbols to dialogue lines were produced. With this dialogue content in place, suitable training data for the unit selection speech synthesis component could be selected and recorded, which was then used to automatically train the unit selection speech synthesizer.

Finally authoring also encompassed an integration stage where data was created that ties these diverse pieces of content together (e.g. story facilitator episode configurations). The integrated content was then tested in order to ensure that episodes varied interactively based on prior interactions and that no possible story paths led to a dead end. Problems found in testing were corrected, followed by further retesting. Throughout the authoring process content creation and software architecture development were tightly coupled. In many cases bugs or missing essential features in the software were only uncovered during authoring.

That this is the workflow that would be adopted for IS authoring was not clear from the beginning. It emerged however quite naturally, which is perhaps not surprising, considering that it bears strong resemblances to the way video game development teams collaborate. The most interesting step in this workflow that is unique to interactive storytelling is the encoding of the knowledge representation, which we now examine more closely.

2.5.3 Knowledge Representation Encoding

During this workflow step, authors add to the configurations of the FAtiMA agents that represent the different characters. This is done by editing the various XML files that define character goals, actions, personality parameters, etc. It would be very inefficient if the author would have to wait for the content integration stage to see the newly configured FAtiMA agents in action. It became immediately obvious that the authors needed some way of testing and debugging the created knowledge representations in a direct manner. To this end a command line based simulation

runtime environment was made available to the authors that allows the immediate observation of the emerging interactions between a set of FAtiMA agents.

Initial Role Configuration

The initial stage of knowledge representation encoding identifies the required characters and their roles. A role is a simplification of character personality that makes authoring easier. Rather than treating each character as an individual, authoring is performed at the level of archetypical roles. In FearNot! these roles are bully, bully assistant, victim, bystander and helper. Dealing with roles rather than individuals immediately cuts down on the authoring effort, as there are at least 2 characters for each role (because FearNot! has separate scenarios for boys and girls). The disadvantage of course is rather shallow characters without much individual personality beyond their role requirements. Also if there are several characters with the same role profile around at the same time they tend to behave like the clones that they arguably are. In FearNot! this is for example often the case with the bully assistants, the characters forming the bully's gang. Nevertheless, the e-Circus team decided that the efficiency benefits gained by using role profiles outweighs their disadvantages. Also it would be always possible and trivially easy to later split a role profile into two or more separate character profiles and start modifying these.

After the set of roles was determined, the emotional personality parameters for each role (threshold and decay for each OCC emotion, see Section 2.2.5) were set to sensible values with the help of the team psychologists. These values could be tweaked further at a later stage if they turned out to represent the role incorrectly.

Identifying the Initial Action Set of an Episode

After the initial roles were defined and their emotional personality parameters were set, knowledge representation encoding was performed on a per episode basis. An author would be assigned an episode and its respective storyboard and be tasked with its implementation as FAtiMA agent behaviour. The e-Circus team established some guidelines on how this task should be approached by authors. In the first instance the author should analyse the provided storyboard and extract from it the contained actions / planning operators. This is not a trivial task as actions can describe behaviour at various levels of detail. Theoretically, an action could be arbitrarily microscopic (e.g. "sit down", "lift fork", etc), macroscopic (e.g. "eat dinner") or anything in between. But practically, the fact that actions need to be visualized by the FearNot! software architecture, put some tight constraints onto what an appropriate level of detail was. Getting this right requires some experience. As all agents share a single planning domain across all episodes, examining the actions that have already been

authored as examples of the right level of detail can help.

Authors are also encouraged to reuse existing actions from the planning domain wherever possible, instead of creating new ones. More experienced authors should ideally go even one step further and proactively look for similarities between a newly identified action and an existing action. If such similarities are found, it is often beneficial to refactor the existing action, i.e. make it more generic / abstract, for example by adding a parameter. A concrete example of an instance where such refactoring happened in the actual authoring process is in the abstraction of question and reply speech acts. Initially there were separate actions for different question speech acts, but when the common reasoning structure behind these was discovered, a single generic abstract question action with the concrete question as a parameter, was able to replace all these individual question actions.

Implementing Action Selection Mechanisms

At this stage the author has identified the actions contained in a particular storyboard. The next step is to implement the necessary action selection mechanisms so that a group of FAtiMA agents actually perform these actions in the right order. In other words, the agent configurations are adjusted so that the input storyboard can be produced as an output. Concretely this involves adding or modifying some of the following: goals (including goal preconditions and success conditions and goal importance parameters for certain roles), action preconditions and effects, action tendencies for reactive behaviour and emotional reactions.

Extracting the action selection mechanisms from the storyboard is more difficult than the previous step of extracting the actions. As in most common forms of narrative presentation, the storyboard does not spell out the character's decision processes. We do not always know why characters in stories do what they do. The author thus has to come up with their own plausible reasons for the character's behaviour in the storyboard. They then need to find a way of formalizing these reasons into a FAtiMA compatible action selection mechanism (e.g. a goal or a reactive action tendency). Again, this is not trivial as the exact same comments that were made above about the level of detail of actions, reuse of existing actions and refactoring of the action library equally applies to goals as well.

Extending Breadth

When the author has succeeded in recreating the storyboard via a simulation of FAtiMA agents, the most difficult work step still lies ahead of them. Simply deterministically producing a linear story using an agent simulation system is clearly not the purpose of an emergent narrative system. The author now has to consider other

possible courses of events that could arise in the situation depicted by the original storyboard (in effect producing, at least in their head, alternative storyboards) and implement these alternative plots just as they implemented the initial storyboard. In order to test these different paths the command line simulation environment allows the simulation of user interaction and the manipulation of internal variables to test their effect on the simulation outcome.

The e-Circus authoring guidelines advocated approaching this task from a character-centric perspective. Instead of imagining what other plots could happen, authors were encouraged to think about other things characters could do. The distinction between these 2 approaches is subtle but the latter one is a much closer fit to the character-centric story world encoding used in FearNot! Character-driven thinking also enables an interactive FATiMA implementation cycle discussed in Louchart et al. (2007b) and illustrated in Figure 2.9.

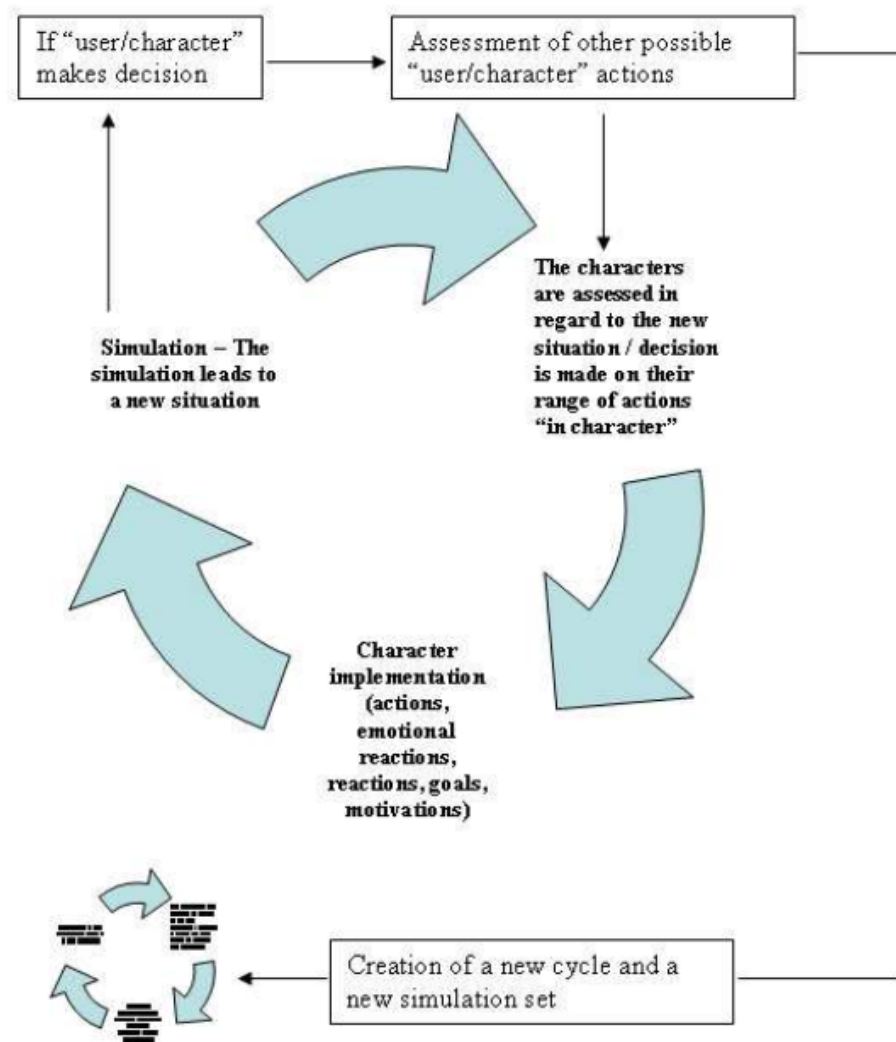


Figure 2.9: Iterative interactive implementation cycle (from Louchart et al. (2007b))

Essentially the approach requires early and constant involvement of the simula-

tion environment in the authoring process in order to obtain new situations derived from the interplay of the authored characters. This has some important advantages: Firstly it incorporates debugging into the authoring process and allows mistakes in the knowledge representation to be detected early. Secondly this approach also promotes real emergence in the sense that the system might surprise the author and arrive at a situation that was unexpected. After all the system makes decisions based on a complex interplay of many variables and factors that will not always be possible to predict for the author, especially once the planning domain has reached a certain size. Swartjes and Theune (2009) make a similar argument and use the term co-creation for such an iterative process. They advocate a mindset where authors should try to accept and incorporate the surprising situations that the simulation produces rather than discarding and preventing them.

2.6 Authoring Observations

The following observations about authoring are based on an analysis of the process described above and of its final outcome: the FearNot! story world. The problems that are highlighted in this section are by no means meant as a criticism of the authoring team, which after all the author of this thesis was a part of as well.

2.6.1 Actual Use of Planning

In Section 2.2.1, we have explained how planning sits at the heart of FAtiMA. In theory, modelling the agent's decision making processes as a planning domain is the key to characters that exhibit intelligent decision making and react to unexpected situations by specifying what the agent can do (actions) and what the agent wants (goals) without explaining how the agent reaches its goals. Unfortunately FearNot! exhibits very little of such emergent behaviour, and this is not for a lack of trying on part of the authoring team. It just turned out that modelling stories in a way that makes proper use of FAtiMA's planning abilities is very hard.

To explain the problem, we turn to an analysis of the set of active pursuit goals authored for the emergent FearNot! episodes². The complete list of these goals can be found in appendix A. In total 83 goals were authored. A manual static analysis of the planning domain revealed that for 80 out of these 83 goals there exists only a single valid, rational, partially ordered plan that can be constructed from the planning domain. This means that for only 3 goals the agent had a choice in the course of action to take in order to achieve the goal (between 2, 2 and 3 alternative plans respectively). In all other 80 cases the authors have essentially hard-coded the unique solution of

²Goals that drive the dialogue with the user are not considered here as they comprise a completely separate goal set that is only activated in interaction episodes.

how to achieve this goal into the planning domain. Clearly this is not making good use of the flexibility that a planning based architecture offers. If the agent has only a single way of achieving its goal then it would be more effective to specify this plan directly rather than having a planner assemble it expensively at runtime.

The second realization from the planning domain analysis is that the majority of goals (67 of 83) will be achieved by a plan containing only a single action. In these cases the goal's success condition directly specifies the action that has to happen to make the goal succeed and this action itself will have no preconditions that require the agent to perform another action before it. A possible explanation for the prevalence of both these phenomena (one step plans and one plan goals) is that most of the goals that were authored are expressed at a level of detail that is very close to the level of detail at which the actions are expressed. If this gap in hierarchical abstraction is very narrow then a goal is not able to effectively decompose into complex plans. As a result of very little decomposition plans are short and alternative-less. Apparently, creating goals with a wider hierarchical gap to the actions is conceptually difficult. For example the goal "ConfrontVictimTellTeacher" as the name suggests is a goal of the bully to "confront the victim after he went to the teacher for help". This is one of the many goals for which only a single plan consisting of a single action can be synthesized from the planning domain. The action in this case is a speech act verbally abusing the victim for being a "snitch". What else could the author who was tasked with implementing the action selection processes that trigger this speech act then have done, instead of authoring this primitive goal? A goal with a wider hierarchical gap would have to address the question of why the bully wants to confront the teacher and express this motivation as a goal. There are several reasons the author might consider: for example the bully might like to scare and threaten his victim or he hopes by threatening the victim he will avoid future trouble. In the first case we have a very generic goal that covers almost anything the bully ever does. As such it will become extremely complex and difficult to test and debug. It also involves reasoning about the emotions of others, which is a feature that was only recently added to FAtiMA by Dias and Paiva (2011) and not available when creating FearNot! The second suggested alternative goal (avoiding future trouble) is equally challenging to implement. In essence the agent tries to prevent a future event from happening, by acting now and taking certain preventative measures. But it is not at all obvious how a goal success condition for such an avoidance goal could be expressed.

2.6.2 Decision Making

The prevalence of goals that produce alternative-less plans begs the question how agents make decisions, i.e. choose between alternative courses of actions. In FearNot! most choices are encoded in the preconditions of goals. A recurring pattern in the

FearNot! goal library (listed in appendix A) is the creation of pairs (or larger sets) of related goals that are used to represent choices. Due to mutually exclusive preconditions, an agent can only ever have one goal of such a set active at the same time. An example of such a goal set is the pair “JoinGroupAccept” and “JoinGroupRefuse”, which is shown in listings 2.4 and 2.5. The only precondition that differs between the two goals is the property check of the like relation. Depending on the value of the like relation, one of the two goals gets activated. The planner will then resolve the respective success condition which triggers the speech action corresponding to accepting/refusing the request.

The problem with this pattern is that the decision is hardcoded to be solely dependent on the value of a specific single variable. This prevents the decision from being taken by the planner itself. For example, someone who actually likes the victim might still decide to refuse the request to join the group in fear of reprisals by the bully. A planning based agent architecture like FAtiMA would in theory allow for this kind of decision making to be modelled, but in the content authored for FearNot! this is rarely encountered. The fact that FAtiMA lacks any form of hierarchical goal management probably exacerbated this situation. The ability to express sub-goal relationships should have enabled the FearNot! authoring team to more effectively encode decision making. But even though this was discussed several times during the project, the e-Circus team unfortunately lacked the resources to implement this feature in FAtiMA.

Listing 2.4: The JoinGroupAccept goal

```
<ActivePursuitGoal name="JoinGroupAccept([joiner],[group])">
  <PreConditions>
    <Property name="[group](isGroup)" operator="=" value="True" />
    <Property name="[SELF](inGroup,[group])" operator="=" value="True" />
    <Property name="[joiner](inGroup,[group])" operator="=" value="Pending" />
    <!-- only active, if a join request was made -->
    <RecentEvent occurred="True" subject="[joiner]" target="[group]" action="Question"
      parameters="joingroupquestion" />
    <RecentEvent occurred="False" subject="[SELF]" action="Reply" target="[joiner]"
      parameters="joingroupquestion,positiveanswer"/>
    <RecentEvent occurred="False" subject="[SELF]" action="Reply" target="[joiner]"
      parameters="joingroupquestion,negativeanswer"/>
    <!-- choose pos unless I strongly dislike joiner -->
    <Property name="Like([SELF],[joiner])" operator="GreaterEqual" value="-2" />
  </PreConditions>
  <SuccessConditions>
    <RecentEvent occurred="True" subject="[SELF]" action="Reply" target="[joiner]"
      parameters="joingroupquestion,positiveanswer,[group]" />
  </SuccessConditions>
</ActivePursuitGoal>
```


Listing 2.5: The JoinGroupRefuse goal

```

<ActivePursuitGoal name="JoinGroupRefuse([joiner],[group])">
  <PreConditions>
    <Property name="[group](isGroup)" operator="=" value="True" />
    <Property name="[SELF](inGroup,[group])" operator="=" value="True" />
    <Property name="[joiner](inGroup,[group])" operator="=" value="Pending" />
    <!-- only active, if a join request was made -->
    <RecentEvent occurred="True" subject="[joiner]" target="[group]" action="Question"
      parameters="joingroupquestion" />
    <RecentEvent occurred="False" subject="[SELF]" action="Reply" target="[joiner]"
      parameters="joingroupquestion,positiveanswer"/>
    <RecentEvent occurred="False" subject="[SELF]" action="Reply" target="[joiner]"
      parameters="joingroupquestion,negativeanswer"/>
    <!-- choose neg if I strongly dislike joiner -->
    <Property name="Like([SELF],[joiner])" operator="LesserThan" value="-2" />
  </PreConditions>
  <SucessConditions>
    <RecentEvent occurred="True" subject="[SELF]" action="Reply" target="[joiner]"
      parameters="joingroupquestion,negativeanswer,[group]" />
  </SucessConditions>
</ActivePursuitGoal>

```

2.6.3 Use of Emotion Model

The FearNot! authors have not shied away from attempting to configure the agent's emotion models. It would have been easy to ignore the emotional aspects of FAtiMA as a troublesome burden and just author content in a way that makes agent behaviour independent of the state of the emotion model. But as evidenced by the number of emotional reactions authored as shown in Table 2.2, emotions were used extensively.

Role Name	# of Emotional Reactions	# of Action Tentencies
Victim	85	4
Bully	48	6
Bully Assistant	13	0
Bystander	43	0
Defender	42	0

Table 2.2: Number of Emotional Reactions and Action Tendencies authored for FearNot!

Nevertheless, a common complaint voiced by the authors was that it is difficult to assign numeric values to feelings and that there is little transparency whether one has chosen the right value or not. And in hindsight another problem related to the authoring of emotions was discovered. The fact that the act of planning also gener-

ates emotions (positive emotions whenever a goal succeeds and negative ones when it fails) places some special constraints on the authoring of goals: a goal always has to represent something that we would expect to make the agent happy. If not, the positive emotion generated by the planner distorts the realism of the model represented by the emotional state. However, authors were not sufficiently aware of this constraint and in several cases misused goals as trigger mechanisms for single actions that should clearly not be associated with positive emotions. For example, the goal “GotHitComplain” is activated when the bully hurts the victim by throwing an object at him and merely acts as a trigger for a speech act voicing the victim’s pain. But because it is authored as a goal, succeeding will absurdly actually boost the victim’s mood.

A better solution would have been to make more use of reactive behaviour in the form of action tendencies in these cases, which is a feature that was not much used as shown by Table 2.2, as authors in general preferred to trigger behaviour through the planner even when the plan as discussed above is alternative-less and only involves a single step. We could identify several reasons for this in retrospect unjustified dismissal of action tendencies. Authors preferred to employ the planner with the hope that this would lead to more reusable, generalised units of behaviour. As shown earlier, this anticipated generality did rarely materialise. The issue can also partly be attributed to training as the FAtiMA teaching examples provided to the authoring team disproportionately favoured planning, due to its complexity. As a result the FearNot! authors developed authoring patterns focussed on the planner. It was easy for them to forget about the availability of action tendencies altogether. This is similar to programmers that rarely use all features that a certain programming language has to offer but instead often develop their own style using only a subset of the language. So far we have discussed authoring effort that went towards generating emotions. These emotions were used to influence behaviour in a number of ways. The relation variables like and respect are directly derived from the emotion model. In 38 of the 83 goals discussed, either emotion variables or relation variables were used in a goal precondition. In most of these cases this particular precondition was the crucial one for decision making in a set of mutually exclusive goals as discussed in Section 2.6.2. Emotions also have a more subtle influence on planning. Emotions influence the overall mood, which in turn determines the magnitude of generated emotions including prospect based ones like Hope and Fear that are generated by the planer itself. These emotions in turn influence the behaviour of the planner, e.g. when to give up on a plan. This complex relationship is not very transparent, but it does mean that emotions add some entropy and unpredictability to the system that can be the source of emergent behaviour. Finally emotions are also routed to the presentational layer and used by it to drive facial expressions. While we did not run any studies determin-

ing the impact of facial expressions, it is quite reasonable to believe that they may contribute positively to character believability.

2.6.4 Interactivity

Every play-through of FearNot! looks slightly different. This has 2 causes: indeterministic / random behaviour and interactivity. As was explained before there are quite a few sources of random indeterminism, for example, which episode out of an eligible set is selected next, timing (e.g. how long the user takes to interact) affecting emotions which decay in real time or the language engine picking a random utterance out of a set of possible utterances mapping to a speech act. This means that even if the user interaction sessions were taken out completely, most times one would run FearNot! the story would differ a bit.

So how does user interaction in FearNot! actually influence the story and how much variety does it add to the produced story? Recall that after each multi-agent emergent episode, there is an interaction session, where the victim engages in a free form chat with the user. During this dialogue the user can possibly affect the mood of the character. Each sentence that the user types is mapped onto a speech act and for many of these speech acts the victim role will define an emotional reaction. From this an emotion is generated which will carry over to subsequent episodes and thus might impact the behaviour of the victim in these episodes. This of course can then in turn also affect the behaviour of other agents and so on. These hard to determine chains of knock on effects might be considered by some to make the system less reliable, but they are actually appreciated and valued in emergent narrative systems like FearNot! Furthermore, in each interaction session the user suggests a coping strategy to the victim or is asked to weigh in with an opinion about coping strategies that the victim suggests. In either case the interaction session ends with the victim having decided on a coping strategy, which is stored in a variable. The value of the coping strategy variable influences the episode selected next by the story facilitator and also can lead to the activation of certain special coping goals in bullying situations.

None of this user interaction has a fundamental long term impact on the story, but in FearNot! this is mainly by design. Considering its purpose as an educational tool, there was not supposed to be a way to beat the game and stop the bullying prematurely. Every user was supposed to be exposed to a variety of bullying situations and interactively explore the effect of various coping strategies. The fact that users are not able to significantly alter the course of events in FearNot! directly contradicts the agency requirement of IS (see Section 1.1.3). As predicted by the narrative paradox theory, reconciling agency with a fixed plot (in this case the educational requirement that the victim cannot escape his role) proved to be difficult. Because designing an educational tool had a higher priority for the project as a whole than designing a fully

realized IS artefact, a large potential of agency was sacrificed. It is worth noting that the work presented in this thesis does not attempt to solve such narrative paradox issues and instead focuses on the scalability of the authoring process.

2.6.5 Content Reuse and Abstraction

There are several strategies that were used by the FearNot! authors in order to reduce the amount of authored content:

- **Defining Roles instead of Characters:** As mentioned before, only 5 FAtiMA role profiles were authored, while there are 11 characters in total in FearNot! Every role profile is used by at least 2 characters. This significantly reduces duplication.
- **Reuse of Goals:** Almost all goals in FearNot! are parameterised. This allows them to be reduced in different contexts. The “JoinGroupAccept” goal from listing 2.4 for example is modelled in such a way that it applies to any situation where someone wants to join in a group activity, rather than a more specific situation like someone wanting to join a football team. This reusability is not only a theoretical possibility but is also often made use of in FearNot! “Join-GroupAccept” for example is used in several situations in FearNot! including Football (boys), Netball (girls), playground games and study groups.
- **Reuse of Actions / Speech Acts:** The above mentioned reuse of goals is possible because actions and in particular speech acts were also modelled where possible in a generic way. This however has the downside that a particular dialogue line that this speech act is resolved to, might be out of context in the story presentation layer. In the FearNot! language engine rules we can find two strategies for dealing with this problem. Wherever possible utterances are formulated in such a way that they apply in a variety of situations. This means, omitting references to any specific details and focussing on conveying the core message. These highly reusable utterances will usually be rather short. In order to avoid repetitiveness, often a variety of alternatives are defined. Where using generic language is not possible one can often encounter another pattern: Writing a set of utterances for a speech act, each of which is very context specific. The context is captured via the current episode, which is added to the language engine rule and used to select the correct utterance. See listing 2.6 below for an example of this approach.

Listing 2.6: Examples of utterance selection for a speech act based on context

```
Type(value:joingroupquestionnegativeanswer) episode(value:"B05")  
-> Utterance(value: "No, you suck at football.")
```

```
Type(value:joingroupquestionnegativeanswer) episode(value:"G06")  
-> Utterance(value: "No, we don't want you to work with us.")
```

2.7 Conclusion

This chapter has set the scene for why IS authoring is a difficult challenge. IS authors are required to define story worlds in terms of abstract data structures, while ensuring that interactions with the final IS artefact result in interesting, varied and meaningful stories. This process was illustrated by giving an insider perspective of the efforts involved in creating an IS artefact, the educational anti-bullying application FearNot!. We have explained the FearNot! software architecture, described the process employed in creating FearNot! and characterised the authored story world in terms of emerging design patterns. FearNot! is a widely cited major achievement of IS research. But it is also true that FearNot! only scratches the surface of the powerful emergent behaviour that its software architecture would in principle allow. We have shown the techniques that authors have used to translate a set of storyboards into emergent agent based simulations, but also concluded that authors were not able to make effective use of the capabilities offered by the FAtiMA architecture, especially planning. In short we have described the authoring bottleneck in one of its many guises. The next chapter will give an overview of other approaches to interactive storytelling and demonstrate the universality of the authoring bottleneck.

Chapter 3

Data Structures for Story Representation

The last chapter has described the details of one particular IS runtime engine, the FearNot! system utilising the FAtiMA affective agent architecture and an explanation of the authoring process for this system was given. The FearNot! system however only represents one of many diverse technical approaches to realizing an IS runtime engine. This chapter gives an overview of this variety of approaches. Particular emphasis is put on the data structures for story representation, i.e. the types and characteristics of the basic story building blocks contained within the IS story worlds. This aspect of storytelling engines is the most important for this thesis' discussion of authoring, as the data structures employed by a system dictate how authors need to express themselves and define what exactly it means to be an author.

Our survey of story representation is structured into four parts. We will first discuss the simplest possible approach, namely **explicit branching**, which simply means manually enumerating all possible story branches, a technique that is commonly used in video games. Academic prototypes of IS runtime engines however, typically favour a more generative approach. Generativity means that the system makes story progression decisions at runtime as a stand-in for the author. In such generative systems, two fundamentally different approaches to runtime story generation can be identified, which this chapter will discuss in turn: **Plot-centric** systems have some notion of what constitutes a good plot and employ this knowledge in order to sequence plot elements into a satisfying story, taking into account user action. Authors are primarily concerned with creating atomic plot elements and annotating them in some way that allows the system to sequence them at runtime. This is in contrast to **character-centric** systems, whose primary concern is the simulation of believable artificial character behaviour. Authors of such systems are primarily concerned with encoding the psychology of a cast of characters in a computer understandable fashion. There is an inherent conflict between these two view points. If our only concern is

generating an interesting plot then character believability might fall by the wayside, whereas a simulation of believable characters interacting with each other might not result in an interesting story. This conflict is in fact well known to non-interactive fiction writers too. It has been argued that learning to balance the needs for character and plot is one of the key skills a writer has to learn (Gerke, 2010). The need for a balance between these two extremes was also recognized by the IS community (Riedl, 2004; Si et al., 2008). Consequently, many IS systems adopt a **hybrid** approach. Such hybrid systems will be discussed in the final section of this chapter.

3.1 Explicit Specification of Branches

The earliest attempts of creating IS artefacts can be traced back to the birth of gamebooks in the 1970s and the most successful incarnation of this format, the *Choose your own adventure* books. Gamebooks present the reader with a story that is interspersed with choices. Each choice gives the reader a number of options, each of which has an associated page number. The reader chooses by turning to the indicated page, where they continue their reading. This leads to interactive stories with multiple endings. The author has of course specified all possible decisions and their consequences explicitly in advance. The authored story world can be visualized as a directed graph, which the audience directly interacts with. The electronic version of the same principle is known as Hypertext Fiction. This is a format which presents choices as hypertext links instead of page references but offers otherwise the same type of interactive narrative experience as Gamebooks. This approach to interactive storytelling is not limited to gamebooks but is also prevalent in video games.

While games offer players much agency on the surface, most of this freedom does not result in any changes to the plot. In fact in most games, story is little more than the glue that connects sections of gameplay and motivates the action but not an interactive experience by itself. Often the story is limited to some cut-scenes between levels. Costikyan (2007) coined the term “Beads on a string” to describe this type of rudimentary integration of story and gameplay. If games offer the player choices affecting the story line at all, then these choices are typically explicitly pre-authored as is the case with gamebooks. Because game companies do not like to create content for only a portion of players these choices typically never reach far. A popular way to minimize the impact of a choice are so called foldback schemes (Crawford, 2005), where a story diverges at a branching point only for the separate branches to eventually merge again into a single plot line. Another relatively cheap way to offer story choices one often encounters are dead ends. While choices seemingly affecting the plot are being offered, only one of them actually advances the plot, while the others

lead nowhere or only to death (i.e. a Game Over screen). Both these techniques are also heavily employed in gamebooks.

Some games have higher storytelling ambitions, especially in the role-playing genre. In best role playing fashion, titles such as for example *Mass Effect*, *Fallout*, *Fable* or *Knights Of The Old Republic* attempt to let players decide for themselves what kind of character they want to be and show them the consequences of their actions. All these games offer multiple endings and a variety of moral choices. Technically, however, they are still operating on a variant of explicitly authored plot graphs. However, rather than mapping story branches directly to a single choice made by the user, the selection of the branch often depends on internal game state that represents an accumulation of user actions. For example, all of the games mentioned above feature some kind of karma system. Many actions carried out by the player are classified as good or evil and consequently increase or decrease the player character's reputation and moral standing in the game world. Its value can then be used to select a story branch when the player reaches an intersection point.

Tying choices to game state variables in this way, makes the author's job slightly more challenging as it involves one more level of indirection. But even with this slight complication, explicitly authored story graphs are arguably the most straightforward way for authors to think of interactive stories. The author is in complete control and carefully crafts any story related decisions the user might be able to make and their consequences. Technically this type of interactive storytelling is trivial to realize but huge amounts of manual authoring are required for complex graphs. This can easily lead to a "combinatorial explosion" (Stern, 2008) and therefore, the general consensus in the IS community is that explicitly authored plot graphs are not the way forward to achieve fully realized IS artefacts (Crawford, 2005; Stern, 2008). Nevertheless, many of the most engaging interactive story experiences to date were realized this way. This is because explicitly specified branching story graphs offer the greatest possible level of authorial intent (Riedl and Bulitko, 2013): The author is in full control of every possible trajectory through the narrative space and can thus craft a highly engrossing experience for the audience. Wei (2011) for example analyses how explicitly authored branching is used to great effect in the critically acclaimed game *Heavy Rain*: The player can make a few key choices in each chapter that have mostly local consequences (contained within the chapter) but to a lesser degree also a few global ones (across chapters). Wei concludes that restricting oneself to a manageable amount of explicitly authored branches can still result in satisfying illusion of agency for the player if done well.

For the original work on crowdsourced authoring, presented in the later chapters of this thesis, explicit branching is considered a viable (although by no means the only) option. The CROSCAT system discussed in Chapter 7 uses an explicit branching

story representation. Enlisting potentially thousands of prospective authors working together has the potential of creating branching structures of unprecedented scale and therefore the scalability argument does not necessarily hold anymore.

3.2 Plot-centric Story Representation

In Plot-centric systems, authors associate units of story with some contextual data that allows the system to connect story units at runtime in order to achieve a satisfying plot. In the following we examine several ways in which such systems may obtain their notions about what constitutes a satisfying plot, namely planning, case-based reasoning, narrative theories and user modelling.

3.2.1 Planning

The **Mimesis** system (Young, 2001; Young et al., 2004) represents a purist planning approach to generating interactive stories in game environments. A Mimesis story world consists of a set of planning operators that describe story events and a story outcome composed of one or more goal states. Similar to FATiMA as described in Section 2.2.1, MIMESIS employs a STRIPS style representation of planning operators. Unlike FATiMA, the MIMESIS system performs planning at a global level, i.e. not just from the perspective of an individual character. The goal states that the planner aims to achieve represent the end of the story and are pre-specified by the author. Figure 3.1 shows an exemplary small MIMESIS plan. Such a universal plan representing the entire story is initially generated when a new session is launched. The plan includes actions for all non-player characters, exogenous events and the actions anticipated to be carried out by the player. An execution engine will then schedule the visualization of the plan steps (except those attributed to the player) within the game environment (e.g. as 3D animations) and monitor their progress.

It is highly unlikely that the player will behave entirely according to the initial plan. If the user performs an unanticipated action, the system checks in realtime whether this would threaten / invalidate the current plan. If such a threat is detected, Mimesis employs one of 2 mediation strategies. Accommodation, the first of these strategies, is the planner's attempt to repair the plan. E.g. if the player decides to kill a character that was due to provide a vital piece of information later, a repair to the plan might involve finding this information in the deceased character's diary. If there is no way to feasibly repair the plan so that it accommodates the users action then the intervention strategy is used as a last resort. Intervention replaces the intended action with an alternative that does not threaten the plan, a so-called failure mode (Riedl et al., 2003). Failure modes constitute a set of actions that are not

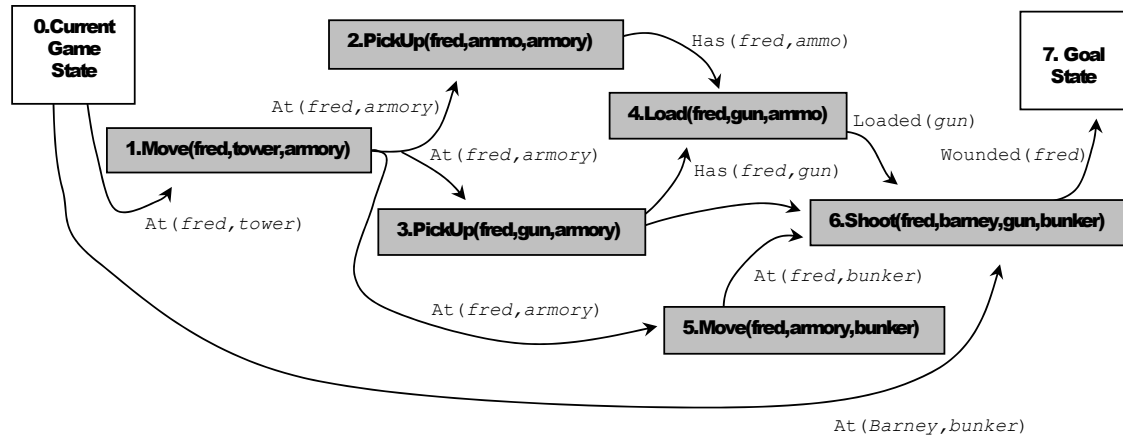


Figure 3.1: Illustration of a plan in the MIMESIS system. (from Young et al. (2004), p 9)

eligible for regular planning and that are explicitly specified as possible replacements for other regular action. For example, a possible failure mode for a shoot action is a jammed gun.

Riedl (2009) discusses author goals as an extension to Mimesis' representation of a story as a universal plan to achieve a single outcome. Author goals specify intermediate states that the story is required to pass through along the way to its ultimate outcome. This gives the author finer control over the desired story arc and effectively segments the planning problem into individual episodes, thus reducing the computational cost of planning.

From the author's perspective, writing story worlds for a plot planning based IS runtime engine like MIMESIS is mainly a matter of planning domain construction, i.e. a) the conceptual modelling of the story world as symbolic entities and their properties and b) the specification of goal states and planning operators, including their pre-conditions and effects, which reference the symbolic world model. Depending on the system, additional knowledge, e. g. failure modes or author goals may be required as well. This demands a similar understanding of planning concepts from authors as a FAtiMA based system. The fact that the planning is not distributed as in FAtiMA but centralised can be both a help and a burden. On the one hand, the author does not need to anticipate the interactions of multiple autonomous planners, which makes plans more readable but on the other hand, distributed per-character planners allow to break the planning space down into smaller individual units.

A major disadvantage of plot-level planning systems that use explicitly authored goal states like Mimesis is that endings are unchangeable. For some stories e.g. about the inescapability of fate this can be acceptable but in general it limits the author's expressive possibilities and may diminish a user's sense of agency, especially on repeat interaction sessions with the same IS artefact.

3.2.2 Case-Based Reasoning

Kolodner (1992) describes case-based reasoning (CBR) as the process of “using old experiences to understand and solve new problems”. A typical CBR system compares a current situation and problem to a library of reference cases (i.e. known solutions for previous situations), retrieves a solution for a similar situation from the case-base, adapts it to fit the current situation and finally uses it to solve the problem. This approach lends itself quite naturally to IS generation, if we consider a case-base of exemplary stories or story fragments that demonstrate the quality of a “good” plot. These example stories are then potential solutions to the central problem that all IS runtime engines aim to solve: generating a good narrative continuation for the current plot situation.

Minstrel (Turner, 1993) is an early (non-interactive) story generation system that employs CBR to model the creative thought-processes of an author. Story generation for Minstrel is the process of building a plot graph that adheres to several author-level goals that operate on a meta level and specify desired thematic, dramatic, consistency and presentation properties of the story to be generated. During its construction of the story graph, Minstrel queries its episodic memory (i.e. case-base) to find fitting story fragments and when failing to locate a direct match, uses one of its twenty-four built-in Transform Recall Adapt Methods (TRAMs) in order to transform the query. TRAMs are creativity heuristics that implement specific strategies for lateral thinking. The TRAM dictionary includes for example a TRAM for generalizing the role of an actor and one for replacing an event’s outcome with a similar one. TRAMs are recursive and if necessary a whole series of them is applied until a solution is found. Upon finding a match, each TRAM on the current query transformation stack adapts the found story fragment, effectively reversing the transformation applied to the query at each transformation stage until the completely adapted story fragment is inserted into the plot graph.

As Minstrel demonstrates, the key ingredient for a CBR based storytelling system is its case adaptation mechanism. Where Minstrel uses its dictionary of built-in creativity heuristics, Fairclough (2004) and Gervás et al. (2005) have both independently adopted a CBR approach to story generation that employs the structuralist model of narrative by Propp (1968) (see next section) for retrieving related story pieces from the case-base. Example story fragments have to be annotated with their respective Propp functions that imbue them with additional semantics.

Riedl and Sugandh (2008) describe a way to augment a story planning algorithm with the ability to reuse pre-defined story pieces (termed vignettes). Their system represents these vignettes as plan fragments, i.e. partially ordered sets of events with preconditions and effects that can be woven into the generated story plan like ordinary planning operators. Vignettes that are specified in a different source domain can

be transferred to the current target using the Connectionist Analogy Builder (CAB) model (Larkey and Love, 2003). Their analogy-based case transformation establishes similarities between operator structure and associated domain state in the source and target domains and exhibits some similarities to Minstrel’s Cross-Domain-Solution TRAM.

From the authoring perspective, CBR requires the creation of (pieces of) example stories. These have to be specified in a formal structure (e.g. schemas in the case of Minstrel, the vignette’s plan fragment representation) so that they can be adapted to a different context by the system. None of the systems above handles example stories expressed in a human-friendly format like natural language. Case-based reasoning alone therefore does not make the authoring process less technical but it does make it more declarative as opposed to e.g. authoring exclusively at the level of planning operators. This increases authorial intent, as authors can demonstrate to the system via examples, what kind of stories they prefer.

The biggest potential advantage of using CBR however is that case libraries may be shared across multiple story worlds. On the one hand this would diminish authorial intent as the system now makes decisions based on a large library of cases that the author has no control over but on the other hand it would reduce authoring effort immensely. This is the type of usage that Riedl and Sugandh (2008) imagine for their vignette based system. However this would require large general purpose reusable case libraries, which so far do not exist.

3.2.3 Narrative Theories

For a long time, literary scholars have tried to build models of the defining characteristics of stories, going all the way back to Aristotle (330 BC) and his identification of a narrative arc based on three parts. Aristotle’s model was further refined by Freytag (1863), who proposed a five-partite dramatic structure (see Figure 3.2). Polti (1921) identified 36 common narrative situations while Thompson (1955) and Propp (1968) both created catalogues identifying the basic structural elements common across a wide range of folk tales. Campbell (1972) established the monomyth model which describes the typical structure of a hero’s journey (see Figure 3.2) and which has been successfully applied to characterize e.g. the plots of movies like *Star Wars* and *The Matrix*.

All these models derive from a literary analysis of narratives but can be employed as a guiding principle for arranging story units in a plot-based story generation system. Instead of sequencing events purely based on causality, events may be chosen that make the generated story adhere to a given narrative theory. Several existing systems have chosen this approach:

The Faade system (Mateas and Stern, 2003) arranges its content in small dramatic

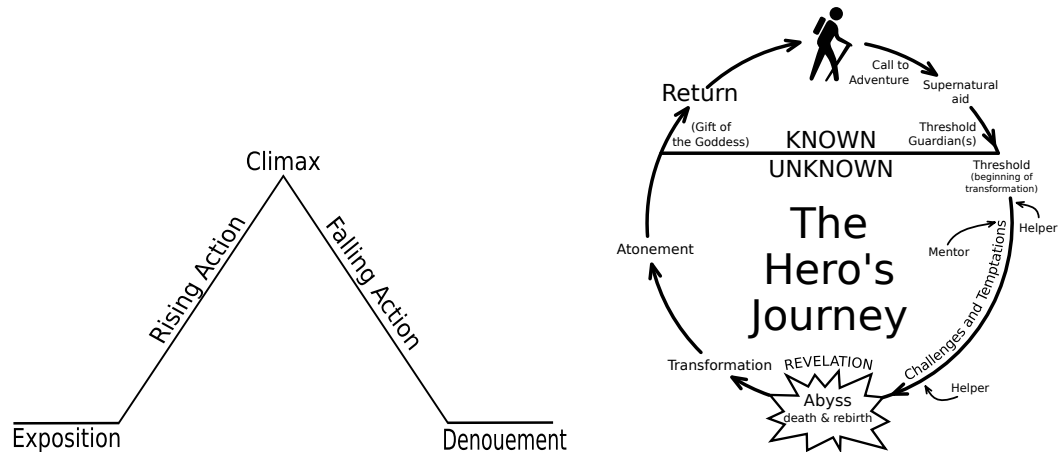


Figure 3.2: Examples of influential narrative theories. Left: Freytag's dramatic pyramid, Right: Campbell's monomyth (source: Wikipedia)

units called beats. Its drama manager tries to sequence beats according to an Aristotelian story tension value arc (see Figure 3.3). Grasbon and Braun (2001), Fairclough (2004) and Gervás et al. (2005) have all built IS Runtime Engines that structure their stories according to Propp's morphological functions. The PASSAGE system (Thue et al., 2007) employs Campbell's monomyth model for selecting encounters in a fantasy quest scenario. The GADIN system (Barber and Kudenko, 2007a) is based on a view of narrative as conflict. It contains a library of dilemmas (conflicting goals) and uses a planner that tries to incorporate dilemmas into a never-ending soap opera style narrative.

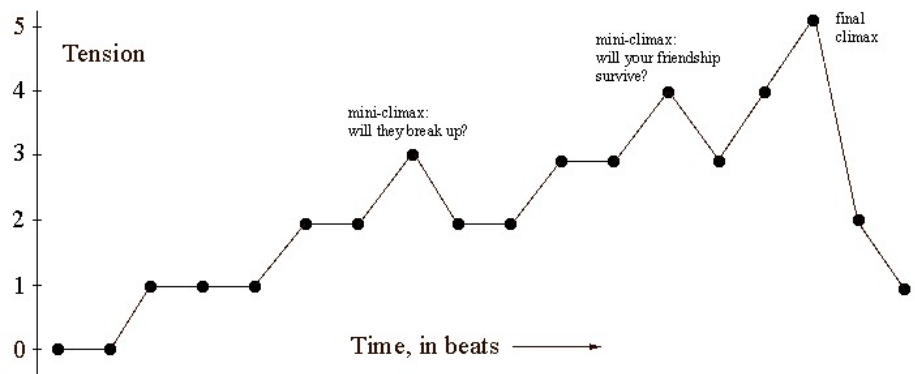


Figure 3.3: An example dramatic arc generated by Façade (from Mateas and Stern (2003), p 15)

The IDtension system (Szilas, 2003) is based on the theories of Bremond (1974) and Todorov (1970). They inspired an ontology of basic actions that are hard coded into the system (e.g. Inform, Incite, Disuade, Perform, Refuse, Renounce, Congratulate or Condemn) and informed the rules that encode IDtension's narrative logic. IDtension's event representation allows combining authored actions to form more complex

statements such as $\text{Inform}(Y, X, \text{CAN}(X, a))$ (Szilas, 2002). This increases the space of possible events without the need to author many actions.

In all the systems mentioned above, the particular choice of narrative theory to some degree influences the representation of story world knowledge. The system needs a way to associate authored story world content to its narrative model, e.g the knowledge which Propp function or stage in the monomyth model the content represents or a dramatic tension value is associated with it. The greater concern for authors is however not how a narrative theory adopted by an IS system impacts the knowledge representation but rather how it restricts the types of stories that the system is able to generate. Tomaszewski and Binsted (2007) for example discuss the inherent limitations of using the Propp model.

3.2.4 User Models

The end users of IS Artefacts may have differing opinions about what constitutes a satisfying story experience. Therefore several systems employ models of learned user preferences to guide plot generation. PASSAGE (Thue et al., 2007) incorporates its story into a RPG video game experience and characterizes the player's preferred style of playing as a vector of weights for the following player types: Fighter, Method-Actor, Storyteller, Tactician and Power Gamer. The story is structured into encounters, which are organized and sequenced according to Campbell's monomyth model (see previous section). Encounters are essentially explicitly authored branching plot graphs. Player decisions are annotated with player type weights. Thus, through every decision the player makes, the player model is updated and refined. For example if in a dialogue the player shows an interest in going on a bounty hunt, then this increases the Power Gamer dimension in the player model. The continually updated player model is used a) along with the monomyth model for encounter selection (every encounter is annotated with player model weights) and b) to automatically select certain branches within encounters

The dilemma-based GADIN system maintains a user model based on the following dimensions (Barber and Kudenko, 2007b): honesty, faithfulness, responsibility for actions, selfishness, preference for relationship or friendship, strength of character and morality. These values attempt to be an estimation of the personality of the character that the user is impersonating. The model gets updated based on how the user chooses to deal with dilemma situations and it effects the selection of future dilemmas. Of course, for this mechanism to work, dilemmas stored in the system need to be annotated with user model values.

The IS system described by Seif El-Nasr (2007) is architecturally inspired by Façade (Mateas and Stern, 2003) but adds a user model, which like the one in GADIN, intends to model the personality of the player character. Explicit rules added by the

author specify how the user model gets updated and how it is used to influence the story.

User modelling is also an important component of the IDtension system (Szilas, 2003). IDtension’s narrative sequencer module ranks, filters and chooses actions that the narrative logic component has selected as eligible according to 8 criteria (termed needs) such as Consistency, Conflict, Demonstrativeness, etc. When scoring actions, a model of the user’s current 8 needs is taken into account and actions are promoted that satisfy this model. By modelling the user’s narrative needs, IDtension aims to characterize the user on a meta level rather than as a story entity as GADIN and OPIATE do. In this regard it shares similarities with PASSAGE’s user model concept but is less video game centric.

In conclusion, there are several ways in which an IS system can model the user and several exemplary implementations of how such a model can be used to adapt a story at runtime to the demands of the current user. By definition, the user model is a data structure whose contents are updated and populated dynamically at runtime and thus not part of the story world that the author needs to create. However, typically authors need to enhance the story world content with user model annotations (e.g. annotating encounters with player type weights in PASSAGE), so that the runtime system knows how to update and make use of the user model.

3.3 Character-centric Story Representation

Character-centric (i.e. emergent narrative) approaches to Interactive Storytelling are more concerned with encoding IS story worlds in terms of character behaviours and motivations instead of story-units and story-sequencing heuristics. Common implementation characteristics of such systems are their use of psychological simulation, a distributed nature (often in the form of a multi agent system) and an insistence on autonomy of character. In the following we review some of the ways in which character-centric IS systems have been realized.

3.3.1 Agents Using STRIPS-like Planning

The previous chapter has already presented a detailed example of a system that employs STRIPS like planning to simulate the decision making processes of a character in the autonomous agent architecture FAtiMA and its use in building the IS artefact FearNot!, which needs no further explanation here. However, since the conclusion of development work on FearNot!, several features have been added to the FAtiMA agent architecture, which are deserve mentioning here, as they also have an impact on the authoring. None of these extensions relate to the planning directly but they do

serve as further examples of how a planner might be augmented with psychological theories. Lim et al. (2008) integrated the PSI model (Doerner, 2005) into FAtiMA, which describes a set of basic human needs and how they drive behaviour. Integrated into FAtiMA, this model requires the author to annotate actions with the needs that they satisfy and drain (e.g. sleeping increases the agent’s energy level, while running decreases it). By doing this, the author gains improved goal management and is no longer required to explicitly specify goal importance values, as goals can now be chosen based on the agent’s current needs. Mascarenhas et al. (2009) incorporated a model of culture based on the 5 cultural dimensions by Hofstede et al. (1997). To make use of this model, the author has to specify a cultural profile that consists of rituals and norms. The author can then assign the agent to be a member of a particular culture, which makes the agent inherit attributes from their culture. Finally, Dias and Paiva (2011) have added a theory of mind component which allows agents to reason about other agents emotions. In order to make use of this component FAtiMA authors have to use a new type of preconditions.

The Virtual Storyteller system (Theune et al., 2003; Swartjes and Theune, 2008) shares many similarities with the FearNot! architecture, amongst them the emergent narrative approach, implementation as a multi-agent system, using STRIPS-like planning and the incorporation of the OCC emotional model. In addition, it incorporates novel distributed drama management features, which will be discussed later in this chapter.

3.3.2 Agents Using Hierarchical Task Networks

The IStorytelling system by Cavazza et al. (2002) is a prominent IS runtime engine that uses Hierarchical Task Networks (HTNs) to represent character behaviour. HTNs are a representation for explicitly declaring how a goal decomposes into levels of subgoals and ultimately primitive actions. Figure 3.4 shows an example of an HTN used in the IStorytelling system’s main story world, an adaptation of the sitcom *Friends*. In this case the HTN for Ross’ goal to ask Rachel out is shown. At any level a decomposition may either use disjunctions (only one child task needs to be completed) or conjunctions (all child tasks need to be fulfilled). The latter is visualized in Figure 3.4 using an arc with an arrow. Each task node is also associated with pre- and post-conditions (not visualized). The IStorytelling system uses individual HTNs for each character, each representing a character’s entire goal and plan space. As in the FearNot! system, the system uses separate planners for each agent that run in parallel, giving the characters a strong autonomy. At runtime each planner starts decomposing its character’s goal in a top down depth-first fashion, checking the pre-conditions of every node along the way. While this is also called planning, the process is much less computationally intense than STRIPS like planning as the

causality links are already established at design time. The IStorytelling system also allows annotating the HTN with personality values that serve as heuristics for the planner when choosing subgoals. E.g. an angry character may be more likely to choose a sub-goal that is tagged as “rude” than a happy one.

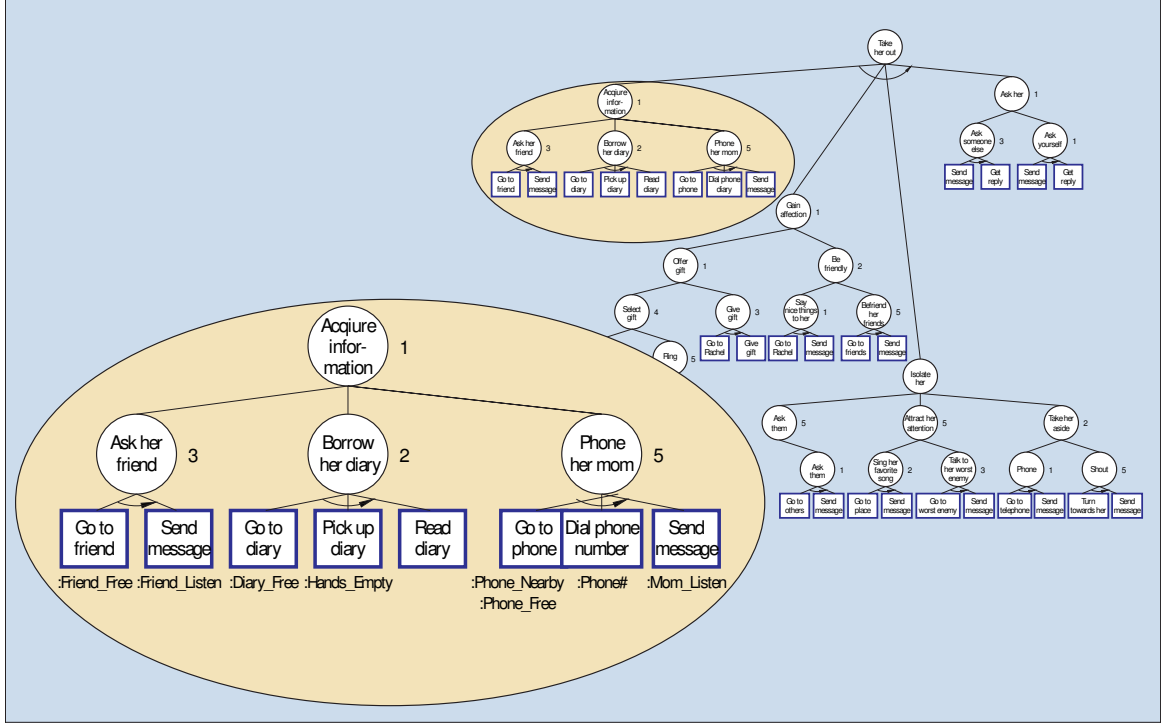


Figure 3.4: Example of an HTN used by the IStorytelling System (from Cavazza et al. (2002), p 19)

The use of HTNs to encode character behaviour has both advantages and disadvantages from an authoring point of view. It is arguably an intuitive data structure that lends itself well to debugging and visual editing. However, having to explicitly spell out every single possible causal link is not an approach that scales well. Cavazza and his colleagues have acknowledged this limitation and have later turned to Heuristic Search Planning (HSP), which is described in the next section.

3.3.3 Agents Using Heuristic Search Planning

Heuristic Search Planning (HSP) is a term describing a family of planners that automatically derive heuristics from a STRIPS encoded domain and use these to guide the search through plan space (Bonet and Geffner, 2001). This means, HSP is primarily an implementation technique for STRIPS style planners that does not significantly impact the knowledge representation compared to the other STRIPS based systems discussed earlier.

HSP has been first used by Cavazza et al. (2003) for planning humorous agent behaviour in an interactive story setting (demonstrated through an example *Pink Pan-*

ther themed story world). Humorous agent behaviour is achieved by allowing authors to add plans with the intentional potential for (comical) failure to the domain. To support this, the standard STRIPS operator description using preconditions and effects was augmented with so-called executability conditions. These conditions have to be met for an action to succeed but are not taken into account while planning. For example the presence of running water could be an executability condition for the action of taking a shower. Starting to take a shower when there is no water can be comical.

The same team later applied HSP to the EmoEmma system (Pizzi et al., 2007) an IS system built to run a story world adaptation of the 19th century novel Madame Bovary by Gustave Flaubert. Standard STRIPS is used to encode the planning domain for each character. The Madame Bovary story world was encoded with operators that not only represent character actions but also character emotion changes (interpretation operators). Figure 3.5 shows an example of an interpretation operator whose effects are only internal emotion changes of the protagonist Emma Bovary, namely increased pride and a decrease in affection for her husband. The use of interpretation operators is an interesting authoring decision but it is not a representation enforced by the engine or HSPs in general.

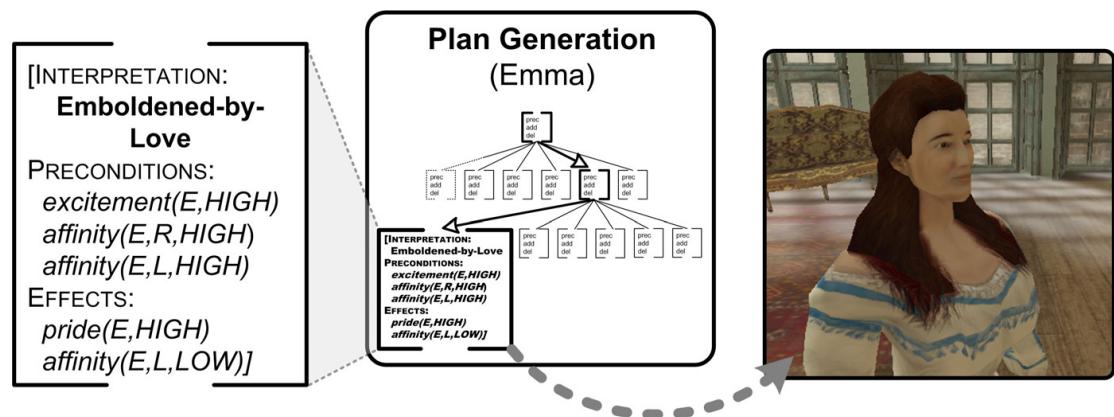


Figure 3.5: Example of an interpretation operator in the EmoEmma system (from Pizzi et al. (2007), p 4)

There is however one possible difference between HSP and other more simple planning algorithms such as forward or backward search through plan space (as for example employed by FearNot!) that also affects authoring. In realtime applications such as IS, the utilization of a heuristic function enables searching at a limited depth without necessarily reaching the goal state. The option to search at a limited depth is not a viable option for traditional STRIPS planners as it is difficult to judge the quality of a partial plan without having seen it's role in connecting the start to the

goal states. In HSP on the other hand, judging the quality of a partial plan is very closely related to what the heuristic function is supposed to do (i.e. ascribe a distance to the goal state to each possible world state).

In the EmoEmma system the planner operates at a search depth of 1 without looking ahead. Whenever it is invoked, the planner would consider all possible actions that could be taken in the current state and evaluate them based on the heuristic, choosing the highest scoring candidate for execution, without necessarily knowing for certain whether this action is really contributing to achieving the goal state. This allows systems like EmoEmma to operate on a comparatively large planning domain while retaining planning performance and thus enables far more complex plans (i.e. plans involving many steps). Authors for such a system need to be aware that due to the incremental nature of the planning, the system may not necessarily execute a plan that can succeed at runtime and design their story world accordingly. Depending on the quality of the heuristic, there is a potential for characters to take irrational decisions. It should be noted however, that this only applies if search depth is intentionally limited. One may of course also utilize HSPs to find complete plans in which case the above considerations do not apply.

3.3.4 Decision Theoretic Agents

Unlike the HSP agents discussed above, Decision Theoretic Agents differ significantly from the STRIPS style agents we have discussed so far. (Russell and Norvig, 2003, p. 466) explain “the primary difference is that the decision-theoretic agent’s knowledge of the current state is uncertain; the agent’s belief state is a representation of the probabilities of all possible actual states of the world”.

The Thespian system (Si et al., 2006; Si and Marsella, 2010) demonstrates an implementation of this approach in an IS runtime engine. Thespian is a multi-agent system of decision-theoretic agents built on top of the PsychSim framework (Marsella et al., 2004; Pynadath and Marsella, 2005). Each Thespian agent has an internal state describing its actual current physical and social status and a recursive belief state about themselves and other agents that constitutes a theory of mind. E.g. an agent A might believe with some level of certainty that some other agent B believes that A is hungry. Each agent possesses a number of competing goals, whose relative importance is decided by the author. Besides domain-specific author defined goals, all agents inherit a set of built-in social norms goals that control the agents behaviour in conversations (e.g. turn-taking, responding to questions). Decision making constitutes a look ahead search of available actions, also taking into account the mental models of other agents. The action that promises the greatest reward (as defined by the agent’s goals) is chosen.

Authoring Thespian agents largely consists of describing actions and goals. Unlike

STRIPS though, their states are described using probability values. A challenge for the author is to tweak these values in a way that the agent behaves in a desired way. The same is true for balancing the goal priorities. A fitting mechanism to aid the author in this task was integrated into Thespian (Si et al., 2005) and will be described in the next chapter.

3.3.5 Other Approaches

Storytron

There exist other less obviously classifiable character-centric approaches. One of them is Chris Crawford’s Storytron Crawford (2005, 2007). Everything in Storytron revolves around the concept of verbs. Stories simulated by Storytron consists of sentences in a formal story grammar, each of which has a single main verb. Verbs have typed slots that when filled result in an instantiated sentence. Decision making is encoded in a special purpose graphical scripting language (SAPPHO) on a per-character basis through so called inclination scripts. These encode under which conditions actors (Storytron’s terminology for characters) prefer which verbs. In a blog posting, Crawford (2012a) estimates that writing these inclination scripts consumes at least 50 percent of the author’s total effort. Besides inclination scripts the main components of a story world are verb definitions (some generic ones are provided by the system, while further ones can be added by the author), attributes, actors, props and stages. SAPPHO scripting is heavily used throughout the definition for all of the above. As the scripting language is graphical, the authoring experience in Storytron cannot be considered in isolation from the system’s authoring tool SWAT (see Figure 3.6).

Of all the systems we have encountered in this section, authoring with Storytron bears the closest resemblance to actual programming due to its heavy dependence on scripting. This makes Storytron a complex system to master for authors (Crawford, 2012b) but was an intentional design choice, necessitated by the desire to not rely overly much on fixed story generation algorithms and instead give the author full artistic control over decision making processes. As Crawford (1999) states (talking about Erasmatron, Storytron’s predecessor):

“Therefore I claim that interactive storytelling will always be the domain of the artist. Surely computers will play an important role in the creative process (and a necessary role in the performance), but the creative responsibility must lie with the artist, not the algorithm. This does not preclude the possibility of the artist creating the algorithms; indeed, the Erasmatron requires the artist to supply a great many simple algorithms for character choices. But these are narrowly-applied algorithms; overall creative strategy remains a task for neurons, not transistors.”

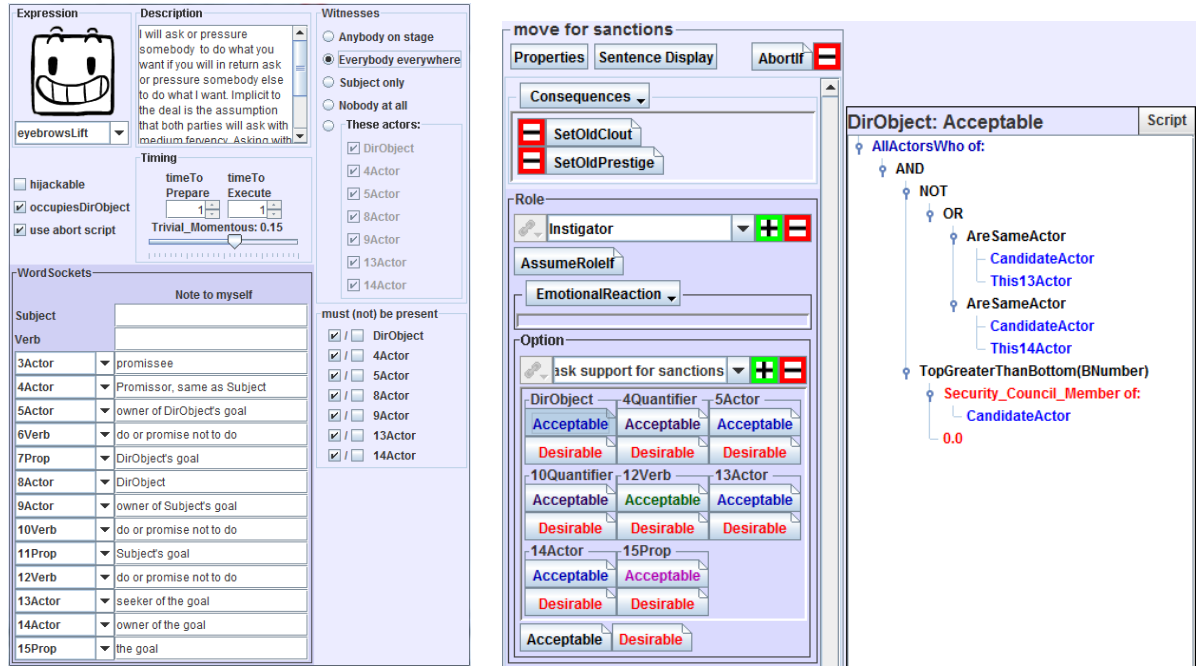


Figure 3.6: Screenshots of SWAT. Left: a verb definition window. Right: a typical SAPHO script, in this case for selecting eligible actors for a particular verb slot.

Certainly, when considering the IS systems reviewed in this chapter, we have seen many examples of IS systems limiting the types of stories that are possible to implement within them, due to their adoption of a particular algorithmic model of story generation. Storytron affords the author with more freedom and creates less such restrictions.

Comme-Il-Faut

Another noteworthy character-based system is Comme-il-faut (McCoy et al., 2010b, 2014). It centers around the idea of social games, formal models of the typical social exchanges that occur between people, which change the social landscape of their environment somehow (e.g. flirting, paying a compliment, gossiping). The example IS artefact built with the Comme-il-faut system is called “Prom Week”, is implemented as a single player Facebook game and puts the player in the shoes of a character having to navigate the complex social networks of teenage high school life. Players are assigned goals such as for example getting the class geek to date the prom queen.

Figure 3.7 shows the elements that constitute a Comme-il-faut story world such as “Prom Week”. We give a very brief description of these here. Detailed information is available from McCoy et al. (2010a).

- **Relationships:** Describe reciprocal state between 2 characters. Prom Week uses the 3 relationships *Dating*, *Friends* and *Enemies*.

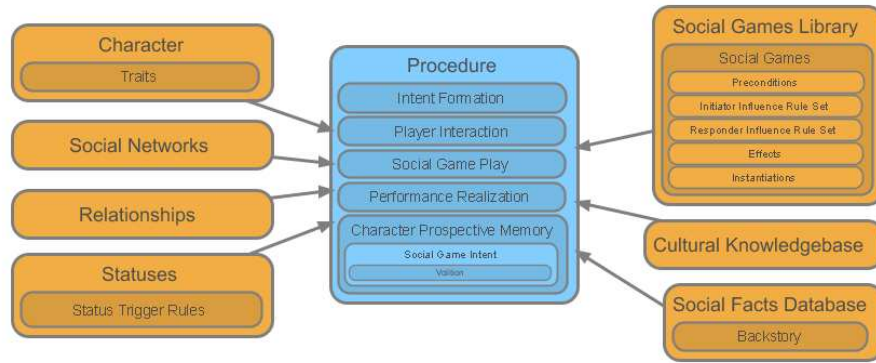


Figure 3.7: High-Level Overview of Comme-il-faut story world components (from McCoy et al. (2010a))

- **Social Networks:** Very similar to relationships but representing less stable relations between characters. Realized as a bidirectional graph, connecting all characters with each other. Prom Week uses the 3 social networks *Buddy*, *Cool* and *Romance*. Unlike relationships, which are boolean, social networks are scalar valued, i.e. able to express varying degrees of a relation.
- **Characters:** Definitions of the characters inhabiting the world. Each character can have additional author-defined traits (e.g. *Confident* is one of several traits in Prom Week). Characters can lose or gain traits through their participation in social games.
- **Character Statuses:** Can be interpreted as temporary one-directional relationships. The author can provide triggers (rules) that cause status changes.
- **Cultural Knowledgebase:** Contains knowledge about the shared culture or zeitgeist in the simulated environment. Entries in this knowledge base are topics of cultural relevance. Each topic is assigned general attributes, representing the consensus about these topics, but additionally characters can have individual relationships to them. In fact, this is a large part of what constitutes character's individuality. For example the topic Skateboarding might be considered cool in general but a specific character may find it boring.
- **Social Games:** The basic units of story progression. Each social game relates to exactly one Relationship or Social Network and has the goal of improving or diminishing it. Furthermore, each social game has preconditions that define whether the game is eligible to be played at this moment or not, a rule set for the initiator that encodes how desirable it is for them to play the particular game, a rule set for the respondent that determines the response, effects that describe how the world changes as a result of this game and instantiations, which

describe staging directions including dialogue on how to present and visualise the game.

- **Social Facts Database:** The system's memory of all social games played in the past. It is automatically filled during runtime but may be pre-populated by an author to provide a backstory.

Versu

Versu (Evans and Short, 2014) is the most recent system covered by this literature review. At the time of writing two story artefacts running on Versu have been produced and released for iOS devices on Apple's app store to favourable reviews from both users and the gaming press. This foray into the mainstream alone makes Versu a noteworthy system that in due time may help popularize IS, but it also uses a unique and interesting story representation. Versu artefacts are text-based simulations with which the user interacts by choosing from a list of actions presented by the system. The user can choose when to act and when not to act. Unlike other IS systems the role played by the user is not predetermined by the author (i.e. the user can choose to control any character) and neither are the actions available to the user at any given time during the simulation. The eligible repertoire of actions stems from the currently active social practises, which sit at the heart of the Versu system.

A social practise is a structure which describes a social situation such as having a meal. Several social practises may be active at the same time, e.g. the process of having a meal itself, a conversation occurring at the dinner table and an ongoing flirtation between two of the dinner guests. Social practises are associated with a set of actions that are appropriate for that particular situations. Characters in the Versu system are implemented as utility-based agents and limited to choose from the actions which are enabled by any of the currently active social practises. Among these actions however, agents choose autonomously. Their internal utility-based short term planner will select the action that best satisfies the agent's desires.

Listing 3.1: Example of a social practise in Versu (from Evans and Short (2014))

```
process.greet.X(agent).Y(agent)
  action "Greet"
  preconditions
    // They must be co-located
    X.in!L and Y.in!L
  postconditions
    text "[X] says 'Hi' to [Y obj]"
end
```

Listing 3.1 shows a very simple example of how a social practise is encoded in Versu. In this case the social practise of one agent X greeting another agent Y makes

the "Greet" action available to X if the precondition is fulfilled that both agents are situated in the same room. A social practise is activated by adding a sentence that instantiates it to the system's knowledge base, e.g. `process.greet.jack.jill`. Many other elements are involved in the Versu architecture, whose description would be outside the scope of this literature review, but the idea of having social practises provide affordances to a set of autonomous agents is the central distinguishing element of the system.

3.4 Hybrid Solutions

While most IS systems as discussed above are clearly primarily either character or plot-centric, many aim to incorporate some aspects of the opposing approach as well. Several primarily plot-centric systems have found ways of representing the motivations and behaviour for believable characters and many character-centric drama management have been augmented with some drama management functionality that aims to improve the conditions that are necessary for dramatic plots to occur without restricting character autonomy. From the author's standpoint, the hybrid nature of these systems adds duality to the story world knowledge representation. The author will be required to provide both plot- and character-centric data structures to make effective use of these systems.

3.4.1 Character Autonomy For Plot-Centric Systems

Systems that center on plot-centric story representation may run into the risk of creating stories with unbelievable characters. This may occur when character actions are selected to serve a plot goal without considering whether the character is motivated to perform this action. Swartjes (2010) gives the following example to illustrate this potential problem:

"The goal of the planner may be to create a plot in which a beggar becomes rich. Without considering character motivation, a plan that satisfies this plot goal might be that the beggar goes to the bank, the bank owner gives the beggar all of the bank's money, and the beggar is rich. This creates a believability problem upon execution of the plan: there is, for instance, no believable reason for the bank owner to simply give away the money."

Of course, character motivations may be encoded implicitly in plot-centric data structures. E.g. in the bank director case one might add preconditions to the give-money action that explicitly rule out states, in which this action cannot apply. Applying such an implicit representation of character motivation can however quickly

lead to confusing and bloated story worlds, e.g. if the give-money action is to be reused in different contexts, where different preconditions apply. Riedl and Stern (2006) address this problem by introducing a system that treats the high-level plot points planned by a plot level planner as prescriptive and proscriptive directions given at runtime to autonomous character agents. Prescriptive directions assign a goal to the character. The autonomous agent may achieve this goal either together with its current autonomous goal, after completing its current autonomous goal or if necessary after first believably abandoning its current autonomous goal (e.g. the character gets a phone call with a new order). Proscriptive directions give the autonomous character a list of world states to avoid in its own planning.

Façade also uses a division between high level plot-planning and low-level character autonomy (Mateas and Stern, 2005). As mentioned before Façade’s plot is structured in small dramatic units called beats. Each beat is in fact a collection of autonomous character behaviour scripts expressed in a language called ABL (Agent Behaviour Language). Arinbjarnar and Kudenko (2008) use a similar approach entitled DED (Directed Emergent Drama). Their system does have a central drama Manager which selects episodic drama structures called schemas. These schemas define a set of roles and for each role a set of actions annotated by a set of feelings and characteristics. Agents are dynamically assigned to these roles and have free reign in their action selection as long as it adheres to the schema.

With the IPOCL system Riedl and Young (2010) have explored the idea of intent-driven story planning, by augmenting a plot-level planner similar to MIMESIS (see Section 3.2.1) with the ability to chose believable goals that explain the actions taken by characters and to generate plans that make these goals believable. In order to achieve this, the STRIPS action representation was extended in two ways: Firstly, authors can classify actions into two sets: Non-happenings are character actions that need to be motivated, whereas happenings are actions that the author deems acceptable to appear in a story without a reason or motivation (e.g. accidents). Secondly, authors can specify special kind of effects using the *intends* keyword. This signifies that an action enables a character goal.

Listing 3.2 shows an example. The action of a monster to appear threatening to another character is a happening (i.e. the monster does not need to motivate it). The action enables a goal for said character, desiring the death of the monster. How might IPOCL use such an action? Consider a partial plan, in which a knight kills a monster, the action of killing being a non-happening and thus needing motivation. The planner might then pick the goal of wanting the monster dead for the knight to motivate his killing. In order to explain how the knight came to pursue this goal, the planner might then insert the appear-threatening action earlier into the plan.

Listing 3.2: Example of an IPOCL action (from Riedl and Young (2010))

```

Action: appear-threatening (?monster, ?char, ?place)
  actors: ?monster
  happening: t
  constraints: monster(?monster), character(?char), place(?place)
  precondition: at(?monster, ?place), at(?char, ?place), scary(?monster),
               ?monster≠char
  effect: intends(?char, ¬alive(?monster))

```

3.4.2 Drama Management For Character-Centric Systems

Character-centric systems have been augmented with a level of plot-awareness through the addition of centralised or distributed drama management.

Centralised Drama Management

A centralised drama manager is a component that may fulfil one or more of the following functions:

1. Add a high level plot structure by chaining together multiple character-based simulations
2. Control the simulation: Initialising the environment (including characters) and ending a simulation
3. Limit the character's set of behaviours to a subset of dramatically relevant ones
4. Incite exogenous story events (i.e. those outside of the characters' control)

The story facilitator in the FearNot! software architecture for example, which was discussed in the last chapter (see Section 2.3.1) is a typical example of such a drama manager and does in fact carry out all of the 4 above mentioned functions: 1) it structures the entire plot into smaller episodes, each of which is an emergent simulation of FAtiMA agents, 2) it sets up the initial conditions for each episode and monitors episode endings, 3) it limits character goals in each episode to an author defined subset and 4) external events can be specified by the author for each episode in the form of triggers.

In a similar vein, the Virtual Storyteller contains a plot agent that fulfils the above functions 1 and 2 by starting up scenes (Swartjes and Theune, 2008). Figure 3.8 shows how the plot agent ties into the overall Virtual Storyteller architecture. Besides controlling the simulation the plot agent also captures all occurring events in a fabula model. The Virtual Storyteller also contains a narrator component that selects a

subset of the events in the fabula model and turns them into an actual story. The narrator is another plot-centric element in an otherwise character-centric system, since it makes its decision based on a model of what constitutes a good narrated story. Finally, the Virtual Storyteller also uses distributed drama management (see next section).

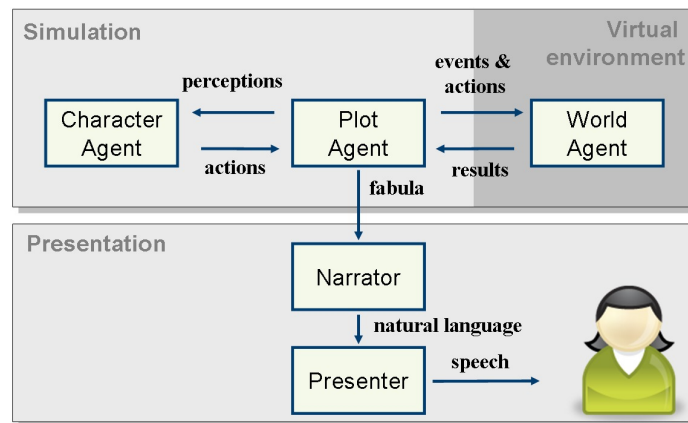


Figure 3.8: Architectural overview of the Virtual Storyteller (from Swartjes and Theune (2008))

In the Thespian System, the director agent is maintaining a set of author defined plot goals (Si et al., 2009). It may exert directorial control by changing character’s motivations and beliefs or even the world state. The director agent only does this, if it expects an imminent violation of one of its plot goals. Changes made by the director agent to a character agent have are motivated, i.e. they are not allowed to break an already established character. Belief and motivation changes can be justified by making use of past unknown events that have not been observed yet. This is an instance of a strategy called late commitment (Swartjes et al., 2008). Rather than pre-defining every single aspect of the world state, late commitment declares all as of yet unobserved state as changeable. This idea also underlies much of distributed drama management (see next section).

Distributed Drama Management

Distributed drama management revolves around the idea of autonomous agents that are aware not only of their role as a character but also on a meta-level of their role as an actor in a virtual drama. Swartjes et al. (2008) draw from improvisational theatre practises and uses the above mentioned late commitment strategy in the Virtual Storyteller system to allow the actor part of a character to fill in as of yet unknown details about the character in a way that supports dramatic plot development. The constructs that support this are called framing operators, special types of author defined STRIPS operators with the effect of asserting some world state proposition. Framing operators constitute the improvisational repertoire for the actor side of an

agent. They may be used both to justify adopting a new goal and within a normal plan. E.g. a pirate character on a ship may invoke a BeCaptain Framing Operator that makes the character the captain and opens up new goals (order an attack). This is only allowed if it does not contradict previously established facts (e.g. no other pirate on the ship has already been named captain). These restrictions are encoded as preconditions of the framing operator. In a similar vein, Arinbjarnar and Kudenko (2009) suggest an approach for representing both actor and character goals within the same character using object oriented Bayesian networks.

Louchart et al. (2007a) have incorporated a double appraisal mechanism into FAtiMA. When an agent has more than one option regarding which action to perform next, it will make this choice not “in character” but instead “out of character” by picking the event that causes the strongest emotional reaction in other agents, as determined by simulation of the emotional appraisal processes of other characters (hence double appraisal). The idea behind this strategy is that strong emotions result in better drama. Weallans et al. (2012) have built upon this work and refined the “out of character” action selection by incorporating a story specification, which defines an emotional target (rather than always aiming for the emotional reaction of the greatest magnitude), behind-the-scenes inter-agent communication of the agent’s actor layers and a virtual user model, as it is really the user’s emotional response to the ongoing action that is of most importance to the drama. Unlike other drama management approaches discussed here, Louchart and Weallan’s work does not require much further authoring effort. It does however pre-necessitate fully specified FAtiMA agents with an accurate emotional model, which as we have shown in Chapter 2 introduces a high baseline authoring load.

3.5 Conclusion

This chapter has covered a variety of approaches for representing interactive story content in IS systems. An obvious overall observation is that authorial intent (how directly can the author specify their vision) stands in direct conflict to generativity (to what degree can the system recombine story world knowledge to create new story variations). While being a popular technique for story representation in video games, explicit specification of branches has been largely dismissed as a viable solution for achieving fully realized IS artefacts, due to its combinatorial explosion. In other words, generativity is deemed essential. Unfortunately, the fact that authorial intent dwindles with increasing generativity, means that the more technically promising solutions are also those more challenging to author for. Generative approaches can focus on describing story elements at the level of plot or character, with both approaches having advantages and disadvantages. While hybrid solutions exist they typically

also introduce a duality of story representation that increases authoring effort and complexity. The fact that all of the reviewed systems are part of the same endeavour towards fully-realized IS and so far none of them has been able to demonstrate any large scale story worlds proves that the authoring bottleneck is very much a general phenomenon. More than anything it is a problem of quantity. While generativity helps with restructuring existing content, there needs to exist enough of it in the first place.

One aspect of authoring this chapter has not touched on is story representation. The discussed systems vary significantly in what story representation and user interaction paradigms they require, and in the degrees of effort that is involved in producing the materials necessary for presentation. Storytron for instance uses the toy language Deikto (Crawford, 2008), a story representation mechanism that directly operates on and exposes its story world data structures.

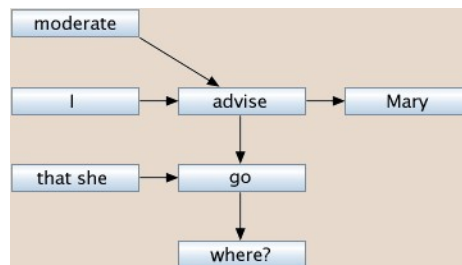


Figure 3.9: Example of a sentence in DEIKTO: I advise Mary with moderate urgency that she should go to “?”. The destination parameter is yet to be filled out by the user. Clicking on it will pop up a context menu listing possible locations.

This requires no further authoring effort for adding story representation to the story world but results in an extremely mechanical experience (see Figure 3.9) that some critics argue would rob any story of all its artistic beauty and remove all incentive for engaging with a story in the first place (Bond, 2007). On the other end of the spectrum, some IS story engines are designed to be coupled with highly immersive story representations in simulated 3D virtual worlds. These systems face a whole battery of follow-on authoring problems for the various pieces or representational content they require (characters, animations, objects, dialogue, etc).

The next chapter reviews the growing body of work that acknowledges, studies and attempts to solve these problems through authoring methods and tools.

Chapter 4

Authoring Methods And Tools

The previous chapter has shown what kind of content the author has to create in a variety of IS systems. All of these approaches have scalability problems and no existing story representation formalism or story generation algorithm offers an obvious solution to authoring bottleneck. This problem has been widely recognized by the IS community (Mateas and Stern, 2005; Spierling and Szilas, 2009; Skorupski et al., 2007; Pizzi and Cavazza, 2008; Medler and Magerko, 2006; Koenitz, 2011b; Iurgel, 2006) and spawned a growing body of work, studying ways to address the authoring bottleneck. This chapter discusses some methodological contributions towards defining the process of authoring, authoring tools that aim to support authors in creating story world data structures and finally data driven authoring approaches.

4.1 Defining The Authoring Process

4.1.1 Who are the authors?

One reason for the lack of large scale IS artefacts, is the lack of dedicated IS authors. For many of the IS runtime engines that were reviewed in the last section, the only story worlds that have been authored were created by the system designers themselves. Only few of them can afford to invest the effort to implement anything beyond small example proof-of-concept story worlds. Mateas and Stern (2005) relate that it took them around 3 person years to author the content for *Façade*. Their effort certainly paid off, as *Façade*, is still widely regarded as the closest anyone has come yet to fully realized IS. A fundamental assumption underlying most of IS research however is that the technology is ultimately build for a group of external authors, the question is only, who are they? Murray (1998) posits that the art form of IS needs a new type of artist, which she names cyber bard, equally comfortable in the realms of programming and storytelling. Crawford (2005) is equally insistent that programming skills are a necessary prerequisite for anyone who wishes to express themselves artistically through

the medium of IS.

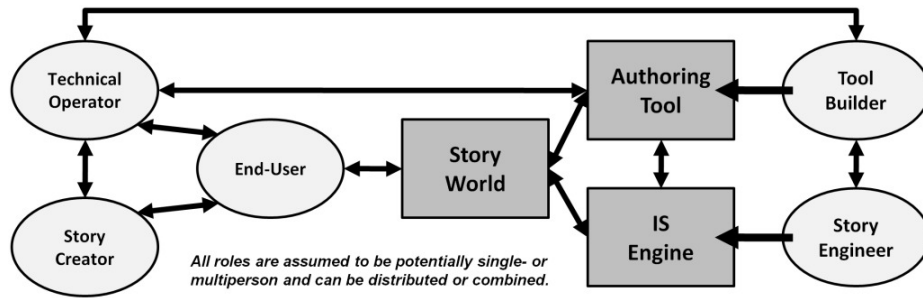


Figure 4.1: A view of the different roles assumed to participate in the creation of an IS artefact (from Hoffmann et al. (2011))

Hoffmann et al. (2011), inspired by the division of labour found in typical game development teams, break up the author’s role into story creator and technical operator, which represent the artistic and technical side of the cyber bard (see Figure 4.1). If these both roles are not consumed by the same person, it will nevertheless be important for both parties to have a common understanding to be able to effectively communicate and collaborate.

This thesis advocates a different distribution of roles that naturally follows from the use of crowdsourcing. The model proposed in the next chapter and implemented by the authoring tools that are described in subsequent Chapters distinguishes between a single principal author and multiple contributing authors. While it would certainly help if all involved are fully fledged cyber bards, a major thread of the work in this thesis is how to simplify the process so as to place as little requirements as possible on the contributing authors.

4.1.2 Authoring Metaphors

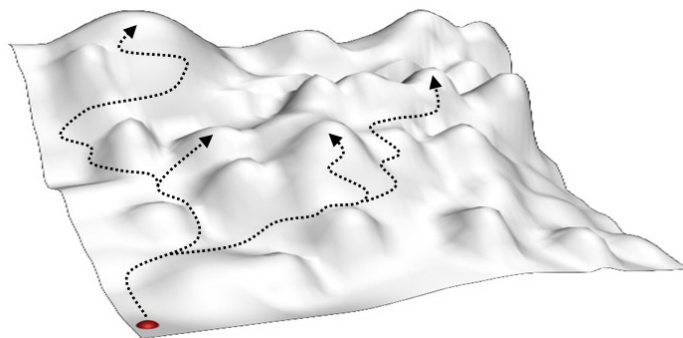


Figure 4.2: View of an IS story world as a landscape with several story paths shown

In a joint piece of work, which this thesis author was involved in (Louchart et al., 2008), we characterized the authoring of emergent narratives using the metaphor of a story landscape, which represents the total sum of all possible story experiences that

the story world is able to generate. Each location on the story landscape represents a possible story event and each actual end-user story experience is a path through this landscape (see Figure 4.2). This metaphor helps in describing three important properties of authored story worlds that the author needs to consider:

- **boundaries:** IS story worlds cannot be about everything. They need to have a theme, message, setting, characters, etc. In other words the story landscape cannot extend infinitely but needs to have a clear set of boundaries. These are not explicitly specified but rather implied in the authored content. Nevertheless, setting the boundaries (conceptually) and finding creative ways to justify them should be one of the first design considerations of an IS author.
- **critical mass:** Within the boundaries of the story world a critical mass of content needs to exist for emergence to occur. One of the conclusions of the FearNot! authoring case study in Chapter 2 was that in the case of FearNot! this critical mass was not quite achieved. In terms of the story landscape, critical mass should be thought of as density. It is not the absolute quantity of content that matters but content per “square inch” of the story landscape. Incidentally, this might be why Façade worked so well. Firstly it had very narrow boundaries. The setting is a single room where the player only interacts with two characters over the course of about 10 minutes. Secondly it had a very high density of content within these boundaries (as mentioned earlier 3 person years of authoring effort).
- **dead ends:** Dead ends are paths that just end abruptly. They signify a lack of content in the authored story world. Finding these dead ends and subsequently filling in the hole in the story content is a challenging aspect of authoring, especially because every new piece of content that is added has the potential of creating further dead ends.

These properties are also useful to characterize the crowdsourced authoring approach advocated by this thesis. Our entire approach (i.e. the use of crowdsourcing) is necessitated by the critical mass property and how to enforce boundaries for authors contributing through crowdsourcing is a key theme of this work. The new paradigm of crowd task adaptation that we introduce has the potential for creating more coherent story worlds, which also means fewer dead ends.

One key concept, which the story landscape is meant to convey is that of emergence. We do not explicitly create a landscape, but it implicitly emerges from the authored story world as processed by an IS runtime engine. It is this aspect of IS authoring that Spierling (2007) calls “implicit creation” and describes through a gardening metaphor (see Figure 4.3).

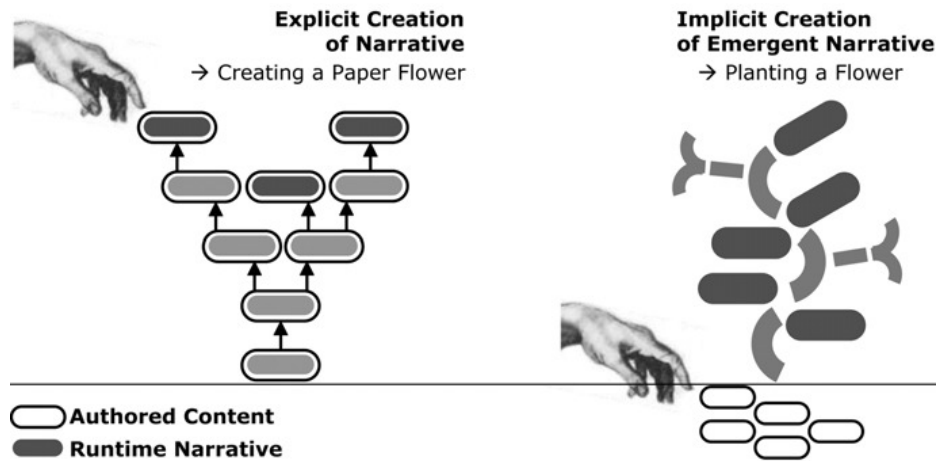


Figure 4.3: Gardening metaphor for implicit creation of IS story worlds (from Spierling (2007))

Unlike explicit authoring, which is likened to building a paper flower, where the author has full control over every extricate detail, implicit authoring relies on the plant to grow from the seeds planted by the author. By definition, it is impossible to predict the exact properties of the growing plant (i.e. resulting narrative). If the author could know exactly in advance what the plant will look like that grows from their seeds, it would have been easier to build a paper flower to start with. Spierling argues that implicit creation of narrative is a new design paradigm with a steep learning curve:

“Faced with the complexity of emergent systems, content creators need to approach implicit creation in steps, starting with explicit creation methods for its greater accessibility. There is a need for future research in identifying appropriate steps and developing supporting tools.”

4.2 Authoring Tools

We use the term authoring tools loosely to describe any software tools that aid authors in the creation of IS story worlds. This involves graphical editing of story world data structures as presented in the previous chapter, but may also cover other related functionality. The relation between an authoring tool and an IS runtime engine (with a particular story world representation) is analogous to that between a programming language and an IDE (integrated development environment). Medler and Magerko (2006) list the following requirements for their ideal IS authoring tool:

- **Generality:** The tool should support a variety of environments and story contexts.
- **Enables Debugging:** The tool should not only be able to create content but also allow the author to test it.

- **Usability:** The system must be user friendly, easy to learn and support efficient use.
- **Environment Representation:** The tool should be independent of any particular environment representation and instead use an intermediate format (e.g. of maps) in order to support different back ends.
- **Scope:** Ideally all authoring functionality should be integrated into a single tool to facilitate integration of and switching between tasks (e.g. action definitions and dialogue editing).
- **Pacing and Timing:** Authors should be able to control the pacing and timing of events.

In the following an overview of existing IS authoring tools is given. Rather than describing the tools separately in turn we structure our discussion with regards to the various features that IS authoring tools offer and also occasionally refer back to Medler and Magerko's requirements to characterize systems.

4.2.1 Prototyping

A variety of tools specifically aim to support the design rather than the implementation of IS story worlds. Their goal is to elicit and refine ideas and story elements. One such tool is Dramachina by Donikian and Portugal (2004). Its main function is the annotation and identification of narrative elements in written screenplay inputs provided by a writer. Elements that can be annotated can be of a plot-centric structural (acts, scenes, dramatic actions and units) as well as character-centric nature (relationships, character traits, emotions, etc). After the initial annotation in an actual text the elements are presented in a directory structure and can be further edited graphically. The resulting directory has two use cases. Firstly, it helps the author to take a structural view of their screenplay and assists them in making the transition from explicit to implicit creation modes. Secondly, the annotated elements can be exported in XML format to ease knowledge encoding for a particular IS engine (provided that a suitable importer is available).

Struck (2005) suggests the use of a database of film scenes, annotated with a character-driven drama model as a tool to aid authors in mapping out the protagonist's emotional journey. Hoffmann et al. (2011) suggest a paper-based approach for collaboratively prototyping a plot-based planning domain story representation. They achieved this by translating the execution of a planning algorithm into a board game rule set. This is intended to help a team to collaboratively prototype the planning domain design for a plan-based IS story world and to teach planning fundamentals to non-technical writers new to IS.

A different aspect of prototyping is the production of quick and dirty playable interactive stories. The key goal in this case is to produce a sort of interactive storyboard that can be used to demonstrate and discuss drafts of IS story worlds among teams. Tools for authoring of branching narratives as discussed in the next section are well suited for this purpose.

4.2.2 Authoring Explicit Branching

Branching narrative structures are easily understood and visualized. Authoring tools supporting this paradigm focus on Medler and Magerko's requirements of *usability*, i.e. making it easy for the user to view, navigate and edit branches and *scope*, i.e. allowing users to also edit most presentational aspects of the experience. INSCAPE (Balet, 2007) was devised as a full authoring suite encompassing all stages of design and production. INSCAPE was designed as an open architecture, which can be extended through plugins. Its core modules support visual editing of branching story lines. U-Create (Sauer et al., 2006) is a tool for creating mixed reality storytelling applications that run on mobile devices. It organizes stories in a hierarchical structure. A story world is a directed graph of scenes, which themselves are directed graphs of actions. Figure 4.4 shows the visual editing style of U-Create's scene graph view. Macvean et al. (2011) have built a similar system for geo-location enabled mobile story telling called WeQuest. Unlike U-Create, the WeQuest editor runs directly on a mobile device and thus allows authoring story points directly in their corresponding real world location.

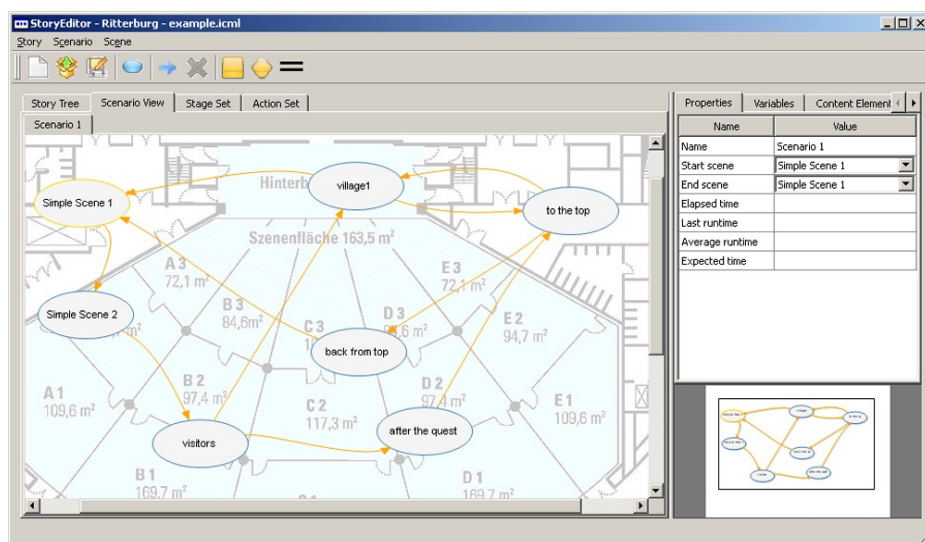


Figure 4.4: The Scene Graph View in the U-Create tool (from Sauer et al. (2006))

The Advanced Stories Authoring and Presentation System (ASAPS) by Koenitz (2011a) is a more recent tool for authoring branching interactive stories with a 2D graphical presentation and hyperlinks. Figure 4.5 shows the story graph editor of

the system. Koenitz and Chen (2012) call the ASAPS story representation model “procedural branching”, which they describe as follows:

“While branches have to be pre-determined by an author, the concrete decision on which branch to take can be determined at runtime depending on the state of a particular counter, the inventory system, or a variable.”

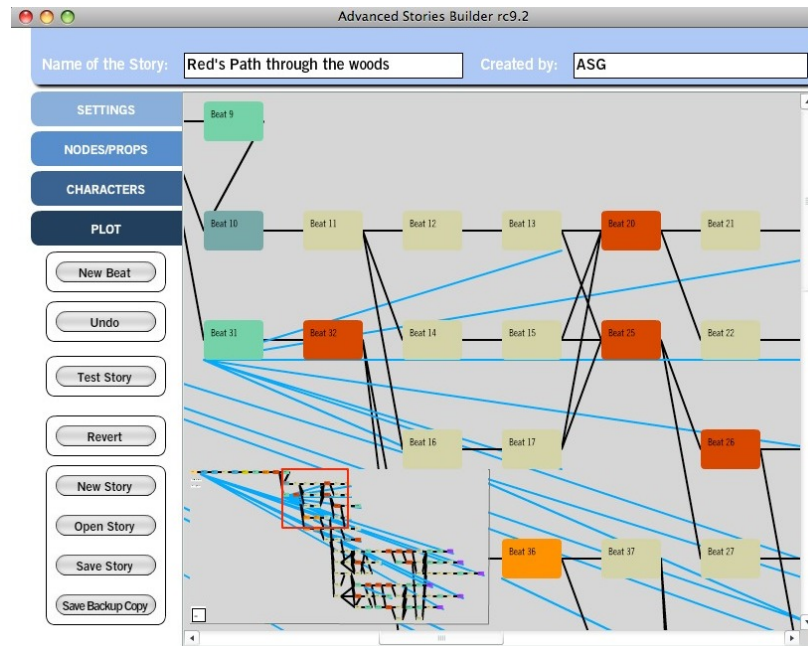


Figure 4.5: Visual Story Graph Editor in the ASAPS system (from Koenitz and Chen (2012))

4.2.3 Authoring Generative Data Structures

Authoring tools for creating generative story representations face the fundamental problem of making implicit creation intuitive. They usually offer features such as views to organize the story world elements, dialogues and wizards for creating new elements and context sensitive item selection through means such as drop down menus. These are all quite familiar features of programming IDEs¹ and as any programmer can attest they undoubtedly help increase productivity but cannot make programming conceptually easier. The actually challenging aspects of software development cannot be simply addressed with a fancier editor. The same is true for the authoring of generative IS as Spierling and Szilas (2009) argue. Authoring tools can help by “replacing the code generation form of typing by clicking” (Spierling, 2007) but they cannot take the conceptual challenge out of implicit creation.

We have already encountered one tool of this category. The Story World Authoring

¹IDEs such as Eclipse, NetBeans or Visual Studio offer for example object and class browsers, new class wizards, and text auto-complete while typing

Tool (SWAT) for the Storytron system (Crawford, 2007) was shown in Section 3.3.5. SWAT offers myriads of dialogues and views for manipulating all aspects of a story world, as Storytron is exclusively based on visual editing, i.e. there are no “code editor” windows in SWAT.

WideRuled (Skorupski et al., 2007) is a graphical authoring tool for the Universe story model (Lebowitz, 1985), an author-based model of planning that represents stories as pre-authored hierarchical plans not unlike HTNs. However, in contrast to the HTN based work of Cavazza et al. (2002), discussed in the last chapter, Universe plans are not character but plot-centric. The main structural units of Wide Ruled (and Universe) are “Author Goal” and “Plot Fragment”. Each plot fragment belongs to exactly one author goal and may link to further children author goals. Plot fragments contain a set of preconditions that need to hold true for the fragment to be eligible for execution by the runtime engine and a set of actions that the plot fragment executes. Plot fragment preconditions can bind variables that can be reused in the actions. Figure 4.6 shows the author’s view of a plot fragment. In this example *victimName* and *friendName* are two variables defined by the plot fragment. preconditions and actions are created and edited through a number of popup dialogue windows.

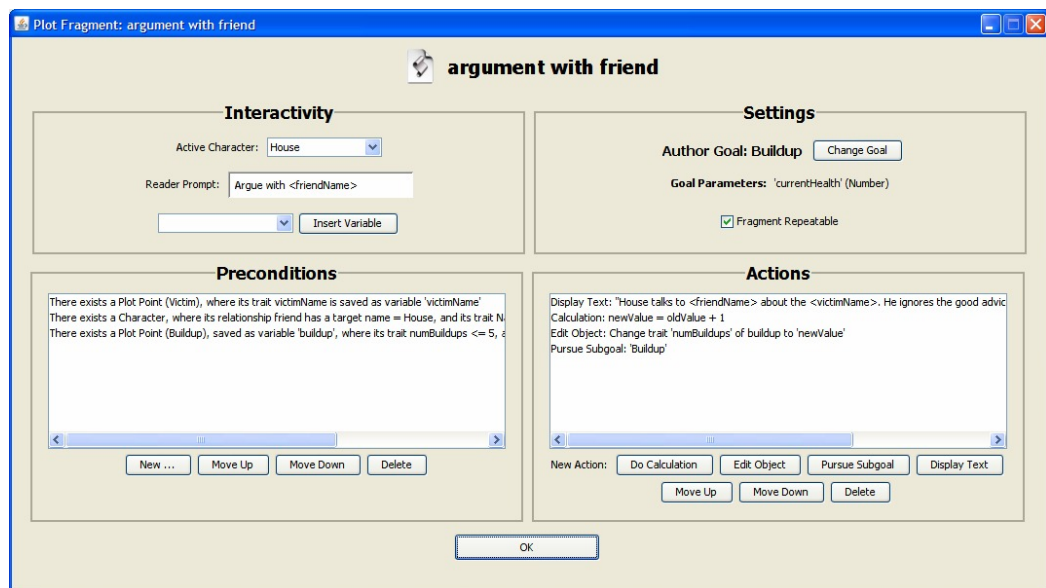


Figure 4.6: Wide Ruled: Plot Fragment Editor (from Skorupski et al. (2007))

A similar style of planning domain authoring is afforded by the Scribe authoring tool by Medler and Magerko (2006) and the tool by Pizzi and Cavazza (2008) for the EmoEmma system. Renaissance by Zancanaro et al. (2001) is a graphical knowledge base editor for a frame-based production rules system used in educational story based games, while Scenejo by Weiss et al. (2005) is a tool for authoring dynamic dialogue scripts. The latter two systems are not based on planning formalisms as the others mentioned before but they offer similar authorial affordances and use similar UI

paradigms. The Bowman authoring tool (Thomas and Young, 2006) takes an unusual approach in that it interleaves plan execution with author feedback to develop plan heuristics in a meta-language. Its mixed-initiative approach can be used to gain data for augmenting a plot planner with heuristics to guide which of otherwise identical plans to prefer.

A version 2 of Wide Ruled with minor UI improvements was introduced by Skorupski and Mateas (2009), but the authors eventually concluded that a more novice friendly authoring environment was needed. Addressing this, they produced Story Canvas (Skorupski and Mateas, 2010), a radically redesigned authoring tool for the same underlying Universe story model. Story Canvas uses a comics story visualization and in general more visual editing and less dialogue windows in comparison to Wide Ruled. Its most innovative contribution is the representation of temporary variable bindings. Figure 4.7 shows the plot fragment editor of Story Canvas, which offers exactly the same plot fragment editing capabilities as that of Wide Ruled shown in Figure 4.6. Characters that are unspecified at authoring time are represented as colour-coded silhouetted shapes in the comics visualization, while character relationships are represented as bidirectional arrows between character shapes. Unfortunately there are no studies available that relate how Story Canvas’ use of visual metaphors affected the performance of authors.

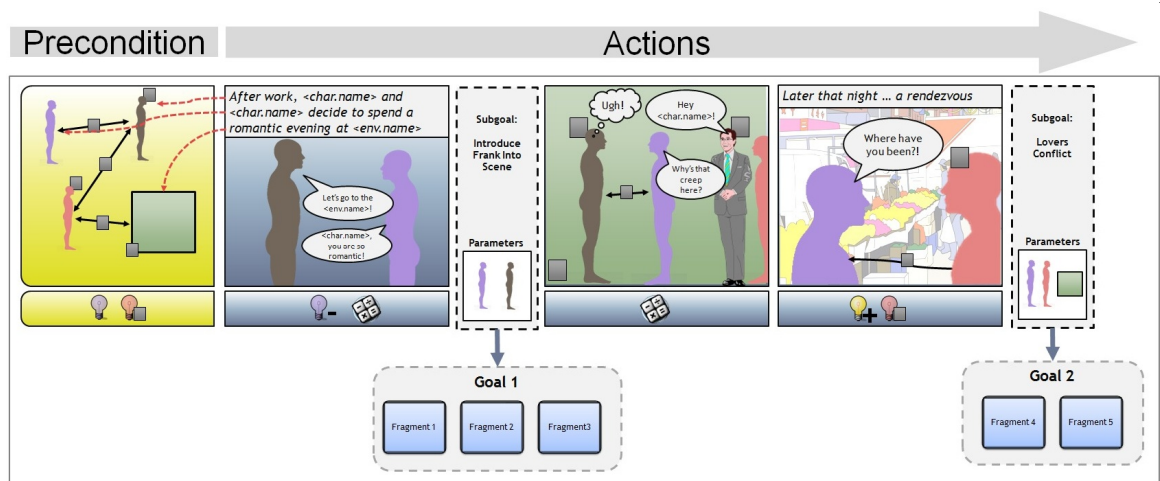


Figure 4.7: Story Canvas: Plot Fragment Editor (from Skorupski and Mateas (2010))

4.2.4 Debugging

With growing complexity of authored story worlds, authors will inevitably make mistakes that need to be corrected. Authoring tools can support this through debugging features. The simplest of these is for the authoring tool to allow the author to swiftly switch to story execution mode and observe running the authored story world. This also enables iterative authoring and cocreation methodologies (Louchart et al., 2007b;

Swartjes and Theune, 2009). Most authoring tools support this basic functionality, but some go further in that they allow the step wise execution of plans (Pizzi and Cavazza, 2008) or the manipulation and monitoring of story world states during execution (Medler and Magerko, 2006; Pizzi and Cavazza, 2008). This is in line with the functionality provided by debuggers for programming languages.

Another popular way for finding bugs in software programs is the use of static analysis tools. These tools perform an offline analysis of the entire source code of a computer program and detect problematic issues such as memory leaks. Equivalent functionality also exists in IS authoring tools, especially those based on planning formalisms. The EmoEmma authoring tool by Pizzi and Cavazza (2008) checks for completeness and consistency of the authored planning domain. Similar functionality is provided by Dang et al. (2011) in a stand alone tool that uses a linear logic proof to identify for example unused operators, dead ends, unused propositions and unreachable goal states. The Storytron editor SWAT (Crawford, 2007) also offers a suite of analysis tools to authors. An author may for example invoke its “Scriptalyzer” module on a script of their choosing, which performs a Monte Carlo simulation on the script by reporting the effects of different random combinations of the script’s input variables.

4.2.5 Affecting Story Presentation

As mentioned earlier, story presentation styles vary wildly in different IS systems. If a system employs any form of graphical story presentation, some sort of tool support for configuring the presentation is absolutely vital. Common tasks include configuration of the environment, placement of elements in the virtual world, definition of character animations and behaviours, association of sound effects, etc. However, much of this functionality can be found in professional third party tools. Therefore many systems use professional game engines and their associated editors for the majority of presentation related authoring such as the Unreal Engine editor (Cavazza et al., 2002; Magerko, 2002; Young et al., 2004; Pizzi et al., 2007), the Neverwinter Nights editor Aurora (Thue et al., 2007) or the Unity3D game engine and its editor (Nazir et al., 2012).

Scribe (Medler and Magerko, 2006) has an emplacement editor that allows users to manipulate the spatial configuration of the story world but for more detailed visual configurations (such as animation phases of a character) external tools still have to be used.

Many of the tools reviewed above for authoring explicit branching offer in-tool editing of all aspects of the visual appearance of the resulting story. ASAPS (Koenitz, 2011a) for example offers simple layouting of 2D scenes. INSCAPE (Balet, 2007) also supports complex 3D visualizations and while addressing the *scope* requirement, raises the question how much more this project could have been achieved in terms

of authoring interfaces for generative IS, if the resources spent on replicating well understood game editor functionality would have been directed towards it.

4.2.6 Educational Authoring

A discussion of authoring tools would not be complete without mentioning those systems, whose primary or secondary goal is computer science education. These systems take the view that storytelling is a suitable gateway domain to get acquainted with basic computer science concepts such as data types, variables or function calls. This approach has been successfully applied across different age groups.

Adventure Author (Robertson and Nicholson, 2007) is a tool targeted at young children for making story-based adventure games. Storytelling Alice (Kelleher, 2009) is a storytelling variant of the visual programming language Alice that allows users (the designed target group are middle school children) to explore the basics of scripting. Emohawk (Brom et al., 2009) targets a more mature audience with existing computer science knowledge. It is designed primarily as a vehicle for CS students to experiment with autonomous agent technologies and algorithms.

4.3 Data Driven Authoring

Several IS systems have implemented authoring solutions that use data-mining techniques in order to configure an IS story world. Data being mined is predominantly provided in one form: example stories or story fragments. The author's task then is the collection or creation of these examples in order to provide them to an authoring system that processes these cases and automatically extracts features from them to incorporate into the story world. Swartjes (2007) calls this process "authoring with narrative cases". There are also many parallels to the "programming by demonstration" paradigm (Cypher, 1993). Authoring for IS systems that use case-based reasoning (CBR) is practically identical, the difference lying solely in whether the cases are used online (cases are part of the actual story world and processed at runtime) or offline (cases are an intermediate representation and used to modify the actual story world data structures in a separate processing step at authoring time).

A system that adopts this latter strategy is Thespian, which was introduced in Section 3.3.4. Thespian represents agent personality as a vector of relative goal weights. It was found that setting these weights to suitable values by hand is a time consuming and tedious process. The authoring suite for Thespian agents therefore includes a mechanism for fitting an agent's personality (as expressed by its goal weights) so that it best matches a set of author provided example stories, which act as valid demonstrations of believable behaviour of this agent (Si et al., 2005). Figure 4.8 illustrates

how the goal fitting procedure is integrated into the Thespian authoring process.

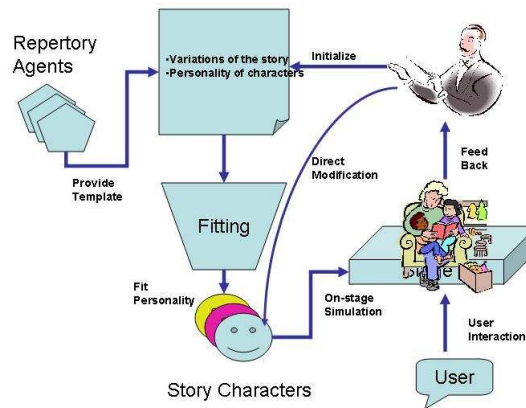


Figure 4.8: The Thespian Authoring Process with its Goal Fitting Procedure (from Si et al. (2005))

The ActAffAct system by Rank and Petta (2012) contains a similar feature that allows the author to provide example episodes of a character’s past, which the system uses offline as in Thespian for tuning the character’s personality parameters. While the story examples here are framed as backstory, whereas in Thespian they specify character behaviour in hypothetical situations, this distinction makes no discernible difference in practise.

Once we consider the provision of narrative cases as a valid authoring method, it is a small leap to consider crowdsourcing the collection of these narrative cases. As already mentioned in the introduction of this thesis, Orkin (2011) with the Restaurant and Improviso systems and Li et al. (2012) with the Scheherazade system have taken this step. As their work is highly related to the original work carried out within this PhD, descriptions of their systems and a comparison to this work are provided in the following sections.

4.3.1 The Restaurant and Improviso

The Restaurant Game (Orkin and Roy, 2007) was launched in 2007 as an experiment in salvaging game play data for the authoring of intelligent agent behaviour. The two-player game lets pairs of players act out the interactions between a waitress and a customer in a 3D virtual restaurant environment (see Figure 4.9 (left)). The game only starts when two players, which are randomly matched by the system, are both present. What ensues is entirely up to the players. A session might follow the perfectly ordinary, if slightly boring steps of greeting, ordering, serving, eating and paying or might turn into a surrealist exchange of oddities. The system does not reward, comment or judge the performance. For players the enjoyment derives from the anonymous participation in an improv session with a total stranger.



Figure 4.9: Left: The Restaurant Game, Right: Improviso

Players can move through the restaurant environment (it is impossible to leave this locale), use a few predefined physical actions such as (eat, pick up, sit down, etc) on the objects available in the environment and communicate with each other through text input (As there is only one interaction partner this is simply a matter typing and hitting Enter). All these actions of both players are recorded by the system as a gameplay trace, which is The Restaurant's equivalent of a narrative case. Orkin has recorded over 5000 of such traces. Later work of the same team describes how one of the roles (waitress or customer) could be automated using this corpus of recorded traces (Orkin and Roy, 2009) and how manual task annotation of the traces (i.e. describing which task each recorded action is related to) improves believability of the generated behaviour (Orkin et al., 2010).

The project demonstrated how data generated as a by-product of multiplayer gaming that would otherwise just go to waste can be used to drive the behaviour of AI agents. But the interactions recorded in The Restaurant were mostly not inherently dramatic and as a result the model of behaviour built from the Restaurant game play traces was mostly a representation of normal, acceptable behaviour in a restaurant situation. Improviso (Orkin, 2011) was an attempt to apply the same technology to a game play situation that is more amenable to dramatic improvisation. Improviso allows two or more players to enact the chaos that ensues around a UFO crash site, with playable characters including aliens, FBI agents, scientists and a reporter (see Figure 4.9 (right)). Interaction modalities are similar to the Restaurant but players are also allowed to switch between characters and can even perform some directorial actions. Unfortunately, no further publications relate how the system was received and what kind of data was collected.

4.3.2 Scheherazade

The Scheherazade system (Li, 2012; Li et al., 2012) uses crowdsourcing on Amazon’s Mechanical Turk platform to collect example scripts for a given social situation (e.g. a bank robbery). Posting the HIT² on Mechanical Turk and collecting the incoming results is a built-in automated part of the system. The “author”, i.e. the initiator of a story picks a few characters and a social situation. A short description of the characters and the type of social situation is then provided to the workers who have to provide write a text that details one possible chain of events describing how the situation unfolds. Workers are asked to use short simple sentences that describe one event and have only one verb.

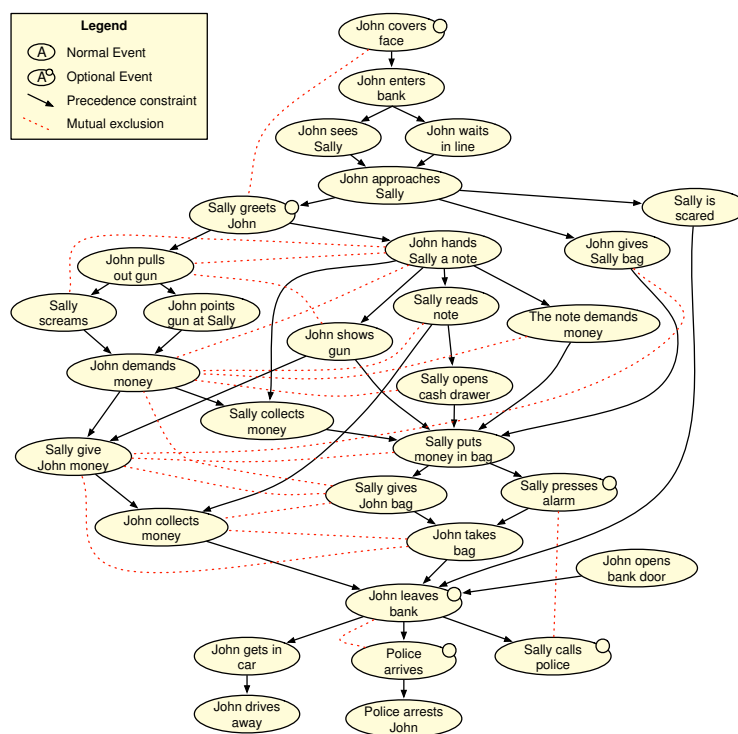


Figure 4.10: A Scheherazade plot graph for a bank robbery situation (from Li et al. (2012))

The system uses the collected corpus of textual situation descriptions to build the following plot graph representation: a directed graph where nodes are events, which may be optional (event may be skipped without making the story invalid), precedence constraints (in any story event A must happen before event B) are represented as directed edges and mutual exclusion relations (events A and B may not both appear in the same story) as undirected edges. To derive such a graph as shown in Figure 4.10,

²HIT stands for Human Intelligence Task and is Amazon’s terminology for a task to be completed by Mechanical Turk workers

the system first splits each story into individual sentences and performs clustering on the set of all sentences to identify unique events. It then derives precedence constraints from the observed probabilities that 2 events follow each other. These probabilities need to be above a certain threshold to combat noise. With precedence constraints in place mutual exclusion links and optional events are calculated based on the statistical interdependence between pairs of events. Traversing a complete plot graph yields an interactive story if at stages where multiple options are available, these are presented as choices to the user.

More recent publications on Scheherazade (Li et al., 2014b,a) have focused on the automated narration of the generated stories. In order to produce more readable and entertaining accounts of paths through the plot graph, the system employs a second round of Crowdsourcing (after the plot graph has been learned) in which explicitly colorful textual descriptions of the same social situation are being collected.

4.3.3 Comparison

The systems discussed are the most similar related work to the systems implemented for this PhD: ENIGMA (subject of Chapter 6) and CROSCAT (subject of Chapter 7). In order to contextualize our work, Table 4.1 briefly summarizes some of the main differences between these systems. It shows that our work provides some novel view point on the use of crowdsourced authoring. Most importantly we uniquely use crowd task adaptation (a concept defined in the next chapter), but we also differ from both the Restaurant / Improviso and Scheherazade systems in several other aspects.

	The Restaurant Improviso	Scheherazade	ENIGMA CROSCAT
multiplayer creation	Yes	No	No
natural language processing	Yes	Yes	No
focus on dramatic stories	No / Yes	No	Yes
story representation	3D graphics	text	2D comics
specialised authoring tool	Yes	No	Yes
story domain independent	No	Yes	Yes
authors have time to deliberate	No	Yes	Yes
crowd task adaptation	No	No	Yes

Table 4.1: Feature comparison between IS authoring systems that employ crowdsourcing

4.4 Conclusion

This chapter has reviewed a variety of IS authoring approaches. Authoring tools offering debugging, visual editing and intelligent decision making support features certainly can contribute to educating authors, shaping their approach to story world design and construction and aid them in the actual implementation. However, none of the tools address the fundamental problem of scalability. The ideal authoring tool requirements by Medler and Magerko (2006) also do not include scalability, indicating that this is an aspect that has been given little attention so far. We would argue that for the ambitious goal of achieving fully realized IS, tools designed for scalability are needed and that the best *modus operandi* for these tools is to facilitate collaboration of multiple authors.

This chapter has also reviewed data-driven authoring approaches as a promising way to tackle the scalability / quantity problem. But only few systems have explicitly considered how data-driven authoring can be effectively applied to multiple authors. The remainder of this thesis tries to address this question. First we establish effective collaboration models for IS authoring, building upon patterns shown by the data-driven systems. In subsequent chapters we then describe our own authoring tools based on these models.

Part II

Crowdsourced Authoring

Individually, we are one drop. Together, we are an ocean.

Ryunosuke Satoro

Chapter 5

Crowd Task Adaptation

The first part of this thesis has demonstrated the conceptual and scalability difficulties found in contemporary approaches to IS authoring. Independent of the approach taken to encode IS story worlds, IS runtime engines are by their very nature extremely data-hungry and it is the author’s responsibility to provide this data. As discussed in the previous chapter, involving an online crowd in its creation is a promising approach for obtaining a critical mass of data. In this chapter we home in on this idea and turn to various types of mass collaboration systems enabled through the pervasiveness of the World Wide Web (Doan et al., 2010), discuss how they can be applied to IS authoring. As a result of this exploration, the novel concept of Crowd Task Adaptation is suggested as a modification to the crowdsourcing workflow. This is followed by a discussion of reasons for why this new approach should boost efficiency and the value of the resulting data.

5.1 Defining Crowdsourcing

5.1.1 Crowdsourcing and other crowd-powered approaches

Crowdsourcing is one of the most common buzzwords of the computing industry in the early 21st century. Howe (2006) defines crowdsourcing as “... the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call... The crucial prerequisite is the use of the open call format and the large network of potential laborers.”. Based on this definition, crowdsourcing possesses some crucial characteristic that distinguish it from related large-scale problem solving and content production approaches.

Figure 5.1 contrasts Crowdsourcing with Data mining and Commons-based peer production. All these three approaches make use of the unprecedented scale of connectedness provided by the internet and enlist a crowd of internet users for achieving

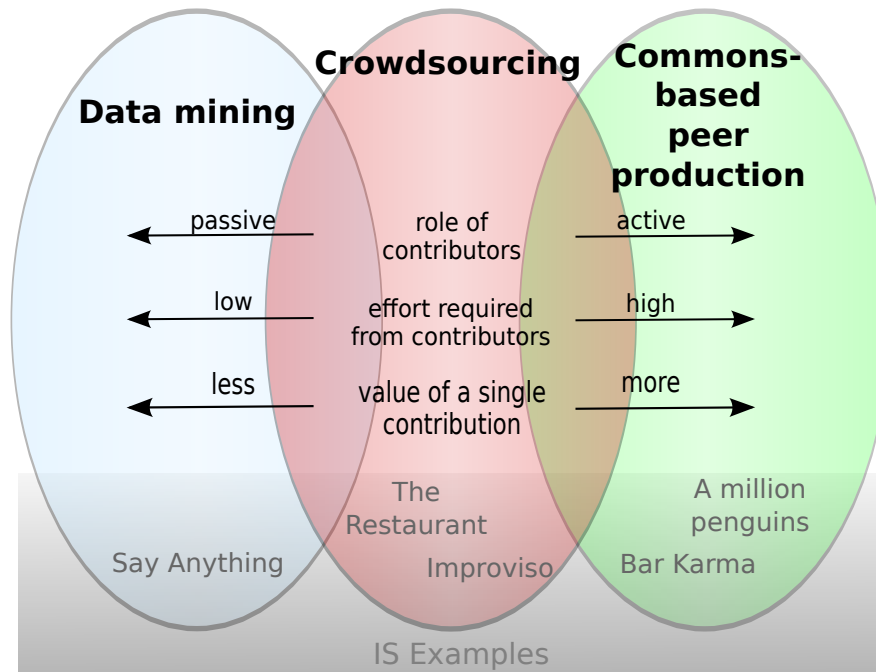


Figure 5.1: Crowdsourcing and related crowd-powered approaches

some goal.

Data mining underpins the business model of many modern internet companies, which capture the online activities of their users to generate value. The collected data is processed using machine learning techniques and used for example to provide various types of recommendations, from targeted advertising to shopping recommendations or customised internet searches. Users are usually passively contributing to such a data mining effort and are often not even aware of their own participation. The collected data is merely a byproduct and users do not put any effort into creating the data. Data mining is crucially lacking the “open call” to adhere to the definition of crowdsourcing.

At the other end of the spectrum, **Commons-based peer production** (Benkler and Nissenbaum, 2006) refers to highly democratic internet based production processes. The internet gives people interested in a common goal a platform for collaborating in an efficient manner. Two prime examples for successful Commons-based peer production can be found in the online encyclopaedia Wikipedia and large scale open source software projects such as the Linux operating system. In both cases a large number of volunteers have pooled their resources to produce world-leading artefacts. The key distinction between crowdsourcing and Commons-based peer production lies in the organization of work.

Crowdsourcing implies a distinction between a central organizing entity and crowd workers. The central organizer issues and promotes the open call for participation, defines the tasks for crowd workers and finally collects and processes the collection

results. In contrast, participators in a Commons-based peer production play an active role in the organisation of work and definition of tasks. Wikipedia is an excellent example to illustrate this point: A contributor gets to choose which article they want to edit or create. Differences of opinion are dealt with by the community through discussions and an elaborate set of rules of conduct, which are in turn established by the community. In contrast, in a crowdsourcing model a central encyclopaedia editor would farm out tasks for the creation of certain articles and consolidate any inconsistencies using their own judgement. They might also use crowdsourcing to obtain a vote on certain decisions, but the decision on whether to involve the crowd in the first instance and whether to adhere to the voting result or not would always lie with the central editor.

5.1.2 Storytelling Examples

The degree of involvement of a contributor (passive to active) in the 3 crowd-powered approaches naturally influences the amount of effort that contributors have to invest in order to participate. However, this effort does also pay off in terms of the average value that can be derived from a single contribution. Figure 5.1 shows how some examples from the realm of Storytelling (in some of the cases interactive) relate to our distinction of crowd-powered approaches.

Say Anything

The “Say Anything” Open Domain Story Writing Companion by Swanson and Gordon (2008) crawls a huge amount of openly accessible blogs, which contain personal stories of internet users and stores single sentences from these stories in a database. This database enables an Interactive Storytelling mode where the user and the system jointly construct a story in turns. The user types a sentence, which provides the features for looking up suitable follow up sentences in the sentence database. The user is presented with a set of the highest-scoring follow-up sentences. The user picks one of the suggested sentences and then continues the story by typing another sentence. This is followed by another database sentence lookup, which can now take into account features from the entire story history so far. This turn taking continues as long as the user wants.

The stories usually become increasingly bizarre / surreal with every added sentence and fall apart at some point, due to the growing mismatch between the context understood by a human reader of the story and that captured in the retrieval features. Nevertheless, there is no denying that “Say Anything” has entertainment value and potential applications as a creativity / inspirational source for writers aid and as a teaching tool for creative writing. Its open domain nature make it stand out from the

traditional idea of Interactive Storytelling that this thesis focusses on. Subsequent research on the project has focussed on the choice of retrieval models for looking up candidate sentences (Swanson and Gordon, 2009), the ranking of the presented candidate sentences and sentence adaptation to fit the story context Swanson and Gordon (2010).

“Say Anything” firmly falls into the “Data Mining” category of our taxonomy of crowd-powered approaches. The data contributors in “Say Anything” are various bloggers across the world. They were truly passive with regards to their involvement in the project itself. Most of them will not have been aware of “Say Anything” and many contributions might in fact have been made before the inception of the “Say Anything” project. None of the bloggers have invested any additional effort in order to participate in “Say Anything”. Consequently the relative value of a single contribution is not very high but this is compensated by quantity. Finally, it should be noted that Swanson and Gordon (2010) report on their use of crowdsourcing for evaluating “Say Anything”. However, this has no bearing on the authoring process.

The Restaurant, Improviso, Games With A Purpose and SNACS

The approach taken by (Orkin, 2011) on the games “The Restaurant” and “Improviso”, which we have already discussed in previous chapters can be classified as “Crowdsourcing”. With these projects, a central organising entity (Orkin himself) has created a constrained task for contributors. The task consists of collaboratively acting out improvised fictional scenes in a video game style application. The story domain is constrained by the content present in the game and the instructions provided along with it. Consequently participants have to accept certain restrictions and clearly do not have the possibility to significantly influence the overall direction of the project as they would have in a Commons-based peer production. However they do take a significantly more active role than the bloggers that provided the material for Data Mining in “Say Anything”. A contributor to either “The Restaurant” or “Improviso” will have responded to an open call for participation and have had some motivation for participating. They will in all likelihood been aware of their participation in a crowd sourced data collection and the broad purpose of the “game”.

The Restaurant and Improviso can also be seen as examples of “Games with a Purpose”, a term established by von Ahn and Dabbish (2004). The idea of a “Game With A Purpose” is to disguise a Crowdsourcing task behind a layer of gameplay. As games are commonly associated with fun, the gameplay layer provides a motivation for participating. Von Ahn has provided several prototype examples of such games: In the ESP game (von Ahn and Dabbish, 2004) two players try to achieve a joint highscore by scoring points if they come up with the same keywords for a given image. The images fed to them while playing are taken from a corpus of images that need to be

labelled and the keywords they type while playing the game can be used for generating labels. Following similar principles, Peekaboom (von Ahn et al., 2006b) is a game for learning the location of objects in images, Verbosity (von Ahn et al., 2006a) collects common sense knowledge (e.g. the fact that a dog is a kind of animal) through the means of a guessing game and Tagatune (Law et al., 2007) uses the game mechanics of the ESP game for sound and music annotation instead of image labelling.

“Games with a purpose” show that the boundary between Crowdsourcing and Data Mining in our distinction of crowd-powered approaches is blurry. This is especially true, if the game disguises the task extremely well and the participant is not even aware of the fact that they are contributing to some data collection effort by playing the game.

Our final example for the use of Crowdsourcing in creating narratives is the SNACS system (Sina et al., 2013). SNACS is used for collecting social narratives through Amazon’s Mechanical Turk platform. The collected narratives are short diary-like descriptions of some recent social activity the worker has been involved in. In addition to these natural language narratives, contributors are also providing metadata in the form of attributes describing themselves and their narrative by filling out some questionnaires. Based on this data the system is able to create new narratives for a given character profile by adapting collected narratives to a different context.

A Million Penguins and Bar Karma

The “A Million Penguins” project (Mason and Thomas, 2008) sponsored by the publishing house Penguin Books aimed to study the application of Commons-based peer production principles to creative writing. The grand goal was the joint creation of a wikinovel, a novel written by internet users using a Wiki platform. Users had to organize and coordinate their writing contributions. They had to decide on a theme, setting, characters and plot for the novel and somehow resolve their differences of opinion. This was reportedly not always easy. It is not hard to imagine that the types of person that are passionate enough about creative writing to contribute to such an endeavour also feel a sense of pride and ownership in their work and ideas and more importantly have strong opinions. When people’s ideas get rejected, feelings are easily hurt. Not surprisingly the “A Million Penguins” community relatively quickly fractured with several groups working on alternative “versions” of the final work. The outcome of this creative experiment is therefore not a single monolithic novel but a wiki full of fragmented stories and unfinished pieces, which due to its multi-faceted nature has certain similarities to an IS artefact. Nevertheless an astonishing amount of collaboration also took place and many participants thoroughly enjoyed the experience. The lesson to be learned for our discussion of crowd-powered creation processes is that active involvement of participants can lead to conflicts that are not typically

observed in a Crowdsourcing situation. While people may really pour an unprecedented amount of hard work into the collaboration they will also much more fiercely fight for their contribution to be acknowledged.

Another example for Commons-based peer production principles applied to Storytelling was Bar Karma (Zucker-Scharff, 2011), a short-lived community-written TV-show on a small independent channel. Bar Karma was a project by the famous video game designer Will Wright (creator of *Sim City*, *The Sims* and *Spore*). Using a piece of software called “StoryMaker”, users were able to draft storyboards for future episodes on the show’s website. User’s could provide storyboards for entire episodes or only single scenes. Discussions and collaborative editing were also possible, with “StoryMaker” laying out all proposed storylines in a graphical story graph (see Figure 5.2). After a deadline date, the community would be invited to vote for their favourite storyboard, which would then in turn be filmed by the show’s producers. This second voting stage is more characteristic of Crowdsourcing, as during this stage users had a well defined task. However, this is merely a narrowing down of existing content. The actual content generation falls in the Commons-based peer production category.

Unfortunately reports about the authoring process of Bar Karma are hard to come by, therefore we cannot say more about how well the community collaborated and if there were any conflicts. Arguably, Bar Karma qualifies as a form of Interactive Storytelling, as the audience can influence the overall storyline of the show on an episode-by-episode basis, but for the narrower definition of computer-based IS that this thesis has adopted, we do not consider Bar Karma as an IS artefact.

5.2 Attacking the Authoring Bottleneck with a Crowd

5.2.1 Useful Data

The goal of this thesis is to work towards a crowd-powered solution for the problem of the Authoring Bottleneck. This clearly involves the crowd producing either an entire story world by themselves or alternatively at least some data that can be classified as useful during the creation of an IS story world. Useful in the latter case could be more precisely defined as significantly reducing the authoring effort required for authoring this story world. We can identify several potential types of useful data.

Story World Data Structures

In Chapters 2 and 3 we have surveyed the data structures in use for encoding story worlds in a variety of prominent IS Runtime Engines. It would be ideal if we could get data that adheres to one of these formats, so that it is directly readable and

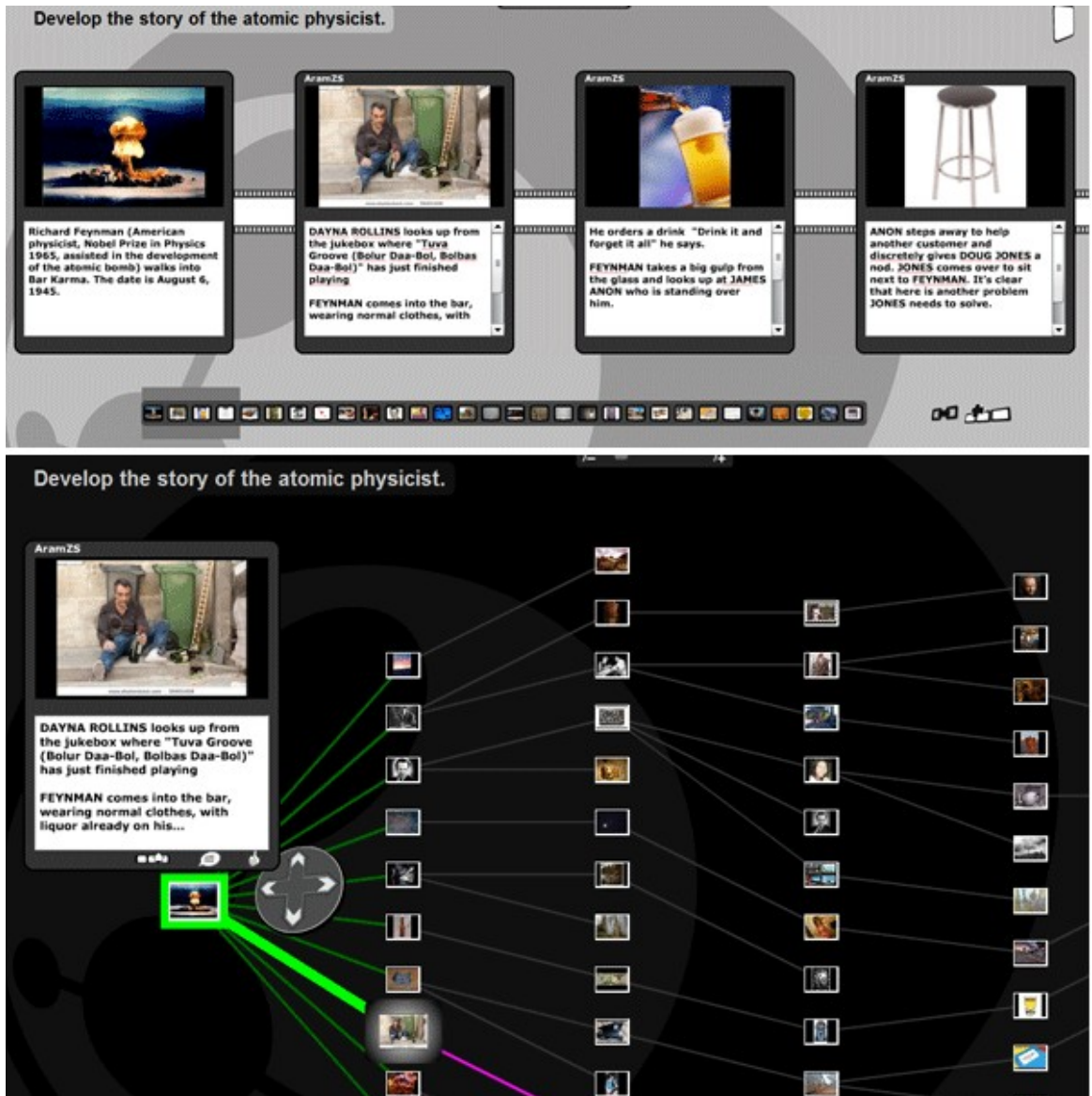


Figure 5.2: Screenshots of the StoryMaker application used to collaboratively edit storyboards for Bar Karma episodes

processable by an IS Runtime Engine. The data on its own could make up an entire complete story world or complement additional data structures that are authored before and / or after the data collection from the crowd. Most story world data structures encode knowledge in one form or another. Several projects exist that have employed crowdsourcing for collecting common sense knowledge (Singh, 2002; Lenat, 1995). Some of them (Lieberman et al., 2007; von Ahn et al., 2006a) follow the “Games With A Purpose” paradigm of hiding the data collection behind a game play layer. Consequently, this type of data-collection is not entirely new ground and templates for how to efficiently collect data exist.

Example Stories

Considering that the majority of crowd members will not possess the expert knowledge for directly producing the required data-structures, we might settle for an intermediate more human-friendly representation. In order to produce an IS story world such an intermediate presentation will then have to be processed further into the data structures required by the IS Runtime Engine in use. This transformation could be done using manual labour or automated processes (e.g. utilising machine learning methods). In Chapter 4 we have encountered multiple instances where example stories were used as such an intermediate presentation (Orkin and Roy, 2007; Li et al., 2012; Rank and Petta, 2012; Si et al., 2005). Not all of these systems have used the ability of providing example stories as input as a means to enable crowd-powered data collection. Instead, some systems simply use example stories as an aid for a trained individual author. The fact that such an aid is necessary and useful, supports our observation from chapter 2 that thinking in linear stories is fundamentally easier than thinking in the abstract data structures in use by most IS Runtime engines. And of course we have seen several case-based reasoning systems, where example stories are the primary data structures used by the IS Runtime Engine.

Furthermore, in our literature review we did not encounter any other intermediate representations of story world content. Based on this evidence we can conclude that example stories are the most natural and preferred medium of specifying the contents of a story world. However, example stories are a high-level concept that can be technically represented in a large variety of ways. When collecting example stories from the crowd with the aim of helping us to design an IS story world, their usefulness will depend partly on their representation. Probably the most natural forms of story representation are those that we are used to consuming ourselves: stories written in prose / natural language, drawn stories (e.g. comics) or film / animation are all human readable representations whose creation processes are well understood. On the other hand, stories could be represented in a more machine-friendly representation. For example the representation could specify a structuring that identifies distinct events, the actions taking place, the characters involved in the action, any parameters modifying the action, etc. Details vary depending on the specific story world format, but in general data provided in such a representation is more useful, as it allows for an easier transformation into the data structures used for describing an IS story world. Consequently there is a tradeoff between authoring effort (how difficult is it for a crowd member to contribute an example story) and transformation effort (how difficult is it for a machine or human expert to extract useful data from the example story that contributes to an IS story world). Figure 5.3 visualizes this tradeoff.

To be clear, a single example story will not yield in an interactive story world. Instead, a large amount of example stories demonstrating a variety of alternative

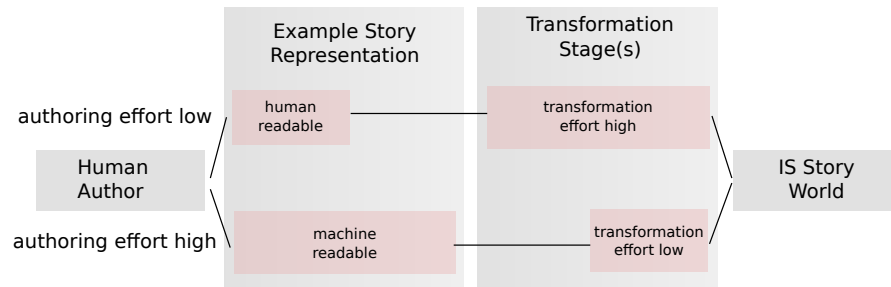


Figure 5.3: The tradeoffs between authoring effort and transformation effort when considering an example story representation

courses of events will be required for the transformation stage to produce meaningful results. But after all, this quantity aspect is why we consider involving the crowd in the first place. Regarding the question of how the transformation from example story into story world data structures works, it is impossible to give a general answer. It depends on both the example story representation and the data structure of the story world. The systems using example story input that are discussed above provide some exemplary answers to this question. Furthermore, Chapter 6 will show another concrete example of such a transformation process in the ENIGMA authoring system. Similarly, the type of information provided to crowd authors before they start producing example stories may differ. They may be given a backstory and be asked to continue it as in the authoring experiments described in this thesis, but could equally be given an existing story and be asked to create variations or extensions of it.

Human Judgement

Another way in which a crowd can contribute is by providing human judgement in certain types of systems where story worlds are partially or fully machine generated. For example in the discussion of example stories above, it is quite possible that the transformation stage (if automated) will have a certain error rate, i.e. generate some story world content that does not make any sense from a human perspective. In such cases the crowd can be used to filter out such errors by applying their common sense to review the content. This content filtering or verification is a much easier task than actual content creation, as it can be reduced to make only multiple choice decision making, i.e. vote.

Branching Story Trees

Finally, instead of relying on an IS runtime engine to generate a story, we may consider returning to the much simpler concept of branching story trees in the vein of “Chose Your Own Adventure” stories. As we have discussed in Chapter 3, branching story trees are usually dismissed by the IS research community because of the combi-

natorial explosion of branches that makes authoring a story world with more than a few branching points unmanageable. However this view is based on the assumption of a single author or small team of authors doing all the work. Crowd-based approaches remove this restriction and make branching story trees a feasible possibility for delivering certain types of fully realized IS artefacts. The CROSCAT system discussed in Chapter 7 uses this approach.

5.2.2 Finding a suitable crowd-powered approach

We are now ready to discuss which of the crowd-powered approaches (Data mining, Crowdsourcing or Commons-based peer production) is most suited for addressing the problem of the Authoring Bottleneck that the first part of this thesis has characterised.

The key question when considering Data Mining is how to obtain useful data. A repository of useful data needs to be found or an existing user activity that produces useful data needs to be identified. Data Mining works well for “Say Anything” because of its open story domain. For such a data-driven open-domain storytelling system, any story is useful data. For the more specific idea of a limited story domain however, we would need to find a repository of data that shares the limitations of the story domain. This is not easy, but possible under certain circumstances. For example, Lin and Walker (2011) data-mine a semantically annotated internet database of movie scripts. In their system the movie scripts are used to automatically learn parameters describing the language patterns of a certain type of character (e.g. an American Italian Mobster) for a natural language generation system. If a data archive as in this case can be found, Data Mining is a useful approach, with the important advantage that it does not necessitate recruitment of and advertising to contributors. However, it will be difficult to perceive general Data Mining solutions for the authoring bottleneck as any solution will be dependent on the availability of suitable data.

In Commons-based peer production situations, contributors collaborate explicitly with each other. In contrast to Crowdsourcing, they are not given specific tasks to perform. Instead each contributor gets access to the entire artefact that is collaboratively created and gets to decide on how to improve it. In Wikipedia the entire encyclopaedia and all editing discussions are publicly accessible and everyone is free to chose, which article to edit or add. Similarly in Open Source software projects the entire source code is available to anyone who wishes to inspect it and contributors have freedom in choosing the features they would like to add or the bugs they would like to fix. How well large-scale Commons-based peer production works, therefore depends on how well the jointly created product is structured. Wikipedia is inherently well structured due to it being an encyclopaedia. Using the main index and search

tools it is trivial to retrieve existing articles and discussions on a particular topic. Furthermore, Wikipedia provides many tools and practises to add further structuring such as for example support for grouping articles belonging to a particular topic, disambiguation pages or naming conventions for articles (Chernov et al., 2006). Consequently, there is a very low entry barrier to becoming a Wikipedian. In the case of software development, modularity and structure do not come as easily as they do for an encyclopaedia. Arguably, the majority of the discipline of software engineering concerns itself with how to structure and modularize code in order make it more maintainable and extendible. Thus, it is unsurprising that Baldwin and Clark (2006) found that open source projects are more successful if their codebase is well structured and exhibits a high level of modularity. If the source code of an open-source project is not well structured or documented, contributing is considerably harder and consequently, the likelihood of the project failing increases with its scale.

In order to apply Commons-based peer production to the creation of IS story worlds, a similar organizational structure for the created IS story world content needs to be found. A way is needed for collaborators to effectively take in the whole state of the story world, in order for them to pick a “module” to work on. What exactly a “module” is in the context of Interactive Storytelling is not clear. Let us consider 2 scenarios to explore this idea a bit further.

Scenario 1: If the crowd is collaboratively constructing a branching story tree, a contributor needs to review whether a specific branch is already present or not before they can add it to the tree. The Bar Karma Storymaker (see Figure 5.2) shows a very straightforward example of how such a collectively created story tree would be presented to users. The application visualizes the entire tree with scenes being visualised as nodes and temporal connections between scenes visualized as edges. Users may zoom, pan and collapse or expand nodes. The problem with this approach however is that it is not scalable, as the time it takes to review the story tree grows proportional to the number of branches in the tree. This was not a problem for the Bar Karma community as they used this format for debating the drafts for a single story board. Thus their trees did neither have nor require the breadth one would expect from a fully realized IS artefact.

Scenario 2: Commons-based peer production may also be used to collaboratively edit the data structures for an IS Runtime Engine directly. We cannot make a generalised statement on how easily such IS story world data structures can be structured in a modular way, as this depends to a large degree on the specific data structures in use. In any case, such a collaboration would have to grow organically. Just as one needs to learn programming first on a small scale before being able to contribute to large scale open source projects, contributors in such a collaboration would also

require a significant amount of training and experience using a particular IS Runtime engine. As it stands, there is currently no IS Runtime engine with a user base that is large enough to spawn a collaborative community of this sort. Thus, while we do not dismiss the possibility of this scenario entirely (e.g. with suitably user-friendly authoring tools the audience of popular IS artefacts such as Façade, Prom Week or Versu may be incentivised to become authors similarly to games that are bundled with editors and have created lively modding communities), there is not much research can currently contribute to foster such a collaboration.

Summarizing, for the purposes of IS authoring, Commons-based peer production suffers scalability problems, while the Data Mining approach gives us no control over the contents of a data source, which makes it difficult to collect data specific to a limited story domain. Crowdsourcing, however, lying between these 2 extremes, is a suitable approach for IS authoring as was already successfully demonstrated by the Restaurant, Improviso and Scheherazade systems. These systems also converge on the idea that the best way for eliciting useful contributions from crowd members is to let them provide example stories.

5.3 Crowd Task Adaptation: A Novel Process Improvement Proposal

As we have discussed, the crowd-sourced collection of example stories is currently the best (and only) accepted approach for generic crowd-based solutions to the authoring bottleneck. Consequently the work in this PhD also uses this approach as a starting point. In this section, crowd task adaptation will be put forward as a suggested improvement to the common implementation of this strategy.

The notion that there is room for improvement in the current model comes from the observation that the explicit collaboration, present in Commons-based peer production can have tremendous advantages. When explicitly collaborating, contributors actively seek out holes in the content and attempt to fill them. This is possible because every contributor can see the overall state and the individual contributions of others. In contrast, the way that crowdsourced example story collection is currently employed in IS Authoring (in the Restaurant, Improviso and Scheherazade) there is no collaboration at all. Every collaborator can only see their own contribution and is agnostic to the overall state and other people's contributions. We argue that this can inherently not be as organized and efficient as explicit collaboration. For example, imagine a Wikipedia where everyone could submit whatever they feel like writing without ever seeing what others have written. Such an encyclopaedia would be chaotic, full of duplicate and contradicting information and unmaintainable. However, we have

also already established above that explicitly collaborating (i.e. Commons-based peer production), is not a practicable solution for IS authoring, as we have no scalable solution for providing structure and ordering to the content.

There exists apparently a contradiction that collaboration is both good and bad at the same time. We propose this contradiction can be resolved by using implicit rather than explicit collaboration between contributors. The means by which implicit collaboration can be achieved is **Crowd Task Adaptation**.

5.3.1 Definition

Crowd Task Adaptation in general is a concept applicable to Crowdsourcing as a whole, i.e. broader than its application in IS authoring. We define the term as following:

Crowd Task adaptation is the property of a Crowdsourcing process, where the collector of contributions generates tasks for workers dynamically, taking into account the history of already collected results.

The collector in the above definition will usually be an automated computer system but may also be a human manually processing contributions. Figure 5.4 shows a conceptualized visualization of the crowdsourcing process.

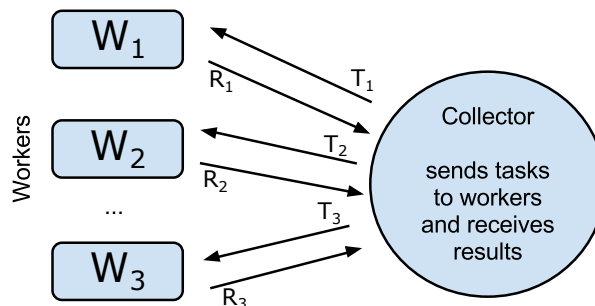


Figure 5.4: Visualization of the Crowdsourcing process

Workers (W) receive tasks (T) from the Collector and send back results (R). There is no Crowd Task Adaptation if all workers receive the same task, i.e. $T_1 = T_2 = \dots = T_n$. This is true for the Restaurant, Improviso and Scheherazade systems. The presence of Crowd Task Adaptation does not necessarily mean that all workers receive different tasks, it is possible that in several instances the adaptation process results in the same task. Thus a more accurate description of Crowd Task Adaptation is rather that a) some tasks may differ and b) it is guaranteed that some tasks will differ if they were generated after certain results have been collected.

We use formal language notation for a precise definition of Crowd Task Adaptation: Let R_{set} denote the set of all results we could possibly collect and T_{set} denote the set of

all tasks we could possibly generate. Thus $R_1, R_2 \dots R_n \in R_{set}$ and $T_1, T_2 \dots T_n \in T_{set}$. If we treat R_{set} as an alphabet, then R_{set}^* denotes the set of all possible words over this alphabet. At any given point in time, the ordered sequence of results already received by the collector will always be a member of R_{set}^* . Task Generation (t_{gen}) is a function that generates tasks from the results collected so far: $t_{gen} : R_{set}^* \rightarrow T_{set}$. Based on these definitions, **Crowd Task Adaptation is occurring if and only if the function t_{gen} is non-constant.**

Let us consider how this general concept of crowd task adaptation enables implicit collaboration between contributors in a crowdsourced IS authoring process. The basic idea is to steer contributors towards making more useful contributions by having the system assess the state of the story world contents collected so far and modify the tasks handed out accordingly.

5.3.2 Crowd Task Adaptation Strategies for IS Authoring

In the following we present a by no means exhaustive set of exemplary strategies for adapting the task of providing an example story based on the data collected so far.

Instruction Modification

In the simplest case a task consists only of a set of instructions. Li (2012) for example, reports on the following task format in the Scheherazade system:

“In an iterative trial-and-error process, instructions given to the crowd workers have been carefully tuned. We supply a few major characters and their roles in the social situation we are interested in, and ask crowd workers to describe events happening immediately before, during and immediately after the social situation.”

It is worth noting that the trial-and-error tuning of instructions in the above quote is not a form of crowd task adaptation but an initial optimization of the task. In the Scheherazade system, once the task has been finalized, it is considered a constant. Instruction modification as a crowd task adaptation strategy on the other hand would mean that instructions are dynamically assembled and changing to elicit optimal contributions. For instance, constraints may be added to or removed from the instructions that will change the possible stories a contributing author considers. This makes not much sense for the current incarnation of the Scheherazade system, which is more interested in collecting scripts of typical behaviour (Schank and Abelson, 1977) than variations of dramatic events. However for the sake of illustrating a possible use case of instruction modification, let us consider a hypothetical system,

which like the Scheherazade system collects written example stories but for dramatic purposes. Assume the instructions given to workers, ask them to tell the story of a bank robbery. If the system for example statistically determines that the majority of stories collected so far are too violent, an exemplary constraint that might be dynamically inserted by the system into the instructions might be: “Your story may not contain any deaths.”.

Back Story Modification

When eliciting dramatic stories from the crowd in a limited domain, it will often be useful to provide the contributors with a back story. The back story sets the scene, introduces characters and might contain the setup of a conflict. It fulfils the important role of communicating the limits of the domain. In many respects the back story is part of the instructions, which makes back story modification a special case of instruction modification. Modifying the back story is a simple way of steering a contributor in a desired direction. Unlike our example above, back story modification is a more natural dramatically embedded way of enforcing certain narrative constraints. Chapter 7 discusses in detail how back story modification is employed in the CROSCAT system, which was designed as part of this PhD.

Building Block Selection

If example stories are created using a specialized authoring tool software then the configuration of this tool may also be considered a part of the task. Such a system might provide a number of building blocks or primitives from which to assemble a story. Consider for example “The Restaurant”. Pairs of contributors jointly enact stories in a video-game environment. Building blocks in this case are for example the characters that users are able to play and interact with (Guest, Waitress, Barkeeper and Chef in the Restaurant), the set of actions (pick up, eat, etc.) that are available for these characters to perform and the objects that characters may interact with (plate, fork, cash register, etc.). In “The Restaurant” this set of building blocks is static, i.e. every pair of contributors enacting a story have at their disposal exactly the same set of characters, objects and actions. Systems may chose to dynamically modify this set of building blocks based on a number of heuristics. For example a system may find that after a certain amount of stories have been collected some building blocks have been overused and consequently remove them from the available selection. A less drastic approach would be to change the ordering in which building blocks are presented to the user (which depending on the authoring tool’s user interface might influence their usage). We should also consider a hypothetical authoring tool which, while relying on building blocks for assembling a story also supports the definition of

new building blocks by the user. If in such a system subsequent contributors have access to user-generated building blocks created by earlier contributors then it would also adhere to the definition of Crowd Task Adaptation. This is one of the crowd task adaptation strategies used by the Enigma system, discussed in Chapter 6.

Realtime Suggestions And Feedback

If workers use an authoring tool to construct example stories, such a tool may be equipped with the facility to enter a realtime dialogue with the worker, as they are constructing the story. Within this dialogue the system may for example suggest how to continue the story or ask the user for an immediate explanation as to why a certain event was chosen. Follow up questions may be posed by the system to gather further knowledge about the meaning of certain story events. This idea of realtime suggestions and feedback lies at the heart of the mixed initiative feature of the ENIGMA system, which will be discussed in more detail in Chapter 6.

As an illustrating example we give at this point only a very brief description of ENIGMA's use of realtime suggestions. The ENIGMA system attempts to build planning domains for FAtiMA agents from collected example stories. With every example story submission, a temporary planning domain is rebuilt by a machine learning algorithm (see Chapter 6 for details) that takes into account all stories collected so far. In ENIGMA, this temporary planning domain now forms part of the task given out to a new worker / user. While that user is creating new story, a planning algorithm using the temporary planning domain runs along in parallel. When the planner reaches the stage of executing an action it may suggest that action as a possible next event to the user. The user may discard or approve the system's suggestion. In case of approving it, the planner gains confidence that the planned action was appropriate. If the suggestion is rejected, the system will ask for clarification why the action was discarded and depending on the reason given might lower its confidence in the current plan. These confidence values are then part of the result submitted to the collector and feed back into the next round of planning domain generation.

From this example it becomes also clear why this particular form of realtime dialogue between user and system falls within our definition of Crowd Task Adaptation. As the task includes a planning domain and that planning domain is regenerated based on the results collected so far, it follows that the task is a non-constant function of the collected results, which is how we defined Crowd Task Adaptation.

5.3.3 Expected Benefits

We now discuss some of the potential benefits that we could expect from Crowd Task Adaptation when applied to IS Authoring. That is, we answer the question of

how stories collected with Crowd Task Adaptation may be more useful than stories collected without it. The difficulty of automating the processes described below in practise may vary a lot depending on the specific system used, especially on the format of the collected example stories. While we therefore can only describe the benefits of Crowd Task Adaptation in general terms here, the following two chapters discuss two concrete systems and provide more specific examples.

Waste Reduction and Increased Variety

We are interested in a wide spectrum of variation in the stories we collect. After all, we aim to transform the corpus of collected example stories into a story world that enables a fully realized IS artefact. We consider a novel story line contributed to our collected corpus more useful than a near duplicate of an already existent storyline. While duplicates can be useful in confirming that certain story lines are more obvious / popular than others, too much duplication is obviously a waste of effort. Crowd Task Adaptation can help in reducing this waste. If the system by analysing the collected corpus identifies a tendency for duplication of certain storylines among previous authors it may adapt future tasks so as to avoid or discourage the occurrence of further duplication of the same kind. It could use one of the following Crowd Task Adaptation strategies to achieve this:

- **Instruction Modification Strategy:** Changing the instructions to discourage recreating an identified duplicate story line.
- **Back Story Modification Strategy:** Insert an event into the back story that would discourage recreating an identified duplicate story line.
- **Building Block Selection Strategy:** Remove the building blocks necessary to recreate an identified duplicate story line.

Correction and Validation

Applications of crowdsourcing will always suffer from some degree of uncertainty regarding the quality of the collected data. Users may willingly sabotage the system and intentionally submit meaningless, misleading, offending or otherwise useless data. There are several strategies for filtering out such contributions made with malicious intent. Having a trusted party such as the collectors themselves manually reviewing all submitted content is often not practical, due to the scale of content to review. In some cases verification can be handled in a separate round of crowdsourcing, by having several crowd members vet a certain contribution as genuine or useful. However, this second round of crowdsourcing creates an overhead.

In the use case of crowdsourced example story collection, intentional spam is also

not the only quality criterion of concern. Users may unintentionally submit data of insufficient quality. This could for example be the case if the contributor has language problems, misunderstands the story domain's context or interprets it differently, e.g. through the lens of a different culture or has conceptual or technical problems with the story creation process.

To address these problems, Crowd Task Adaptation may be used as a self-correction facility, directly built into the crowdsourcing process itself. The validation and correction of content created by some prior contributors can become part of the task given to subsequent contributors. The *Realtime Suggestions And Feedback* strategy mentioned above is one possible way of implementing this approach. As it is used by the ENIGMA system, user feedback to system suggestions corrects mistakes in previous user's submissions not directly by modifying the data collected in the earlier submission, but by adding additional knowledge to the system that changes the way this earlier data is processed.

Homogeneity of Collected Material

The collected material should also be homogeneous. Defining the notion of homogeneity precisely is difficult, as its exact meaning is depending on the data structures used for story world representation. It may sound as though the requirement of homogeneity conflicts with the aforementioned requirement of variety. This is however not the case. Homogeneity does not mean that the collected stories should be similar to each other but that they should be in some way related to each other. A collection of unrelated stories, no matter how large, cannot be turned into a satisfying IS story world.

It would be very difficult to achieve homogeneity retrospectively. Making a story homogeneous is not primarily a case of correction and validation as described above, as there is no clear notion of wrongness associated with a story that is not heterogeneous to its story domain. Crowd Task Adaptation may however help to achieve homogeneity by influencing the author at the time of creating a story towards making it more homogeneous with the collected story corpus. Two of the strategies for crowd task adaptation that we have introduced could be used in such a way. *Backstory Modification* may be used to provide the author with a glimpse of some story content derived from prior contributions by other users. We hypothesize that the fact of having been given this glimpse will influence the author's own story to be more homogeneous. The CROSCAT system discussed in Chapter 7 uses this approach. If the homogeneity of a story can be assessed automatically, the *Realtime Suggestions And Feedback* strategy is also suitable to influence the user as they are telling the story, for example by discouraging the use of certain events, when the passing of a certain heterogeneity threshold is detected. We hypothesize that the mixed initiative

feature in the ENIGMA system (discussed in the next Chapter) also leads to more homogeneity.

5.4 Conclusion

This chapter has looked at different ways in which the creative task of story writing and in particular the authoring of interactive stories can be distributed among members of an online crowd. It came to the conclusion that a crowdsourcing approach where a central authority is in charge of defining the skeleton and boundaries of a story world and online contributors are tasked with providing example stories set in this story world is the most suitable approach to tackling the authoring bottleneck. A generic extension to this workflow termed Crowd Task Adaptation was suggested, wherein the automated story collection system continually analyses its repertoire of results obtained so far and intelligently adopts the tasks handed out to contributors accordingly, so as to maximise their utility for the authoring effort. This chapter has discussed some ideas of how this abstract idea could be realized. The next two chapters follow up on these ideas with two concrete examples of systems implemented within this PhD that realize Crowd Task Adaptation in two very different ways.

Chapter 6

The ENIGMA Authoring System

In this chapter a system called ENIGMA for crowd-sourced authoring of IS systems is discussed. The idea for this system was motivated by the experience of building IS artefacts based on the FAtiMA agent technology. Chapter 2 has discussed the problems encountered while authoring FAtiMA agents for the FearNot! project. ENIGMA was an attempt to create an authoring tool for FAtiMA that addresses these shortcomings through the means of crowd-sourced authoring. The system is designed around the principle of crowd task adaptation described in the previous chapter. The results of a small scale usability trial are given that suggest that the approach taken by ENIGMA was too ambitious to realize within the scope of a PhD project. Nevertheless this chapter summarizes the work in designing and implementing a first prototype of this system and the lessons learned.

6.1 Design

6.1.1 Overview

ENIGMA was planned as an authoring tool for the FAtiMA architecture. Its chief requirement thus was to lighten the authorial complexity of creating FAtiMA agents that was illustrated through our case study in Chapter 2. As we have presented in that case study there are many aspects to creating an actual IS artefact based on FAtiMA agents. ENIGMA was not intended to initially address all of these issues but instead was always firmly targeting the authoring of the FAtiMA layer, i.e. the planning domain encoding the agent’s decision making processes. It is for example not meant to address the creation of graphics, animations, etc. In real world usage, ENIGMA would be one of many tools in a content production pipeline.

Figure 6.1 gives an overview of the ENIGMA system architecture. ENIGMA follows the crowd-sourced story collection approach discussed in the last chapter. A

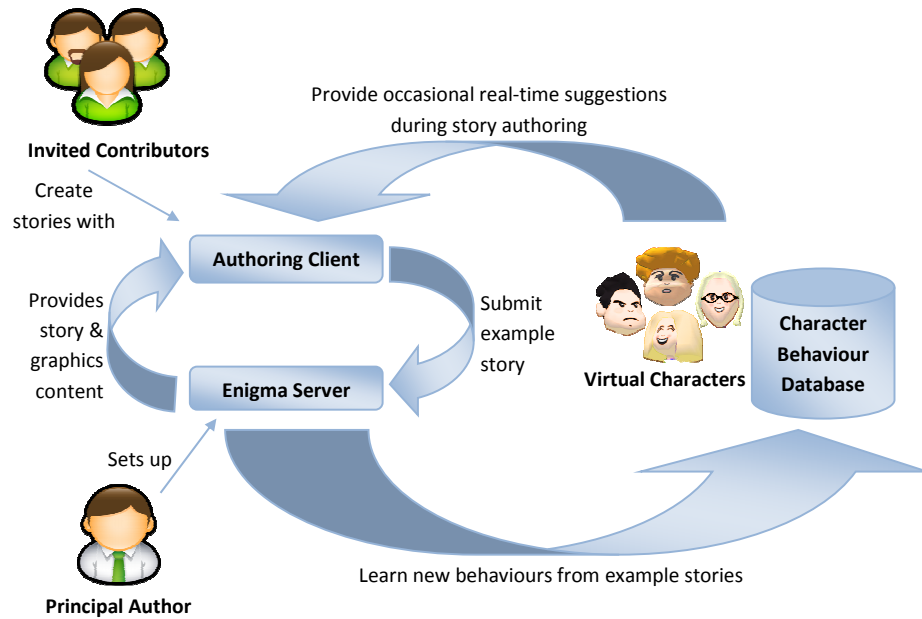


Figure 6.1: Enigma Basic Workflow

client application which can be run directly from the browser allows contributors, who are invited by a principal author (PA) to create a story within given boundaries, which are set by the PA. Those boundaries include a fixed cast of characters, set of props and scenes and authoring instructions regarding the theme of the story world, back stories of characters, etc. Every story that gets created within the client application is submitted to a server where many of these stories are collected and processed by machine learning algorithms to generate the planning domains and personality configuration for FAtiMA agents that can be used as virtual actors within an interactive drama. An important aspect of the design of ENIGMA is the mixed-initiative mode, in which characters are giving occasional realtime suggestions to the user. As we have shown such a built in feedback loop is an example of crowd task adaptation. The design of the ENIGMA system went through several iterations until it eventually arrived at the state that is described in this chapter. This evolution can be traced through the few publications describing the plans for the ENIGMA system. The initial design plans are described by Kriegel and Aylett (2007) and Kriegel et al. (2007). In those early design stages crowdsourcing was not yet a central concern. The vision at that stage was of an authoring tool that supports an individual author through the means of providing example stories. I.e. no distinction between principal author and contributors was made at this point. The realtime feedback was seen as a way for the author to assess the state of learned character behaviour, i.e. a window into the character's mind, as it has emerged from the learned example stories up to this point. Kriegel and Aylett (2008) established the connection between the ENIGMA architecture and the use of crowdsourcing. From that point on ENIGMA

was treated primarily as a platform for crowdsourced authoring. Finally, Kriegel and Aylett (2010) give an up-to-date description of the ENIGMA system. It describes more concrete implementation details and is the only of the four publications that mentions the name ENIGMA.

6.1.2 Storytelling Interface

A central design question for ENIGMA was how the user creates stories. On the one hand the process should be as user friendly as possible, on the other hand there must exist a way for the stories to be processed by machine learning algorithms into FAtiMA planning domains. A possible answer to this question, might be found in the user interaction methods in IS artefacts. After all, users interacting with IS artefacts are also acting out a part of a story, using user-friendly interfaces and their actions are often successfully interpreted by a machine.

As we have discussed in Chapter 2, an IS artefacts based on FAtiMA, will usually contain a story presentation layer on top of FAtiMA that renders the FAtiMA event syntax into a human understandable format. For example, in FearNot!(Aylett et al., 2007) an action executed by a FAtiMA agent is translated into an animation performed by the 3D character representation of this agent. As demonstrated by the left side of Figure 6.2, user interaction also passes through the story presentation layer. For example in FearNot! users interact via natural language text input that is translated by a language parser into a discrete FAtiMA event. In ORIENT (Kriegel et al., 2008), another IS artefact built on top of FAtiMA, user actions are performed using a combination of multi-modal interaction modalities (Stepping on a dance mat, performing gestures using a WiiMote Controller and scanning RFID tags using a mobile phone), which are translated into FAtiMA syntax by the application behind the scenes.

These IS artefacts can allow user interaction to be well integrated with the presentation layer, because the user is restricted to a very limited set of actions. Hand-crafted code exists that translates each possible user action taking into account the context in which the action was performed into its corresponding discrete FAtiMA event syntax. As we do not want to restrict a user's freedom in telling a story with ENIGMA by limiting the set of user actions, this model becomes infeasible. The problem of creating user interfaces that allow acting out the actions of an avatar within a virtual world freely without any restrictions is a well known challenge of Interactive Storytelling Research (Mehta et al., 2010). Systems like FearNot! and Façade (Mateas and Stern, 2003), use natural language input to hide the fact that the user is really limited to perform a restricted, finite set of actions.

We therefore decided to design a storytelling interface for ENIGMA that lets

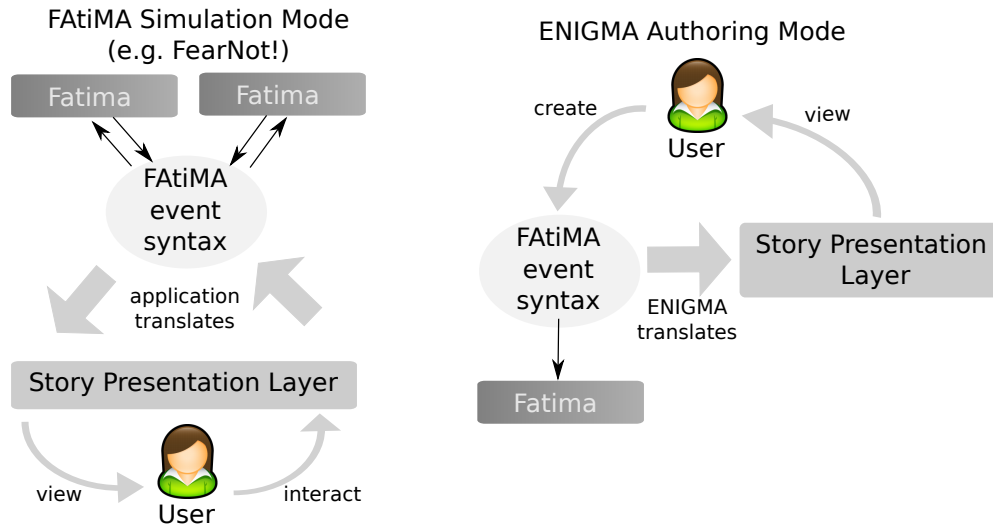


Figure 6.2: Comparison of the user's role in a FAtiMA based IS artefact and in ENIGMA.

the user specify stories directly using the FAtiMA event syntax. This means stories created with ENIGMA will contain discrete events consisting of subject, action and action parameters that are readily understandable by a FAtiMA agent. Nevertheless, a story presentation layer was included to display the created events to the user. This is possible because the translation from FAtiMA event syntax to Story Presentation Layer is achieved much easier than the reverse direction. We hope that the presence of a presentation layer makes the application more user friendly and provides the user with the ability to review the story they have created. The right half of Figure 6.2 illustrates our approach.

Event Specification

The FAtiMA event syntax is very simple. Each event consists of a location, subject, action and a number of (action dependent) action parameters (e.g. *John GiveGift(Luke, Ball) @ Playground*). The actions referred to in events are always instantiated, i.e. all action parameter variables are bound to concrete values. ENIGMA can support the author in the process of specifying events by allowing them to chose values from a pre-filtered selection of choices. ENIGMA stores additional metadata for actions to enable and simplify this selection process, as shown in Figure 6.3.

Figure 6.4 shows the aforementioned GiveGift action as an example of such an action definition.

The descriptions are not required by FAtiMA and were added to ENIGMA purely for the reason of assisting the user in the process of specifying events by means of instantiating actions. After all a short action or parameter name will not always be able to convey the specific meaning that the execution of this action carries. Simi-

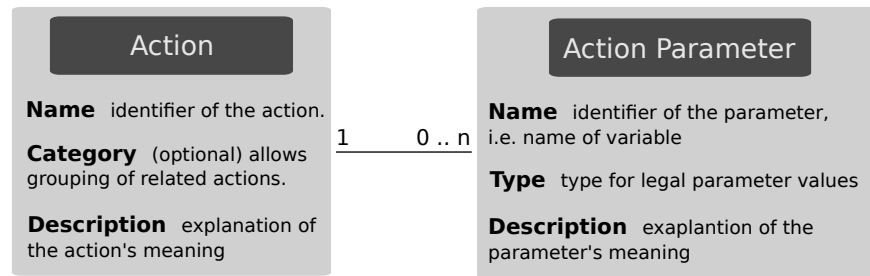


Figure 6.3: Metadata associated with an action in ENIGMA

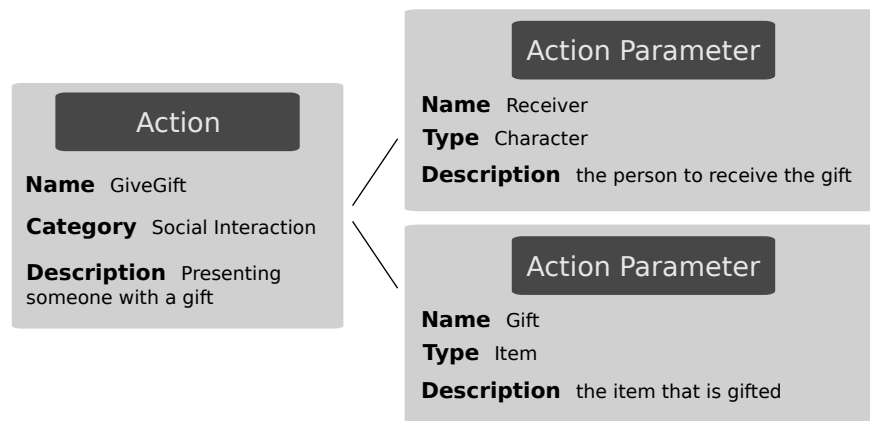


Figure 6.4: Metadata associated with the GiveGift action

larily the only function of action categories is to help users in finding a certain kind of action. The type for parameters however does more than help in filtering the values available for selection when instantiating actions. The type of parameters will also find its way into the FAtiMA planning domain as action preconditions enforcing the type during planning in absence of a real type system in FAtiMA. The GiveGift action demonstrated in Figure 6.4 for example would automatically be equipped with the preconditions *isCharacter*(*[Receiver]*) and *isItem*(*[Gift]*). The type system has a few built in types like characters and items and can be extended with further story world specific types.

Adding New Content

Users are allowed to define new actions from within the ENIGMA application. The definition of a new action amounts to filling out a schema as in Figure 6.4. While there is quite a bit of overhead involved in specifying all the metadata necessary for an action description, that action will from then on be part of the story's repertoire of story building blocks. Subsequent authors will have access to it and reuse it in different contexts. As previously mentioned, this transfer of user-generated story building

blocks between contributors also amounts to a form of crowd task adaptation, as the authoring tool's configuration and thus the task continually changes for subsequent authors. Besides actions, new types may also be manually specified by the user.

One might criticise the requirement of having to specify actions as too cumbersome but the alternative of only being able to choose from a predefined set of actions would be severely limiting the user's ability to exercise their creativity when creating a story.

Bootstrapping

ENIGMA's design requires the principal author to preload the authoring system with the characters, locations and items that will be available to the authors. The set of these entities is one of the means for defining the bounds of the story domain, within which all collected stories should fall. Additionally, the principal author is also encouraged but not required to provide an initial set of relevant types and actions. The advantages of providing an initial set of existing actions and types are that they make life easier for the first few participants, who would otherwise have to specify a new action for virtually every event. Furthermore they provide essential examples for how to structure actions and types and thus make it easier for authors to add additional content.

Story Presentation Layer

The story presentation layer displays the story events in a more human-friendly form than the FAtiMA event syntax in which the story is authored. It involves graphical and textual representation of events.

Graphical Display Initially ENIGMA was designed to use a 3D virtual environment presentation layer. Figure 6.5 shows an early prototype of the application's interface.

This interface was initially intended to match the presentation layer present in FearNot!. Also, especially before its move towards crowdsourcing, ENIGMA's early design was influenced by theatre metaphors, with the author taking the role of a director performing multiple rehearsals (i.e. example stories) with a cast of characters. However, we soon abandoned the visualization through animated 3D graphics and instead decided for a comics based graphical presentation layer, similar to the Story Canvas system (Skorupski and Mateas, 2010). Figure 6.6 shows an example of the style that was eventually adopted.

There were several reasons why the comics interface was eventually preferred over 3D animation. Using comics allows us to have a thin authoring client that can be easily distributed independent of platform. 3D applications on the other hand are typically much thicker applications with more technologically challenging deployment

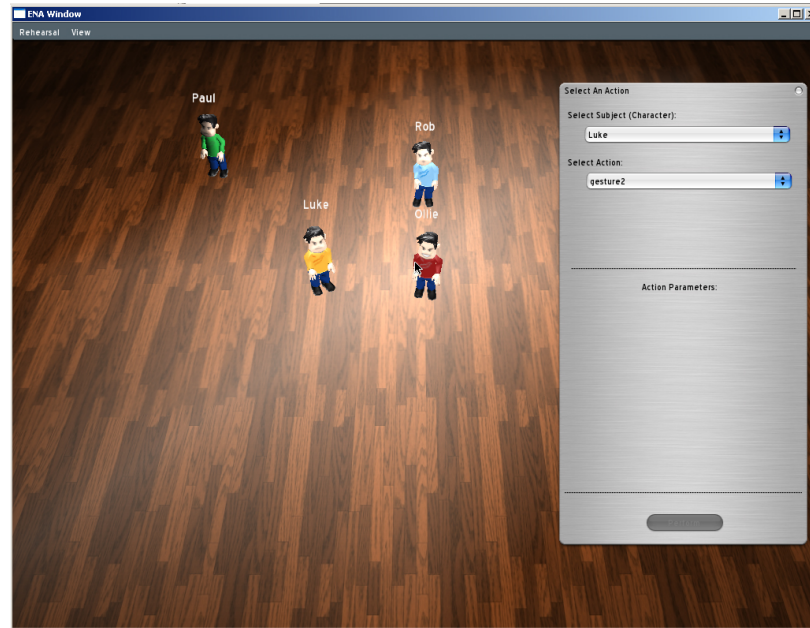


Figure 6.5: Screenshot of initial ENIGMA prototype using a 3D game engine



Figure 6.6: Example of 2 frames using ENIGMA’s comics story presentation layer

issues. This is an important consideration for crowdsourcing applications as we ideally don’t want to exclude any potential participants on the basis of their choice of operating system. Also, we need to represent event sequences within the authoring tool. With the way they are structured (1 panel = 1 event), comics provide a better mechanism for doing that than animations. Finally this form of visualisation allows for uncomplicated graphics content generation independent of any specific tools (characters are represented by a series of annotated head and body pictures, scenes are simply panoramic pictures). This means less work for the principal author when setting up an initial story domain.

In either graphical presentation style, the principal author can provide the system with a list of mappings from actions to character graphics / animations and emotional states to character head graphics / facial animations.

Natural Language Natural language is used in two different ways to augment the representation of story events, dialogue and narration. Any action in the library of available actions can be optionally marked as a dialogue action. This associates the action with a list of alternative dialogue lines, one of which is chosen at random whenever an instantiation of the action is displayed by the story presentation layer. Supporting a one to many mapping from action to dialogue can make the story representation seem less repetitive if an action is instantiated multiple times. Variables referring to the action's parameters and the subject may be used in the dialogue lines. If for example the GiveGift action from Figure 6.4 was marked as a dialogue action, it could be associated with some dialogue lines such as "Here [receiver], I want you to have this [Gift]." or "You can have this [Gift], [Receiver]". When the action is instantiated (e.g. GiveGift(Luke, Ball)) the variables are replaced (e.g. "You can have this ball, Luke."). The dialogue line is presented as a speech bubble attached to the event subject character by the comics system (see right panel in Figure 6.6). In our initially planned 3d animated graphical presentation it would have been uttered by the event subject character in realtime using text to speech technology and additionally displayed as subtitles.

Narration text may optionally be used to describe actions. It is technically handled in the same way as dialogue text, but is presented differently (in the comics presentation as a narration box as in the left panel in Figure 6.6 and in the initial 3d prototype as a subtitle). The main envisaged use of narration text is as a place-holder for character graphics or animations that represent the action graphically. For example an action such as Kick([target]) may have a narration text like "[subject] kicks [target].".

Especially new user-defined actions, i.e. those not in the initial action library created by the principal author, will benefit from this. Extending the authoring tool to let users define their own graphics or even animations is impractical and would distract too much from the application's main purpose of storytelling. Therefore whenever a user defines a new action, ENIGMA allows them to define a narration text and / or dialogues. In order to simplify the process for users ENIGMA does not require users to define narration or dialogue text using variables. Instead if there are any variables used they are inferred from the action's first instantiation.

When story collection is complete and an IS artefact is built using the FAtiMA agent configurations constructed by ENIGMA, the principal author may or may not use the dialogue and narration texts created by the authors. In other words, collecting this textual content is not the primary outcome of using ENIGMA, but it might be a useful secondary side effect.

6.1.3 Annotations

The stories created as described above consist of event traces describing the actions of characters. However, they don't capture necessarily everything that happens in the story, nor do they in themselves provide enough information to infer the contents of a FAtiMA planning domain. Therefore ENIGMA's design includes the functionality for annotating stories with additional metadata.

One might be able to use the corpus of unannotated event traces collected by ENIGMA to drive agents with a case-based reasoning architecture (which FAtiMA is not), similarly to the approach taken by Orkin and Roy (2009) when automating agent behaviour based on the "Restaurant" corpus. But even in such a case-based reasoning architecture, the value of unannotated cases is limited. In later work Orkin et al. (2010) describes a system for annotating the collected Restaurant corpus with tasks in order to improve automated behaviour.

ENIGMA lets the user create several different types of annotations for stories, which are briefly described below.

State Changes

Users may specify how certain events change the state of the story environment (this includes the states of characters, items, etc.). State changes are expressed as property changes of entities, like characters or items. Properties are arbitrary name-value pairs. Property values have types just as action parameters and can thus be comfortably selected using the same type system support. An initial set of properties may be provided by the principal author but users can also define new properties. A possible state change for our exemplary event *John GiveGift(Luke, Ball) @ Playground* could be for example the property owner of the ball item changing to Luke. ENIGMA does not visualize state changes in the story presentation layer, but it might be possible to do so (for example as additional comics panels).

Emotions

When an event affects a character's emotions this is a special kind of state change and treated in a slightly different way from generic state changes. This is firstly, because emotions in FAtiMA are represented different from properties and influence an agents behaviour differently. Secondly, visualizing emotions in the presentation layer is straightforward, requires relatively little and finite graphical materials and adds a lot of flavour to the story presentation. Thus, if users in ENIGMA specify that a certain event has made a character e.g. happier, sadder, angrier or more fearful this will be reflected by a different facial expression.

Goals

Goals are an essential part of a FAtiMA planning domain but none of the information we have described collecting so far can help us in inferring an agent's goals. It is therefore necessary for ENIGMA to support goal annotation if we intend to construct complete FAtiMA planning domains. Goal annotation is performed once the story has been completed and works as follows: The author first selects a goal's name from the library of existing goals (or creates a new one) and the character, whose goal is being specified. Next the author selects all events that are part of the character's plan to achieve the goal. Finally the author specifies whether the plan has succeeded, failed or remains unachieved. This can be repeated multiple times until the author judges the annotation to be complete. If a contributor should fail to perform the goal annotation, a third party (e.g. the principal author) could annotate the goals in their submitted story at a later point.

6.1.4 Mixed Initiative

Mixed initiative planning refers to a planning software that is supervised and assisted by a human being. In the case of ENIGMA as it was originally envisaged, it refers to the author working alongside a planner for each character, which determines that character's behaviour. In line with the theatre metaphor, which permeated ENIGMA's early design, the author is thought of as a director, who controls the characters not as lifeless puppets but rather as autonomous and intelligent actors, who plan their own actions. When a character's planner wants to perform an action it can suggest its execution to the author / director. The author can override the character's suggestions if he does not agree with them. A dialogue between the author and character might ensue in which the author, explains to the character why the suggestion was not accepted. These explanations feed into the next round of planning domain generation as additional metadata.

This mixed initiative mode as it was originally intended should provide 2 core benefits: Firstly it will provide the author with immediate feedback of a character's authored personality so far and thus make it easier for them to "debug" a character and correct parts of its personality. Secondly, especially after multiple stories have been collected, authors will be relieved from the burden of giving the characters repeated instructions. With every additional collected story a character will become more active and autonomous and the author can focus on the input of new knowledge rather than repeating knowledge the character already has.

When the ENIGMA project's focus was extended to crowdsourcing, another benefit of the mixed initiative mode became apparent. As we have already briefly touched on in the last chapter, ENIGMA's mixed initiative mode with its possibility of giv-

ing realtime suggestions and feedback is another Crowd Task Adaptation strategy. Thus, for any contributing author using ENIGMA, traces of previous authors' work can not only be found in the user-generated story building blocks available to them (actions, goals, properties and types), but also in the realtime suggestions made by the characters, which represent the accumulated knowledge and behaviour, which the characters are equipped with. As we have argued in the last chapter, there is good reason to believe that such a type of Crowd Task Adaptation leads to a higher quality of generated planning domain, by providing some sense of coherence and connection between otherwise disjointed example stories.

Suggestion and Feedback Format

Suggestions should not be made too frequently but only occasionally so as to not annoy the user. The acceptable frequency will also depend on the degree of subtlety with which the suggestion is presented. If its incorporation into the authoring tool is very invasive then a low suggestion frequency is advisable, whereas subtle, easier to ignore, suggestions that stay in the background might be more frequent. A real-time suggestion by the system can take one of several forms:

- **Story Continuation:** - Suggest the next event for the currently created story
- **Property Change Annotation:** - Suggest a property to change as the result of the current event.
- **Emotion Change Annotation:** - Suggest a change to a character's emotional state as the result of the current event.

Feedback may be gathered from the author in response to these suggestions using a simple multiple choice format, providing the following options:

- **Accept:** - The author accepts the suggested event / change. Increases the system's confidence in its reasoning that lead to the suggestion.
- **Reject - Illogical:** - The author rejects the suggested event / change on the grounds of it making no sense. Decreases the system's confidence that the reasoning the suggestion was based on is sound.
- **Reject - Not For Me:** - The author rejects the suggested event / change on the grounds of it not fitting into the story they want to tell. As this option can (especially in presence of the illogical option) be interpreted as a mild acknowledgement that the suggestion itself is sound, this option may slightly increase the system's confidence in the reasoning associated with the suggestion.

- **Leave me alone:** - The author does not even want to think about whether this suggestion is fitting or not. The system will draw no conclusions whatsoever from this feedback, except possibly to lower its frequency of making suggestions.

Section 6.3 will provide more details of how the mixed initiative mode is used in the system's learning process.

6.2 Implementation

This section briefly summarizes the final implementation of the ENIGMA design, which was described above. As we have already mentioned in the introduction, during the course of this PhD, ENIGMA was eventually abandoned, or better radically redesigned as the CROSCAT system, which is the topic of the next chapter. As a result, the implementation of the design outlined above is not complete. While the story specification, annotation and story presentation layers are complete and a fully functional client-server architecture for collecting stories is in place, neither the learning of FAtiMA planning domains (see Section 6.3) nor the mixed initiative feature (see section 6.1.4) were ever implemented. This is because the decision to take a different approach for this PhD was already made after a first usability study of the authoring tool interface (see Section 6.4), so there was never any need to implement the outstanding features.

6.2.1 Technology Overview

We have decided to implement both the ENIGMA server and client using the Java programming language, as it allows platform independent deployment. The authoring tool user interface uses the Java Swing GUI library. With hindsight a purely web based authoring client might have been more appropriate, but at the time when this implementation decision was made, Swing was still very popular and complex interactive user interfaces in HTML and JavaScript were more difficult to implement than nowadays. The ENIGMA client and server communicate using RMI (remote method invocation), Java's default implementation of remote procedure calls. The ENIGMA server machine also runs an instance of a comics generation system (Alves et al., 2008) that communicates with the ENIGMA server via a custom TCP protocol. All comics images are generated on the server side. When the client requires a new comics image, it calls a remote method on the server that invokes the comics system. The comics system then outputs the comics frame as an SVG vector graphic, which the server rasterises into a jpg image, which is then sent back to the client.

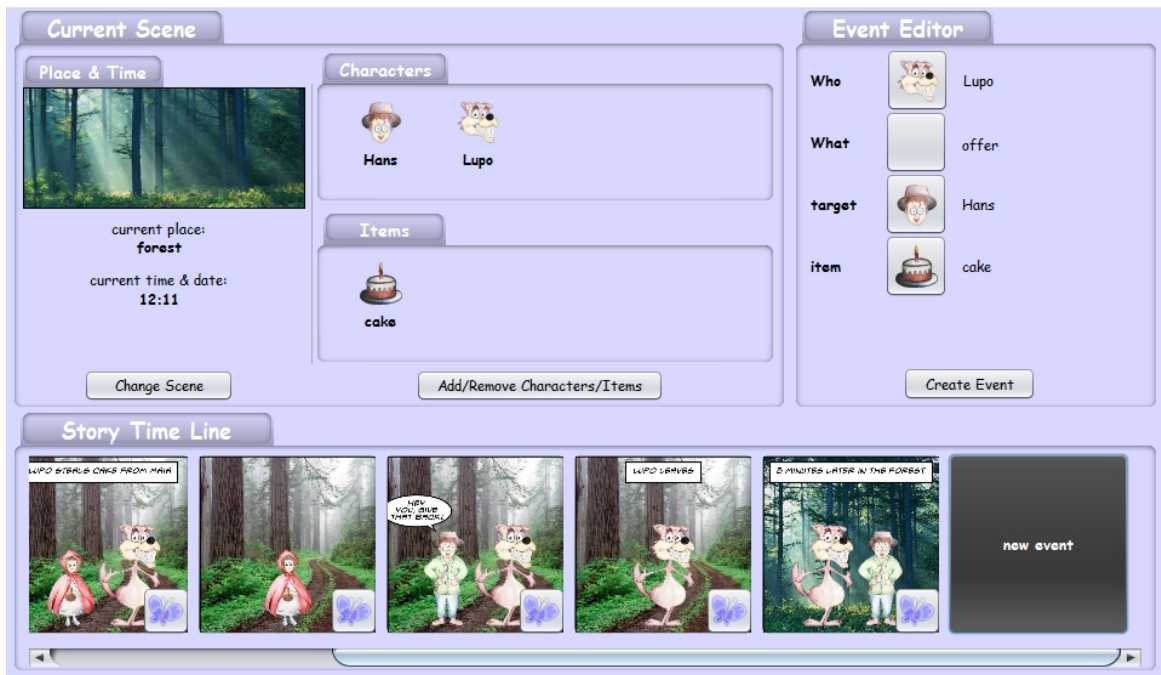


Figure 6.7: ENIGMA's user interface: main window

6.2.2 User Interface

Figure 6.7 shows the main window of ENIGMA's user interface, which is structured into 3 parts. The story time line lets the user review the story they have created so far. Comic panels can be enlarged by clicking on them. The current scene panel displays the context of the current scene (time and place, characters and items present) and allows the author to modify it. When the author modifies the context, an event visualizing the change will automatically be created. For example “Luke enters the house”, “Meanwhile on the playground” or “A few hours later..”. Finally, the event editor lets the author add new events. The parameter slots (target and item in Figure 6.7) are made visible and labelled depending on the chosen action's parameters.

Instantiating Events

Figure 6.8 shows ENIGMA's user interface for selecting an action. The dialog window on the left shows the listing of available actions sorted by category in a tree control. An icon for each action indicates whether it is a dialogue or physical action. If the action that an author wants their character to perform is not in the list they may define a new action. This will initially bring up the action creation wizard shown in the middle of Figure 6.8. In order to simplify the action creation process for users, the wizard makes a few simplifications to ENIGMA's action model for the sake of a simpler action definition process. It allows only the creation of actions that have no more than two parameters: a character (the target of the action) and / or an item (the instrument of the action). From our analysis of the FearNot! corpus, we could see



Figure 6.8: ENIGMA’s user interface for choosing an action (left), defining a new action using the action wizard (top right) and defining a new action in expert mode (bottom right)

that the majority of actions that authors create typically adhere to this restriction. The GiveGift action which was cited as an example throughout this chapter does so too. As a result, when using the action wizard, the user does not have to bother about defining action parameters and choosing their types. If an expert user wants more control they make create an action in expert mode (see right of Figure 6.8), in which case the user has full control over the number, names and types of parameters.

Change Annotations

Whenever a user creates a new event in ENIGMA, the application will ask if the event caused any changes in state or emotions. Answering this question in the affirmative will cause the change annotation dialogue to appear (see left window in Figure 6.9). The changes caused by an event can also be edited retrospectively by pressing the butterfly icon placed on each event panel in the ENIGMA main window’s story time line (see Figure 6.7). The change annotation dialog window allows defining new



Figure 6.9: ENIGMA's user interface for annotating changes after an event (top), specifically state changes (bottom left) and emotional changes (bottom right)

property and/or emotion changes and lists all changes already associated with the event. The middle of Figure 6.9 shows the dialog window for defining state / property changes. The dialog window for defining emotion changes is shown on the right side of Figure 6.9. A selection of emotions directly corresponding to all the non-prospect based emotions of the OCC model ((Ortony et al., 1988)) used in FAtiMA is presented to the user.

Change annotations are necessarily interpreted as postconditions and never as preconditions. While for the purpose of deducing FAtiMA operators, a separate annotation mechanism for preconditions (perhaps presented along the lines of "why did this event happen") would have been useful, this was not added in order to avoid overcomplicating the author's task.

Goal Annotations

Finally, Figure 6.10 shows ENIGMA's dialog window for annotating goals. Goals are selected (and possibly created in the top left section of the dialog window). The arrow button labelled "use in this story" instantiates a goal, i.e. it opens up a dialogue, in which the owner of the goal and any goal parameters are chosen. This goal

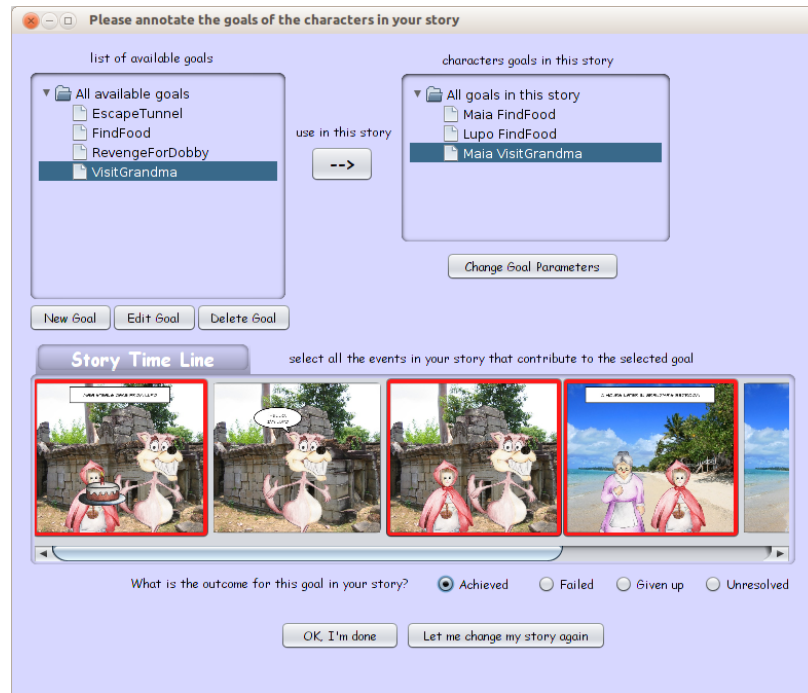


Figure 6.10: ENIGMA's user interface for annotating goals

instantiation dialogue is the goal equivalent to the event editor panel on ENIGMA's main window (see Figure 6.7). Instantiated goals are listed in the top right section of the goal annotation dialog window. The currently selected goal from the list of instantiated goals can be annotated using the bottom half of the dialog window. The user marks all events that are part of the character's plan to achieve the selected goal by simply clicking on them. Contributing events are highlighted using a red frame.

6.3 From Stories to FAtiMA Agents

In this section we outline a procedure by which ENIGMA may transform the collected corpus of stories (ordered series of annotated events) into planning domains and personality configurations for FAtiMA agents. As mentioned before, ENIGMA was never completed as originally intended and in particular the learning functionality discussed here was never implemented. Low level implementation details are omitted for the simple reason that there is no implementation. The design presented here quite possibly contains some flaws that would become apparent in the implementation of the model. Therefore this section is not intended to endorse the presented method as the most efficient, effective or suitable way of solving this category of problem. It is mainly included for the sake of giving a high level overview of how the ENIGMA learning process was intended to be realized, as this is arguably the most intriguing part of the system from a computer science perspective. This section should also prove that despite focusing the direction of this research away from ENIGMA, there

existed a concept of moving forward with the ENIGMA system.

6.3.1 Formalization as a machine learning problem

Mitchell (1997) characterises a formalised machine learning problem as consisting of a task T , a performance measure P , some training experience E and a target function V . The problem of deriving FAtiMA agent configurations from the data collected by ENIGMA could be described as follows:

- **Task T** - Generate a representational model for deliberative and reactive behaviour of FAtiMA agents. In the following we will call this the *Domain Model*.
- **Performance Measure P** - We judge the quality of a derived Domain Model, by how well it enables a set of FAtiMA agents through their interactions to jointly create an emergent narrative that satisfies the criteria we defined for fully realized interactive stories (See Section 1.1.3). In particular the generated Domain Model should allow the agents to exhibit believable dramatic behaviour.
- **Training Experience E** - The system learns primarily from a set of annotated example stories. Furthermore it is able to make realtime suggestions to authors as they generate stories and learn from their responses.
- **Target Function V** - One could take the view that the system aims to learn about the suitability/quality of candidate domains. In this case, one may think of the target function V as mapping domain models to a numeric score.

The learning task described above is highly atypical and to some degree exhibits elements of various machine learning styles. The basic task of inferring planning domain from example stories has a supervised element but is not real **Supervised Learning**. While the example stories could be considered examples of “good stories” provided by a supervisor, this is not the type of classification task one would normally associate with supervised learning. This is also the reason, why the above definition of target function V is rather vague. A more classical supervised learning problem would be the classification of stories, e.g. being able to recognize “good” stories based on the positive examples (collected through crowdsourcing). ENIGMA’s learning task also has some similarities with the discipline of **Case-Based Reasoning** (CBR). However, ENIGMA does not use the set of collected cases themselves to drive its reasoning and instead uses them to derive a domain model for an external reasoning engine (FAtiMA). In other words, the example cases and the reasoning are too decoupled from each other to qualify as traditional CBR. Finally, the mixed initiative mode also introduces elements of **Active Learning** as the algorithm itself may decide when to provide suggestions to the user. Given this mix of properties of the learning task

at hand, it may not come as a surprise that the literature does not seem to offer a template solution to this unique task. In the following we describe one possible way to approach it.

6.3.2 The probabilistic domain model

As described above, conceptually we want the system to associate a score to each possible domain model. But the number of possible models is if not infinite then at least inconceivably large. Clearly a holistic approach of generating and comparing many entire candidate domains is not very practicable. Fortunately it is also not necessary if we introduce a probabilistic extension to the FAtiMA domain model. A regular FAtiMA domain model (i.e. one that can be used by actual FAtiMA agents) contains a set of actions, each with preconditions and effects, goals, with various conditions, emotional reaction rules and action tendencies (see Section 2.2). A probabilistic domain model contains exactly the same types of elements but instead of only one may store a number of alternative versions for each of these elements with an associated probability¹.

The probability reflects the system’s confidence in the element being “correct”. Rather than associating probabilities with entire domain models, they are only associated with individual sub-elements, thus greatly reducing the amount of data necessary for keeping track of generated candidate domains. For example the probabilistic domain model might contain two versions of the action `Hit([target])`, one with the effect “[target](hurt)” and an alternative version with the effect “![target](hungry)”. It may have deduced the latter, e.g. from a story in which a character jokingly says “Now, I’m no longer hungry” after being punched. The system may learn in due time that the former is probably a more useful representation of the action. This would be reflected in the probabilities associated with the 2 versions of the action.

6.3.3 The ENIGMA learning cycle

The probabilistic domain model sits at the heart of ENIGMA’s learning process as illustrated in Figure 6.11. The server manages a single version of such a model, which gets updated whenever a client submits a new story. The most probable domain model is then extracted from the updated probabilistic domain model, by simply choosing the version of each event with the highest probability. When a new client logs in, the server launches a set of FAtiMA agents that are configured according to the most probable domain model. These agent instances run in the background on the server side while the user is constructing a story in the authoring client. Story events that

¹Emotional Reaction Rules are an exception. We do not maintain alternative versions with associated probabilities for these. Section 6.3.4 explains why.

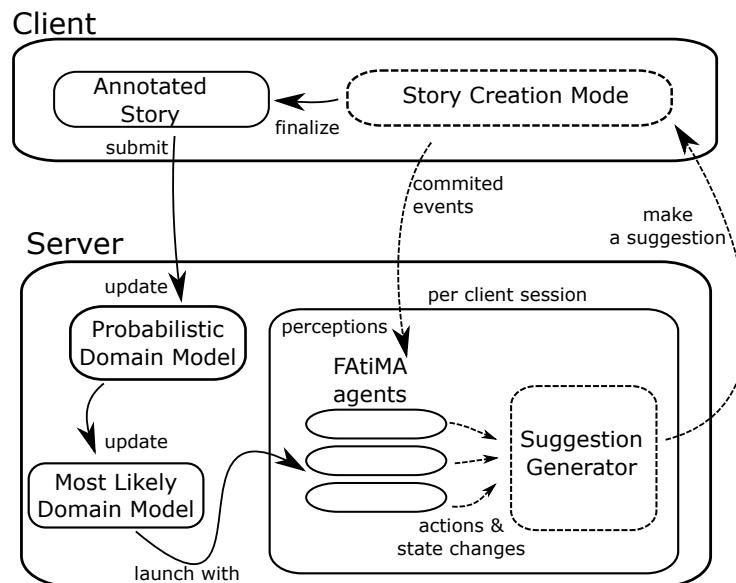


Figure 6.11: Overview of the FAtiMA domain model learning approach for ENIGMA

the user creates in the client storytelling application are routed as perceptions to the agents.

Actions performed by these agents and state changes within them are captured by the suggestion generator component, which may turn them into realtime suggestions. While an action would be turned into a suggestion for story continuation, a state change would be translated into a property or emotion change suggestion. The suggestion generator fulfils 2 functions: Firstly it ensures that the user is not overburdened with too frequent suggestions (see discussion in Section 6.1.4). Secondly it tries to make suggestions that maximise the system's ability to decrease its uncertainty. It does this by preferring to make suggestions that lead to new knowledge about domain elements for which the most probable version is uncertain (i.e. where probabilities for the top two items are very close). User feedback (i.e. responses to the suggestions) are stored on the client side and are sent together with the finalized annotated story to the server, where the cycle is completed and the process starts once again with an update to the probabilistic domain model.

If the above diagram conjures the idea of strictly serial processing, it should be pointed out that this is only a simplification of the illustration. The ENIGMA learning approach supports multiple clients logged in at the same point. As the server maintains a set of separate FAtiMA agent processes, a copy of the associated domain and a separate instance of the suggestion generator per client session, there is no risk of interference between multiple clients. Also the scenario where client A submits a story, while client B is in the middle of a storytelling session is covered by this per-session isolation. The story submitted by A will lead to an update of the probabilistic domain

model and consequently give rise to a new most probable domain model, which will be available for new client sessions launched beyond this point. But this will not affect client B, whose session is already in progress and operates on a copy of the previous most probable domain model.

The underlying hypothesis of this approach is that the quality of the “most probable domain model” should constantly improve, and eventually converge towards a best solution that optimally represents all the provided example stories. The fact that the set of primitives / story building blocks is not necessarily constant, may work against this idea of continual improvement. Recall that users of the authoring client may define new actions, goals and types. When such elements are added, the probabilistic domain model will initially contain no suitable representations for them.

For example when the 21st user defines a new action “x”, the system does not initially know much about “x”, whereas it may have encountered all other previously defined actions multiple times already. It is clear that the initially derived representation of “x”, which first appeared in version 21 of the probabilistic domain model is not as much based on group consensus as that for some other actions. Therefore one might take the view that version 20 had converged further towards the perfect solution representing all stories than version 21.

But viewing this as a decrease in quality is misleading. Previous versions of the most probable domain model were not aware of the added elements at all and are not representing them any better. The fact that they did not need to is irrelevant as the measure of quality for a given domain model is how well it represents all stories. As there now is a story containing new elements all previous best solutions would find it hard to match those stories. In conclusion, even though the pool of domain building blocks is not constant, we can expect our approach to produce a continually improving “most probable domain model”.

6.3.4 Updating the probabilistic domain model

Initialization

The initial probabilistic domain model (i.e. the model that is used when the learning cycle illustrated in Figure 6.11 is first set up and no example story has been collected yet) is empty and contains no emotional reactions, action tendencies, actions or goals. It is however initialised with a set of personality parameters (e.g. emotion thresholds and decay rates) for every character as learning these from example stories is currently outside the scope of the ENIGMA tool. The initial values for these parameters may be provided by the principal author.

Behaviour - Actions, Goals and Action Tendencies

The goal annotations provided by authors (see Section 6.2.2) are the key to building accurate models of those elements that directly trigger behaviour (actions, goals and action tendencies). We work with the assumption that every event that was not annotated to be part of some goal was caused by reactive behaviour and is thus triggered by an action tendency. We further assume that the event was triggered by the preceding event. Thus, if event A is followed by event B and if event B was not annotated to be contributing to any plan, then this leads the system to construct an action tendency (i.e. a reactive behaviour rule) of the form “if A happens then trigger B”. This rule is entered into the probabilistic domain model with an initial probability. Additional instances of encountering A followed by B (not inside a plan) subsequently increase this initial property. The rule competes in the probabilistic domain model with other rules triggered by A. If the system for example encounters “A followed by B” three times and “A followed by C” twice in all the example stories, then the Probabilistic Domain Model will contain two alternatives for the Action Tendency for A: “if A then B” and “if A then C”, with the system being more confident about the former.

On the other hand, all events that are annotated to be part of a plan, will contribute towards building action and goal models. We assume causal links between the ordered chain of events contributing to a given goal. If an event has a property change annotation then we prefer to use the value of this property as the basis of the causal link. Otherwise we explicitly link the two events, i.e. we specify the occurrence of one action as the precondition for another one to occur. The effects of the last action in this causal chain correspond to a goal’s success or failure condition. The goal annotation contains information about whether a goal has succeeded or failed and thus, whether a failure or success condition should be created. Encountering an action in multiple contexts or encountering multiple instances of the same goal across all the example stories leads to either the creation of alternatives for these elements in the probabilistic domain model or an increase in probability of a particular alternative.

Emotional Reactions

Unlike the process for updating behaviour elements described above, where multiple alternative versions with associated probabilities are maintained, in the case of emotional reaction rules we may more conveniently accumulate the collective consensus of all collected stories. For every single action encountered in every example story we record the total number of occurrences of this action across all example stories and every emotion change associated with this action. Updating the model with a new story simply means updating these counts. All this is done on a per character

basis, as the resulting FAtiMA domain model will also contain individual emotional reaction rules per character. When it comes to extracting an actual FAtiMA domain model from the probabilistic domain model (i.e. the most probable domain model), emotional reaction rules are constructed according to the following rules:

- An action, for which no single emotion change annotation was made across all stories, will not result in an emotional reaction rule. All other actions will result in exactly one emotional reaction rule.
- The values for the *desirability*, *desirability for other* and *praiseworthiness* OCC parameters that are part of an emotional reaction rule for a given action are calculated taking into account all collected emotion change annotations for this action.
- All emotional reaction parameters are also scaled by the number of total occurrences of the respective action. This means that actions for which emotion change annotations were specified relatively often, result in OCC emotional reaction parameters of comparatively high magnitude compared to actions which were only annotated seldom.

Looking up an emotion in the OCC model (see Figure 2.5) yields the sign (positive or negative) for the *desirability*, *desirability for other* and *praiseworthiness* OCC parameters. But as the ENIGMA client software deliberately does not ask the user for a magnitude of the emotion change, we also cannot deduce the magnitude of the OCC parameters. Therefore the above rules make use of a) the absence as well as the presence of emotion and b) the averaging of all emotions for a given event to derive consistent magnitude values.

Of course the precision of this calculation could be improved if users would also be able to indicate an intensity of emotion change directly in the authoring tool's user interface but in order to avoid further user interface complexity such a feature was left out. So far we can only speculate that the more crude method of equating frequency with magnitude as described above would yield a useful enough approximation, but this was at least what we intended to try first.

6.4 Usability Trial

When the initial implementation of ENGIMA (excluding any of the learning and mixed initiative features) as described in Section 6.2 was completed, a small informal usability trial was performed as the next step. Five users were asked to try out the software and their usage of the system was observed and logged. The five users were all interactive storytelling experts though none of them had prior experience in

working with the FAtiMA agent architecture. Each of them saw ENIGMA for the first time when participating in the trial. The necessity for running a usability trial coincided with the author's attendance of the Third International Conference on Interactive Digital Storytelling (ICIDS), which made access to this pool of expert users possible. Expert users were the preferred choice for this first trial instead of general users because it was hoped that they would be able to provide more insights and feedback.

6.4.1 Setup

The ENIGMA system was preloaded with graphical resources from a Little Red Riding Hood setting². This included 4 characters (Red Riding Hood, Wolf, Grandma, Woodsman), a few theme related items (cake, wine, mushroom) and locations (a few different forest locations, grandma's house, grandma's bedroom). The action library contained only 2 exemplary generic world events (snow, lightning strike) and 9 generic character actions (steal, offer, accept, refuse, introduce, cry, sleep, wake up, eat). The system had no initial knowledge of any goals. Users were given a short demonstration and explanation of the system and then left to their own devices. The only instruction given was to create a little red riding hood themed story. The users were observed while creating the stories and subsequently interviewed about their opinions about the system. Each user interacted with the system for around 15 minutes.

6.4.2 Observations

The most striking observation was that all five users despite being experts on interactive storytelling had difficulties grasping the concepts underlying the system. In particular the distinction between actions and events (instantiated actions), the type system for limiting the possible bindings for action parameters and the mapping between a symbolic dialogue action and its natural language textual representation seemed to be causing confusion. This became especially apparent when users were creating new actions via the new action wizard (see Figure 6.8).

As a result, users ended up avoiding the creation of new actions and almost exclusively used the pre-defined generic actions to assemble stories which as a result were not very creative. The distinction between knowledge representation and presentation layers in general was poorly understood. Three out of five subjects clearly did not understand that the comics were merely a graphical representation of the authored knowledge and that in order to change the contents of a comics frame the related

²The story domain was inspired by a series of workshops on authoring interactive stories that used Little Red Riding Hood as a common theme for comparing different approaches to interactive storytelling (Spierling and Iurgel, 2008; Spierling et al., 2009). ENIGMA was one of the systems featured in these workshops.

action instantiation has to be edited. Instead the presence of a comics visualisation made them automatically assume that the purpose of the system is to manipulate the comic images directly. When this was not working as expected, these subjects showed signs of frustration.

A common observation across all subjects was that optional annotations (of property changes, emotion changes and goals) were ignored. We discussed earlier how ENIGMA depends on this annotation metadata if it is to make sense of collected stories, so the fact that in its current implementation users do not make use of these annotation facilities highlights a clear shortcoming of the application.

6.4.3 Implications

The implications from the ENIGMA usability trial were discouraging. The system clearly did not make a good enough job of explaining the underlying authoring concepts. Even expert users did not manage to operate it in the intended way so as to produce meaningful results (i.e. stories that can be actually processed). This was especially disheartening as its probably most complex feature, the mixed initiative suggestions were not even integrated into the prototype yet. In particular the distinction between an action definition (template), an action (instantiation) and the action's visual representation seems difficult and needs to be either explained in more depth up front or made clearer in the UI.

As the goal for the production of ENIGMA was to end up with a system that can be used for an actual user evaluation, defeat had to be eventually admitted. The choices for continuing work on ENIGMA were to either produce a high quality system that conveys the underlying concepts effortlessly to the lay user or to train enough participants sufficiently so that they can author stories as intended with ENIGMA's prototype implementation. Either of these paths requires more resources than available in a PhD research project, especially, when taking into account that a number of complex ENIGMA features were still left to implement. With its ambitious design and the eventual realization of the many unanticipated obstacles preventing its successful implementation, ENIGMA is in good company. On his website, Chris Crawford (Crawford, 2012b) published a statement detailing what went wrong in the design of the Storytron engine. While Storytron was arguably a much larger scale project, some problems mentioned by Crawford are similar to those encountered in the ENIGMA project. In particular, also the Storytron system was too complex for the majority of users to understand and effectively use it.

The problems observed above also echo the earlier-mentioned sentiment by Spierling and Szilas (2009) that conceptual and not user interface issues lie at the heart of the authoring bottleneck. Even though the system was explicitly designed to hide the full complexity of authoring FATiMA based emergent narratives, partly in response to

the above mentioned sentiment, we still mainly observed confusion about conceptual aspects in the usability trial. Whether a more intuitive user interface for ENIGMA could allow these concepts to be understood more easily is an open question.

6.4.4 Ways Forward

It would be wrong to write off ENIGMA as a complete failure. For one thing, it might just be unrealistic to expect an immediate grasp even by domain experts of such a fairly complex system. With increased training time and specially prepared tutorials etc users are likely to fare much better. The problem of users not using the annotation facilities could be addressed through either increased user awareness of the purpose of these annotations or by making them compulsory. Doing this however without creating a more frustrating user experience is difficult. In general, further improvements of the user interface could probably improve the users' performance in using ENIGMA, but this would require a similar evolution step as that from Wide Ruled to Story Canvas as discussed in Section 4.2.3. While this work turned out to be outside of the scope of the project, given adequate resources, we considered experimenting with the following ideas for evolving the system's user interface:

- Automatically obtaining (partial) action definitions from a lexicon / ontology, to prevent users from having to define them entirely manually
- Enabling visual editing of actions, making more use of the information displayed by the comics frames. E.g. placing two characters in a scene, placing an object in the hand of one of them and then dragging an "action" arrow from the other character to the object could lead to inferral of action subject and object.
- Adding a natural language input / search component to the user interface, that allows automatic instantiation of recognized actions from a descriptive natural language action string.
- One fundamental problem is the missing ability for visualizing newly added actions. We considered the use of other character animation techniques, including procedural animation, inverse kinematics or even cheap motion capture using consumer-level devices like the Microsoft Kinect for filling the gap between symbolic actions and their visual representation.

6.5 Conclusion

ENIGMA is an authoring system for building FAtiMA agents from example stories collected through crowdsourcing. In order to learn from the collected stories, they need

to be expressed in a computer friendly way and annotated with additional metadata. This chapter has described how users specify stories with ENIGMA and how the system was designed to adopt the Crowd Task Adaptation approach through its mixed initiative mode to elicit user feedback.

Considering the authoring tool requirements proposed by Medler and Magerko (2006) and discussed in Section 4.2, ENIGMA only clearly satisfies one, namely **generality** (i.e. it is story-domain independent). **Usability** was something ENIGMA's design aspired to but that its implementation did not manage to realize. ENIGMA falls short of some of the other listed authoring tool requirements due to it being designed as a crowdsourcing tool. In particular **debugging** and **pacing and timing** are only really applicable where an author is in control of the entire interactive narrative, with ENIGMA such responsibilities are relinquished to the system itself.

It eventually transpired that plans for the ENIGMA system were too ambitious to realize within the scope of a single PhD project. Therefore the research direction has shifted towards a simpler, albeit less generative approach towards interactive storytelling which is described in the next chapter and which salvages much of the development work that had gone into ENIGMA up to that point.

Chapter 7

The CROSCAT Authoring System

The previous chapter concluded that the first authoring system prototype, ENIGMA, was found to be too complex for evaluation purposes within the means of this PhD research. In this chapter, a second authoring system prototype CROSCAT is discussed. The development of CROSCAT had the ENIGMA code base as a starting point and thus both systems have many commonalities, especially in terms of user interface, look and feel and back-end architecture. However, a much simpler story representation model is employed for CROSCAT. While ENIGMA aimed to construct the planning domains for FAtiMA agents from the example stories provided, CROSCAT maps the collected stories into a branching story tree. While such a data structure has much less generative power compared to an emergent narrative based on autonomous agents, it also does not require as semantically rich knowledge acquisition. This allows the authoring task to be much more intuitive and thus makes it easier to learn for contributing authors.

While it superficially looks very similar to one of the many commercially available storyboarding tool and comics editors¹, CROSCAT is fundamentally different from these systems since its ultimate goal is to assemble a branching story tree and not just a single story line. Some of the authoring tools reviewed in Chapter 4 such as INSCAPE (Balet, 2007) and ASAPS (Koenitz, 2011a) are also used for the creation of branching story graphs. But CROSCAT differs from these systems in its reliance on Crowdsourcing. Not a single author, but the system itself creates the branching points from many collected linear stories, which allows the story world creation to be distributed and scalable. Crowd Task Adaptation via the modification of the back story given to each user makes this possible, as without this strategy the system would not be able to merge the collected stories in a sensible way.

¹Some examples of commercial comics editing software include Pixton (www.pixton.com), Chogger (www.chogger.com) and Kar2ouche(<http://www.immersiveeducation.eu>)

7.1 Overview

The design of the CROSCAT system takes into account some of the lessons learnt from the usability trial of the ENIGMA system. This section gives an overview of how the system differs from ENIGMA, both from the perspective of a contributing author using the frontend authoring tool for telling a story and that of the principal author using the system backend for collecting and processing stories.

7.1.1 User Perspective

From an author's point of view the key difference between the two systems lies in the way that stories are created. A user of the CROSCAT software specifies a story directly using the comics story presentation layer. I.e. instead of having to choose or define an action and instantiating it, the user directly specifies the graphical contents of a comics frame. The user interface (Figure 7.1) directly affords the user with means to chose a background scene, place characters and items on the scene, change characters' facial expressions and attach speech or thought bubbles to characters. Narration boxes may also be created and all text used in narration boxes and speech or thought bubbles may be directly edited on the fly.

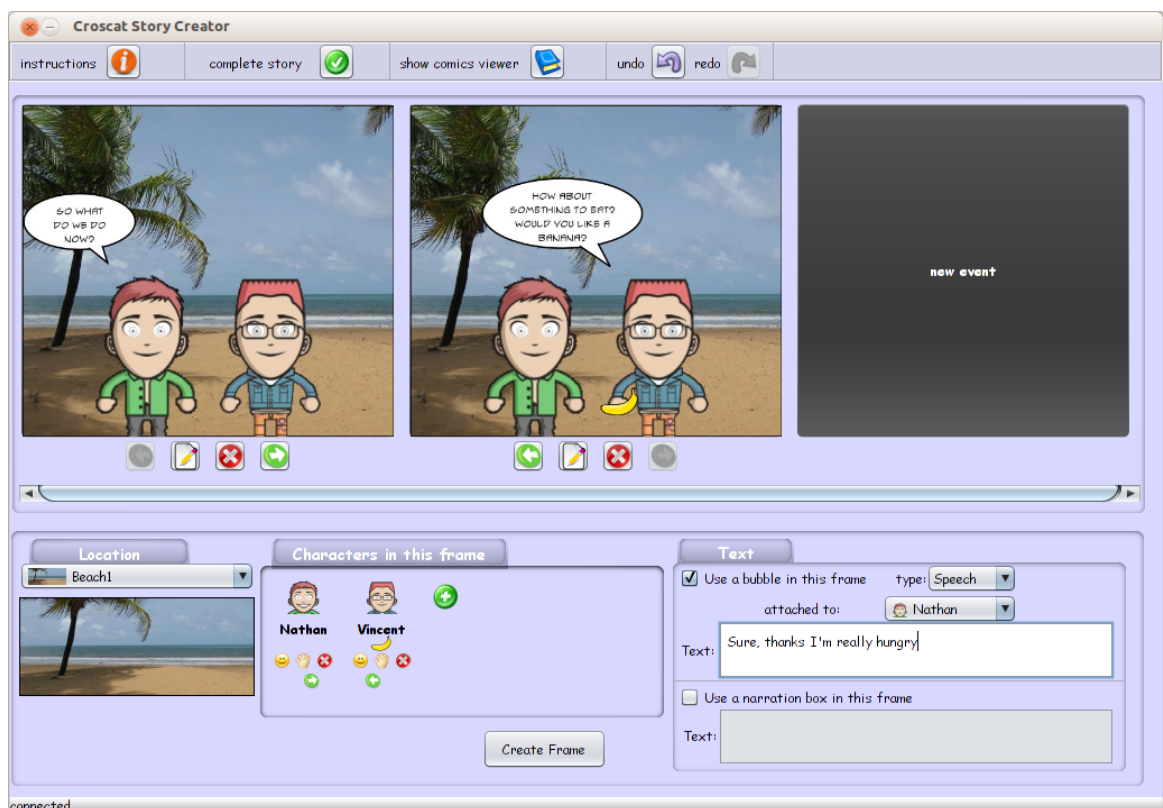


Figure 7.1: Main graphical user interface of the CROSCAT Tool.

In contrast to ENIGMA there is also no annotation of stories required by users of the CROSCAT authoring tool. Also, CROSCAT has no equivalent of ENIGMA's

mixed-initiative mode, i.e. users will not have to enter dialogues with the system. Overall all these changes result in a much more user-friendly and less restrictive user interface. Users are liberated from the constraints imposed by the FAtiMA event syntax and can concentrate on the creation of narratives. Like in the ENIGMA system, users are given a limited set of story building blocks from which the comic frames may be assembled. Locations, characters and their available facial expressions and the items that can be held by characters are all predefined by the principal author. Users are not allowed to add additional content in any of these categories.

A few additional convenience editing functions not present in ENIGMA were also added to CROSCAT. Users are allowed to delete, move or retrospectively edit arbitrary frames in the story time line. A final user-interface difference between CROSCAT and ENIGMA is that the CROSCAT interface has a built-in back story viewer. The back story is also a comic that establishes context by introducing the user to the story domain. The task of a CROSCAT user is to continue the back story they were given, i.e. to create a suitable ending for it. To ensure users actually do this, reading the provided back story is mandatory. The CROSCAT main window becomes only accessible, after all frames of the back story comic have been read.

7.1.2 System Perspective

Compared to ENIGMA, where the authoring experience of the user was designed to produce the story representation requirements of the system, CROSCAT has taken the opposite approach. In its design, the authoring user experience has taken center stage and the backend system is accommodating. The system does not "understand" the events / frames that were authored in any meaningful way. The data representation of individual events is solely concerned with capturing a frame's visual composition from the building blocks available. Listing 7.1 shows an example of how the System represents a CROSCAT frame.

Listing 7.1: Example of CROSCAT frame representation (XML serialization)

```
<StoryFrame id="00000003" location="Beach1">
  <Narration text="They compared the contents of their bags..." />
  <Character name="Laura" emotion="neutral" object="bottle of water" />
  <Character name="Monica" emotion="serious" object="sun cream" />
  <Character name="Nathan" emotion="neutral" object="banana" />
  <Dialogue owner="Laura" text="We only have a single bottle of water so we
    should try and use it wisely." type="SPEECH" />
</StoryFrame>
```

Considering the missing annotations and the human rather than machine-friendly representation of stories, it is clear that ENIGMA's methods for processing the collected stories are not suitable here. The system lacks for example the knowledge to

recognize similarities between two events / frames, something that in the ENIGMA system would have been possible in some cases, e.g. where two distinct events are two separate instantiations of the same action.

CROSCAT thus takes a very different approach, abandoning the use of FAtiMA as the target IS runtime engine. Instead the goal of the CROSCAT system is to build a branching story tree from the collected stories. This much simpler data structure is suited to the low semantic information content of the input data available to the CROSCAT system. CROSCAT is however still a suitable vehicle to investigate the benefits of Crowd Task Adaptation, discussed in Chapter 5. Because the collected stories are expressed as comics, there is an inherent structure present in them with events clearly separated into individual frames. A comparable structuring could not be as easily extracted from only written, textual stories. Thus, CROSCAT can make use of branching story graphs, an IS story world representation format that is only concerned with structure. CROSCAT adapts the strategy of back story adaptation to support merging the collected stories into a single branching story graph.

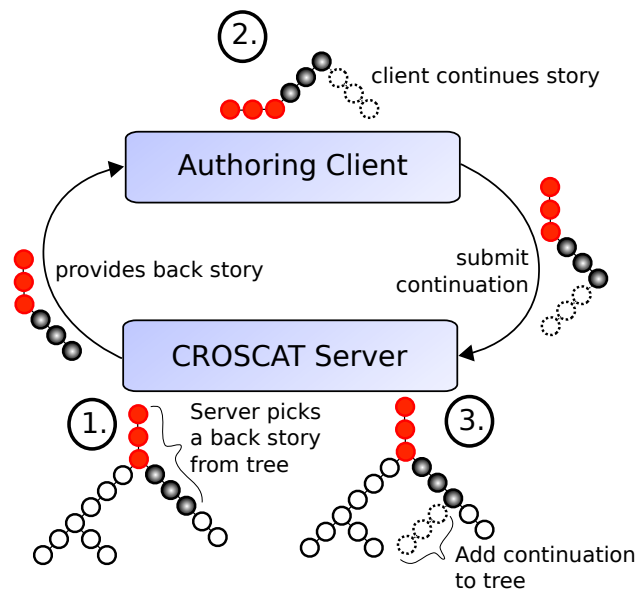


Figure 7.2: Overview of the CrowdTask Adaptation via back story selection feature in the CROSCAT authoring tool.

Figure 7.2 illustrates how this is done. The principal author bootstraps the system with an initial back story (represented in Figure 7.2 by the coloured top nodes of the graph). This is the back story served to at least the first two contributors. The story collected from these first participants are appended to initial back story nodes and the graph is a tree with a single branching point. From this point onwards, the server picks a single partial path from the graph as the back story provided to the authoring client. This path will always include the full initial back story provided by

the principal author. The continuation of this story sent back by the client is inserted as a new branch into the story graph. The story graph assembled in this way will always be a tree, i.e. branches do not fold back and merge with other branches. The back story sent to a client at any given point is chosen so that branching points are evenly distributed in the overall story tree. The next section discusses the algorithm for selecting the ideal back story for achieving this goal.

7.2 The Back Story Selection Algorithm

The goal of the back story selection algorithm is to determine a node in the story tree, from which we want the next contributed story to branch off. This point should be chosen so that we have an overall balanced story tree. The notion of balanced tree we put forward here, should not be confused with balanced binary search trees, such as red-black trees (Guibas and Sedgewick, 1978). CROSCAT's concept of balance is based on the hypothesis that the structure of a branching story tree will affect the enjoyability of a user's interactive story experience, when navigating the tree. Consider for example the scenario, in which every contributor receives the same basic back story provided by the principal author. Simply merging all those stories results in a tree as that shown in Figure 7.3².

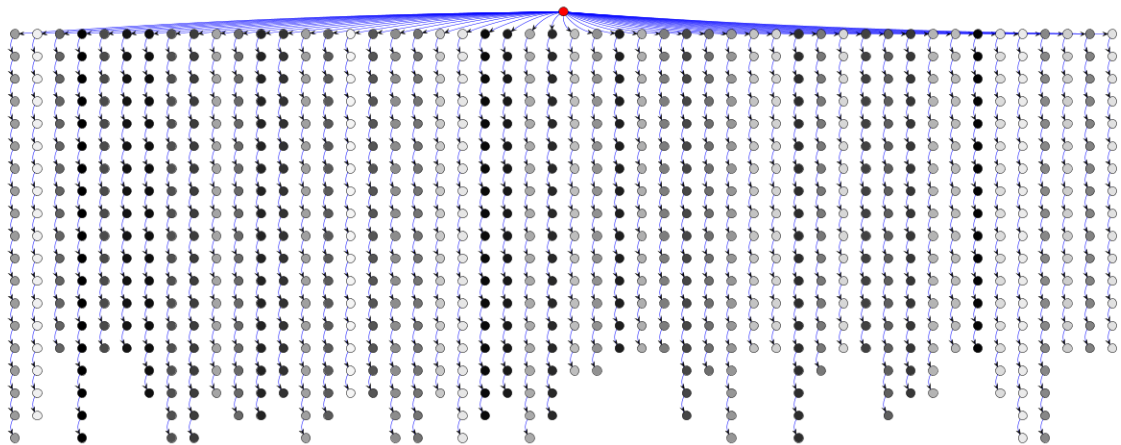


Figure 7.3: Branching story tree if no back story selection would be used.

It is not hard to see that this tree does not capture the ideal of a branching interactive story very well. There is only a single branching point, at which the story branches off into 50 directions. Someone interacting with this story would be first over-

²Some additional merging not shown in this figure may be occurring in this scenario if the first n events in separate stories are identical. In this case, the branches in question would be summarized into a single branch and a branching point would be added at the n th event.

whelmed by the amount of options at the single branching point and subsequently be disappointed by the fact that no further branching points are available. Instead we strive for more evenly distributed branching points with fewer options. We also want each story path to have roughly the same amount of branching points. This notion of an ideal, balanced interactive story tree is intuitive but difficult to precisely define. Idealness in this case is defined as resulting in a more satisfying interactive experience, which is a decidedly subjective measure, but it is clear that we can do better than Figure 7.3.

The CROSCAT back story selection algorithm uses a heuristic for determining the ideal node in the tree for receiving an extension. Each node is assigned a score made up of different components. The highest scoring node (or if there are several highest scoring nodes, a node randomly selected from that set) is the one chosen for receiving the extension. The heuristic's components encapsulate our criteria for what constitutes an ideally balanced branching story tree. Below the individual components of the heuristic scoring function are described and visual simulation results (i.e Figures of the resulting story trees, such as Figure 7.3) are presented. All simulation results reported in this section were obtained by simulating the effect of 50 authors consecutively contributing a story, each of a randomly chosen length between 15 and 20 events. The graphs shown in this section all use the same colouring scheme, which follows the rules below:

- The root node is always shown in red
- Each individually submitted story is assigned a unique grey level.
- Darker grey tones represent stories submitted earlier. Nodes belonging to the first received story are coloured black and nodes of subsequent rehearsals increasingly lighter.
- Branching nodes are always drawn in the colour of the story, which lead to the node's original creation.

By following our development of the heuristic scoring function step by step, i.e. presenting a series of 3 increasingly more complex heuristics, in the order in which they were originally derived, the following section attempts to communicate our criteria for an ideally balanced branching story tree.

7.2.1 Heuristic Function 1: Distance From Branching Points and Leafs

The first proposed property of an ideally balanced graph is that branching points are evenly distributed. Consequently, in order to satisfy this condition, new branches

are preferably inserted as far away from other branching points as possible. This translates very conveniently into a heuristic for our scoring function. The reward for evenly spaced branching R_{DB} is defined as the **shortest distance**, i.e. length of the shortest path in any direction (i.e up or down the tree), **from any branching point or leaf node** (as leaf nodes are obviously story endings, branching there makes little sense). For branching points and leaf nodes $R_{DB} = 0$.

We also add another condition that runs counter to the above requirement of evenly distributed branching points: We want branching points not to be too close to a story ending, i.e. a leaf node. Laurel (1993) characterizes narrative as a flying wedge of possibilities (see Figure 7.4).

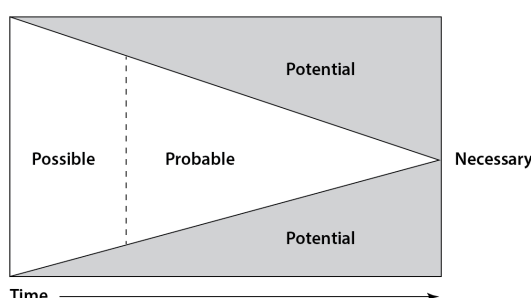


Figure 7.4: Laurel's view of narrative as a flying wedge of possibilities (Laurel, 1993)

The idea expressed by this wedge or funnel is that as the events of a narrative progress, the space of possible narrative futures narrows, until eventually a necessary outcome becomes inevitable. This is a consequence of the establishment of characters and conflicts that demand resolution over the course of the narrative. This insight has important consequences for the collaborative construction of interactive narratives with CROSCAT: According to the flying wedge theory, the further a back story provided to a user has progressed toward a narrative outcome, the more likely it is that the user will not create a novel story continuation, but instead repeat a path that was already present in the graph.

Therefore the heuristic function rewards nodes for their location in a wide rather than narrow section of the flying wedge. Nodes still having many narrative possibilities are more suitable candidates than nodes very close to an inevitable necessary outcome. The CROSCAT heuristic encapsulates this notion in the reward R_{DL} for a node's **average distance from leaf nodes**. R_{DL} takes into account all possible directed paths that connect the given node with any leaf. R_{DL} for this node is defined as the average length of all these paths. Note that the first heuristic component R_{DB} also rewards distance from leaf nodes but unlike R_{DL} it does so only locally and not globally for the entire tree.

The CROSCAT heuristic needs to strike a balance between the two reward components R_{DB} and R_{DL} . We therefore assign weights w_{DB} and w_{DL} to each component

and examine the effects of different weights through simulation. Figure 7.5 shows the simulation results for 2 different sets of weights for the heuristic function H_1 , which is defined as:

$$H_1(node) = w_{DB} * R_{DB}(node) + w_{DL} * R_{DL}(node)$$

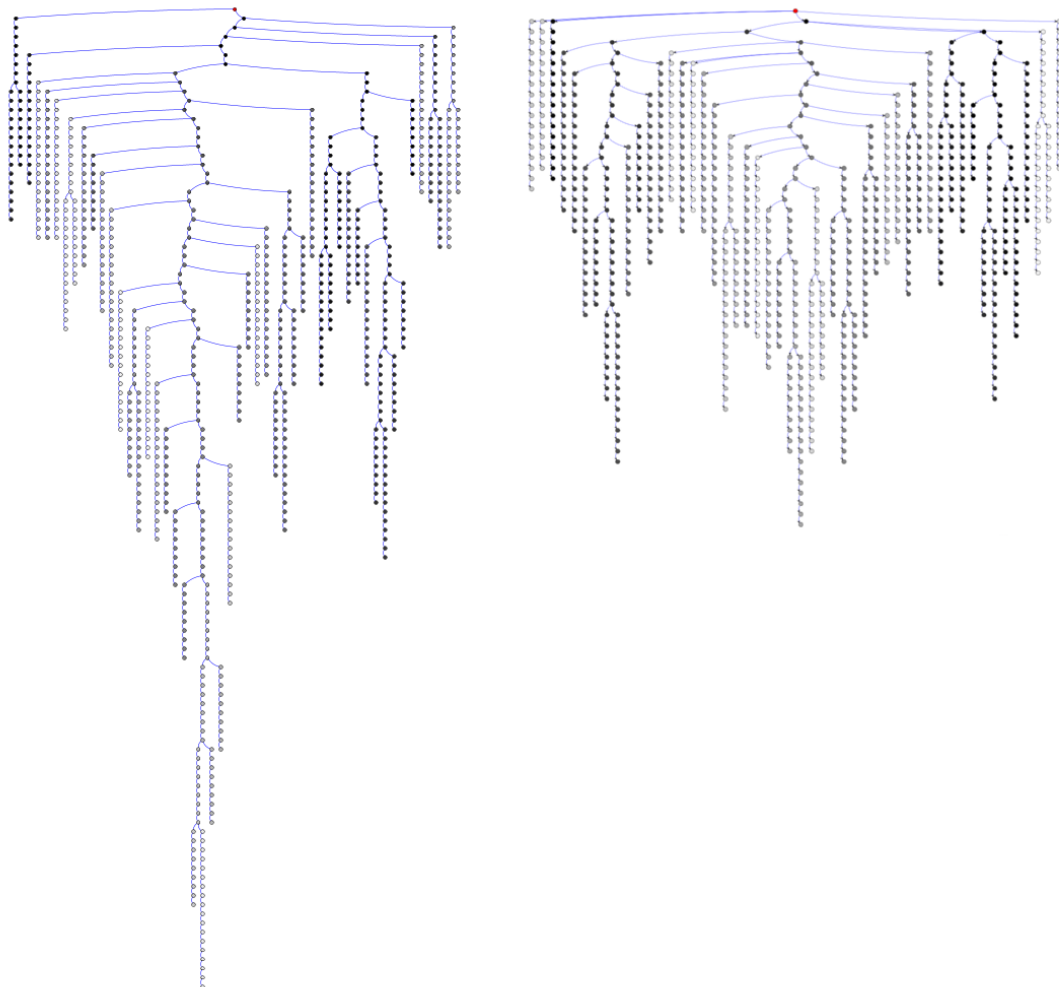


Figure 7.5: Simulation results of 50 story submissions to CROSCAT using heuristic function 1. Left side: $w_{DB} = 2, w_{DL} = 0.5$, right side: $w_{DB} = 2, w_{DL} = 1$

Through manual experimentation and a visual inspection of the story graph, the combination of weights shown in the right tree in Figure 7.5 ($w_{DB} = 2, w_{DL} = 1$), was chosen as the best compromise. This decision is inevitably a subjective one, but the immediately obvious contrast in structure between the two graphs in Figure 7.5 illustrates the aesthetic criteria used for making this decision. The right hand side graph is clearly overall more balanced than the left hand side graph, where a single almost never-ending path dominates the entire graph.

7.2.2 Heuristic Function 2: Adding Branching Ancestor Penalty

Upon visual inspection, the first heuristic function H_1 , even with adjusted weights, still has some problems. Despite the R_{DB} component in H_1 , there is an uneven distribution of branching points. While some paths through the tree are full of branching points, several others have very few branching points. The revised heuristic function H_2 addresses this problem through an additional component: \mathbf{P}_{BA} is a node's **penalty for every branching ancestor node** that it possesses. We define an ancestor of a node x as any node along the shortest path between the root node and x . A branching ancestor node is one that has more than one child node. P_{BA} is simply the count of a node's branching ancestors. As the other heuristic components, P_{BA} is weighted. This results in the following equation for H_2 :

$$H_2(node) = H_1(node) - w_{BA} * P_{BA}(node)$$

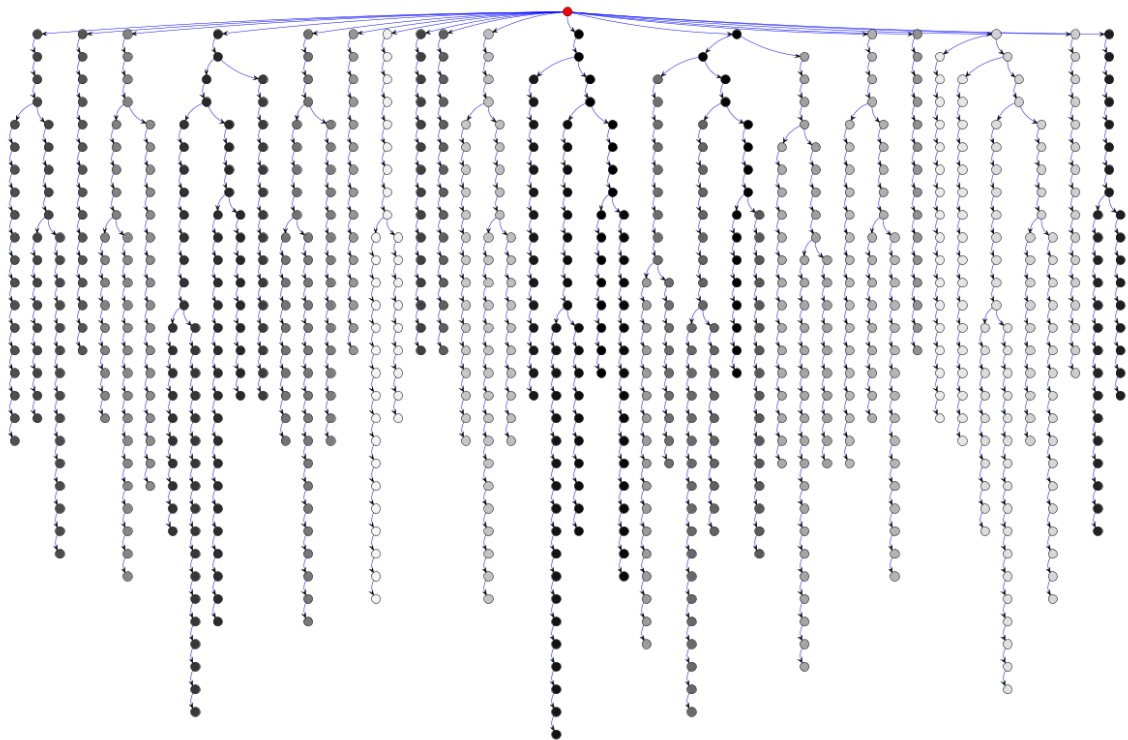


Figure 7.6: Simulation results of 50 story submissions to CROSCAT using heuristic function 2. $w_{DB} = 2, w_{DL} = 0.5, w_{BA} = 3$

Figure 7.6 shows the simulation results for the H_2 function. As the subjective process of assessing the weighting factors is analogous to that already illustrated in Figure 7.5, we omit showing further weight combinations. Suffice it to say that the weights shown in Figure 7.6 ($w_{DB} = 2, w_{DL} = 0.5, w_{BA} = 3$) were chosen as the best combination.

7.2.3 Heuristic Function 3: Adding Child Branch Penalty

The second heuristic function H_2 as shown in Figure 7.6, improves the distribution of branching points across different paths compared to H_1 , but is still not an ideal solution. The apparent shortcoming with H_2 is that it creates too many branches off a single node. In Figure 7.6, the root node has a branching factor BF of 17. This can be explained by the fact that the root node is the only node in the graph where there is no branching ancestor penalty P_{BA} . To counteract this problem we introduce an additional penalty for nodes that have a high branching factor (BF) to the heuristic function. We define the branching factor BF of a node, as the number of direct child nodes. In graph theory the branching factor is also called outdegree.

The fourth heuristic component $\mathbf{P_{BF}}$ is a **penalty for nodes having a higher branching factor than threshold amount** ($\mathbf{T_{BF}}$), which defines an acceptable branching factor. Formally, P_{BF} is thus defined as below:

$$P_{BF}(node) = \begin{cases} BF(node) - T_{BF} & \text{if } BF(node) > T_{BF} \\ 0 & \text{otherwise.} \end{cases}$$

A value of 3 was chosen as the acceptable branching factor threshold T_{BF} in CROSCAT. This value is based on the intuition that 3 is a number of choices that is most definitely positive to the user experience, i.e. it is better to have 3 than 2 or no choices. Furthermore, having a few more than 3 choices (as the function penalizes but not outright forbids a higher branching factor than 3) is also still positive, although the benefit diminishes with every added choice until some threshold is passed where the number of choices becomes a burden. The exact value of this threshold is unclear, but 17 for example as seen in Figure 7.6 is probably exceeding. Unfortunately we are not aware of any research investigating the ideal number of choices in interactive stories. Widely cited work in a related, more general context is the theory of the magical number seven plus or minus two by Miller (1956). It claims that human memory is limited to processing around 7 items of information (with some individual variation of ± 2) at any given time. Miller bases this on experimental evidence. He cites several examples of one-dimensional judgement tasks (e.g. distinguishing the pitch of a set of n tones) where performance drastically falls off when more than 7 (± 2) choices are presented. The application of Miller's theory in inapplicable contexts (in particular user interface design) has been widely criticised e.g. by Góczy (2010). It is unclear, whether the magic number 7 applies to choices in Interactive Storytelling but in lack of other evidence it provides at least some guidance. The theory resonates well with our chosen branching threshold of 3 (with some leeway for increase).

Apart from the threshold value T_{BF} , an additional weighting factor w_{BF} also influences

the effect of the penalty P_{BF} . This leads to the below definition of H_3 :

$$H_3(\text{node}) = H_2(\text{node}) - w_{BF} * P_{BF}(\text{node})$$

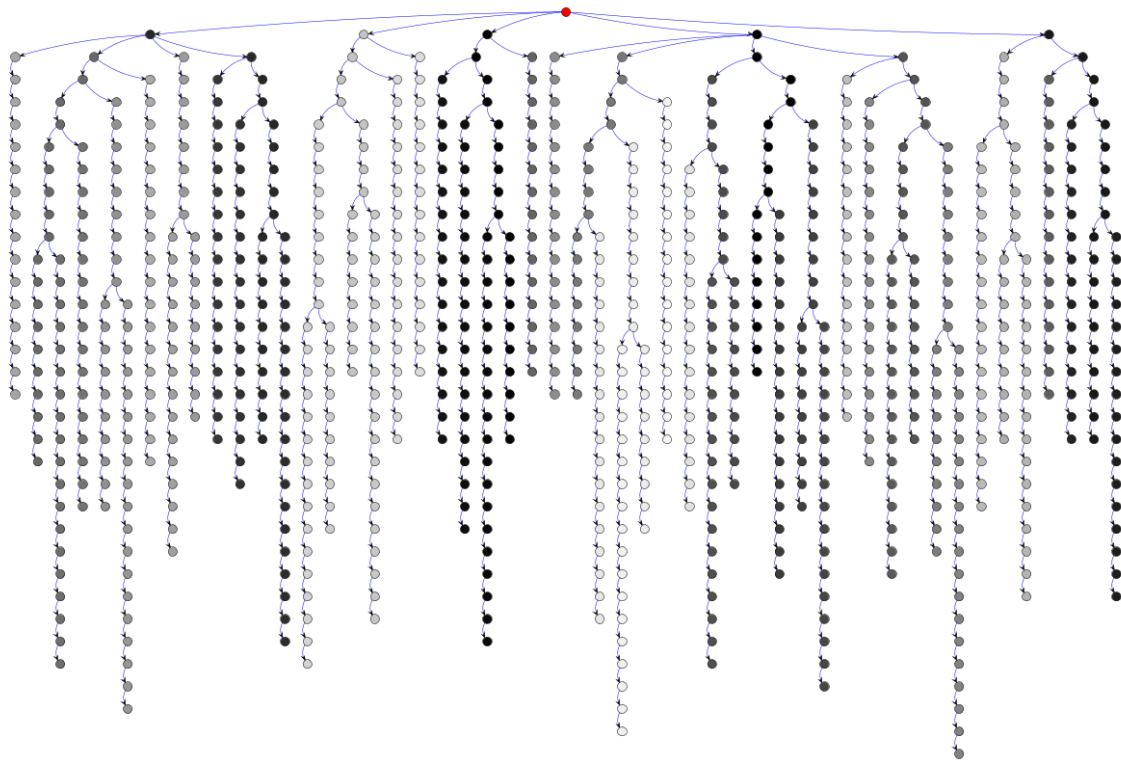


Figure 7.7: Simulation results of 50 story submissions to CROSCAT using heuristic function 3. $w_{DB} = 2, w_{DL} = 0.5, w_{BA} = 3, w_{BF} = 4, T_{BF} = 3$

Simulation results for the H_3 function are shown in Figure 7.7. The branching factor of the root node has noticeably reduced compared to Figure 7.6. We again abstain from showing different weight combinations and instead immediately report the final weights as $w_{DB} = 2, w_{DL} = 0.5, w_{BA} = 3, w_{BF} = 4, T_{BF} = 3$.

7.2.4 Other Heuristics

The heuristic H_3 uses only topological information about the story graph itself to determine the best node for receiving an extension. With a deeper understanding of the narrative meaning of events, this heuristic could of course be significantly improved. Some possibilities for additional heuristics, are:

- **Dramatic Impact:** Users like to make meaningful choices with dramatic consequences. A choice of whether the hero should pull the trigger on the villain or not is more meaningful, then a choice of whether he should eat a banana or an

apple for breakfast. The event of contemplating murder has a higher dramatic impact than the event of contemplating breakfast. Dramatic impact will likely extend beyond a single event. In many stories there will be entire stretches of high and low dramatic impact. If we had a way of evaluating the dramatic impact of individual events or sequences of events, we could reward nodes for their placement in dramatic regions.

- **Branch popularity:** Under certain conditions, the illusion of choice may be sufficient for creating a feeling of agency in the audience of an Interactive Story. This was for example found in a study by Fendt et al. (2012). In similar work Figueiredo and Paiva (2010) argue that psychological persuasion techniques may be used for influencing the choices users make in interactive stories. If there was some way to estimate the persuasive potential of the available choices at a given branching point, this knowledge could feed into an additional heuristic for CROSCAT. Surely, it is overall more valuable to extend the content in branches that are more likely to be chosen. The details of how the persuasive potential of a certain choice may be automatically evaluated go beyond the scope of this PhD.

While the above two possibilities were not implemented in CROSCAT, the next section describes a similar feature called antonym insertion. A semi-automatic version of this feature has been implemented in CROSCAT.

7.3 Antonym Insertion

We have already described above, how branching at a dramatic choice point is advantageous. If such a point is chosen as the winner by the back story selection algorithm, a contributing author will be presented with a back story that ends at an interesting dramatic junction. There is, however, nothing to guarantee that their created story will actually lead down the opposite path as intended. If the author replicates the same option that was already present in the graph, then this interactive choice point will be disappointing for the (non-authoring) end user.

Consider the example story fragment shown in Figure 7.8. In this story, event 4 is a dramatic choice point (should or shouldn't Tom obey Ringo's command). If event 5 is chosen as the winner by the back story selection algorithm, then that decision is already made and included in the back story. If event 4 is chosen on the other hand then the contributing author has to start their own story with a dramatic decision. However, the author might make the same choice as that made in event 5 (Tom obeys) and essentially just replicate event 5 (though probably with slightly different wording or details, so that it will be difficult to automatically determine that both events are



Figure 7.8: Example of a story fragment containing a dramatic choice point (Node/Event 4)

equal.). Furthermore, not all dramatic choices are necessarily preceded by an event announcing and presenting the choice as event 4 in this example.

CROSCAT implements the antonym insertion strategy as an alternative way of promoting branches at dramatic choice points that avoids the problems described above. We define antonymy as a symmetric relation between two events that describes the events as being direct opposites in a given story context. These two events are then called antonyms of each other. This is analogous to language where antonyms are pairs of words with clearly opposite meaning, e.g. (yes/no, agree/disagree), etc. For example, an antonym of event 5 in the example story in Figure 7.8 in its story context would be “Tom disobeys and refuses to hand Ringo the bag”. While there might be other ways to express the same meaning and thus other possible antonym events of event 5, these alternatives are all essentially synonymous. For the antonym notion in CROSCAT it is only important that the events are obvious opposing alternatives in a choice story situation.

Almost every event has an antonym. Simply negating an event (e.g. “Tom does not crack his piggy bank open” as a negation of event 1) will typically create an antonym. However these antonyms are not necessarily meaningful for our purposes. Negation of events often leads to an event describing the absence of action. This is only meaningful in certain story contexts. For example if Tom pondered for a long time whether to open the piggy bank or not, maybe talked about it and maybe even already holds the hammer in his hand then an event like “Tom does not crack his piggy bank open” has meaning. If however the action whose absence is described has no bearing on the story, the antonym is not meaningful. For example, in a story where neither the piggy bank, nor Tom’s need for money is ever mentioned before, an event such as “Tom does not crack his piggy bank open” is meaningless.

CROSCAT aims to use only meaningful antonyms. This is because, a pair of meaningful antonyms represents a meaningful choice in the story. In general, we characterize the admittedly fuzzy notion of a meaningful antonym as an event, which represents an alternative course of action that a) an audience could reasonably expect to occur and that b) in the eyes of the audience has the potential to alter the course of the story. A relevant piece of supporting related work in this context is the study per-

formed by Cardona-Rivera et al. (2014) that found users to achieve a higher sense of agency when given choices that could be foreseen to lead to meaningfully different story outcomes in contrast to those that didn't.

7.3.1 Using Antonyms

We have so far described the concept of antonym events, but we have yet to discuss how CROSCAT would use antonyms in combination with its back story selection algorithm. Figure 7.9 illustrates this process. It shows several events in the branching story tree (1,5,6,8,10,12,13), for which CROSCAT is aware of possible antonyms. An event for which an antonym event is known can contribute an additional constant reward R_{an} to the back story selection heuristic. This is illustrated by the plus signs in Figure 7.9. Importantly the reward is not given to the event node, for which the antonym is defined but to its parent node. Furthermore, the antonym reward is only granted if said parent node does not yet have a child node that is equivalent to the antonym event.

For example in Figure 7.9, the system knows an antonym event for node 10, which results in the antonym reward R_{an} being granted to node 10’s parent, node 9. In contrast, node 10 does not receive the antonym reward, because it does not have a child, for which an antonym is known. Node 4 has a child for which an antonym is known (node 5), but since node 4 already has the antonym event in question as a child (node a5), the antonym reward R_{an} is not granted to node 4. If a node has more than one child for which an antonym is known (case not illustrated), R_{an} is still only granted once. When it is given to a node x , R_{an} is simply added to $H_3(x)$ to form the final score for x .

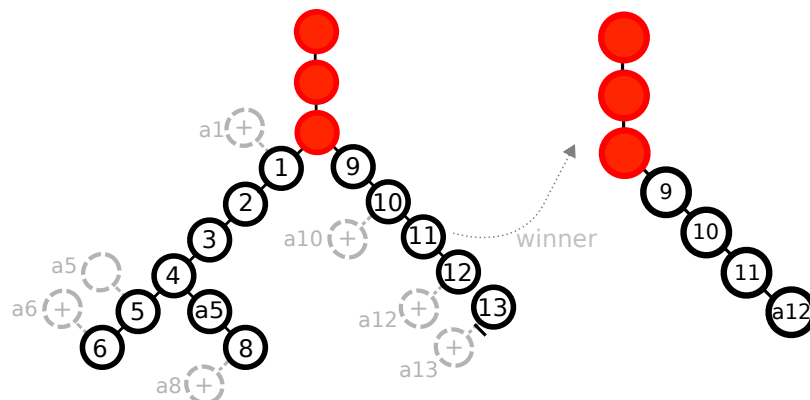


Figure 7.9: Example of a CROSCAT story tree containing some events, for which the system knows how to create antonyms.

If a node is selected, that was awarded the antonym reward R_{an} , then a new event node is instantiated for the antonym event that has contributed this bonus. This new event node is appended to the back story, which is sent to the client. Figure 7.9 illustrates this by showing the back story that would result from node 11 being selected. Event a12 is instantiated and appended to the back story.

We have previously discussed parameters for weighing the different components of the heuristic. The value chosen for R_{an} is another such parameter. If R_{an} is very high, then nodes below which a new antonym event can be inserted will almost certainly always be selected, if such nodes are available. The other topological criteria in the heuristic H_3 will then only influence which of the nodes that allow antonym insertion is chosen. If R_{an} is modestly low on the other hand, then topological criteria can still outweigh antonym insertion and nodes have a chance to win, even if they do not lead to the appending of an antonym. Figure 7.10 shows the effect that different values of R_{an} have, given the weights we have discussed so far are chosen. The graph shows the results of a simulation where for different values of 10 different values of R_{an} , 1000 contributions each were simulated. Each generated event has a 50 percent chance to have an antonym. If R_{an} is 0 as we would expect, events that allow antonym insertion are not preferred and are only selected in 50 percent of cases, which is the probability of picking such an event by chance. As R_{an} is increased, supporting antonym insertion becomes a more important criterion for node selection, until at a value of $R_{an} = 10$ no nodes are selected on topological criteria alone anymore.

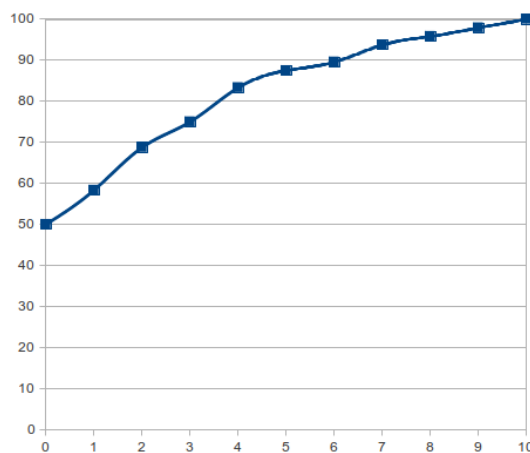


Figure 7.10: Simulation results showing the percentage of times that a winner that allows antonym insertion is selected plotted against the value of R_{an} , when antonyms are randomly distributed with a probability of 0.5

How R_{an} is chosen thus depends on how much value we assign to branching at antonym points compared to branching at topologically useful points. We will return to this question in Chapter 9, when we discuss the results of authoring experiments carried out with CROSCAT.

7.3.2 Obtaining Antonyms

The above explains how we use available antonym events, but how do we obtain them in the first place? Is it possible to automate the process of a) determining whether a meaningful antonym for a given event exists and b) generating the antonym for this event? To our knowledge there is no general purpose solution for negating natural language sentences and the problem statement we define here is even more complex than this. A conceivable solution might involve the WordNet (Miller, 1995) lexical database for the English language. We already mentioned the linguistic meaning of antonym (a pair of opposing words). If one could determine the central verb of an event (e.g. “obeys” in event 5 our example story: Tom obeys Ringo...), WordNet could be used to look up the antonym “disobeys”. If an antonym is found by WordNet then we would assume the existence of a meaningful antonym event. This is based on the assumption that verbs which have antonyms signify good choices and good choices in turn result in meaningful antonyms. For example, “obey” is such a verb, while WordNet does not define an antonym for “crack”, and indeed we have discussed how event 1 (Tom cracks his piggy bank open...) does not have a meaningful antonym. Generating the antonym event would then only amount to replacing the central verb with its antonym verb.

However, this approach really only works in the simplest of cases. In practice there are lots of problems. Even in our example we have neglected to address how the central verb of a sentence is determined. Composites such as the second half of the event 5 sentence (... and hands Ringo the bag) are also a problem. The assumption of equating the existence of an antonym verb with a meaningful antonym event is weak and language-dependent. There are other methods of negation, which this approach ignores, e.g using the particles no or not, or the antonym of a noun, adjective or adverb. Finally, events in CROSCAT are not only narrative sentences but entire comic frames. There might be no narrative text at all and instead the antonym might require negating the dialogue of a character, changing the facial expressions of characters and other graphical properties of the comics frame.

Creating a robust solution to all these problem is outside the scope of this PhD. Therefore CROSCAT does not include an automatic antonym generation capability. It does however provide the option of manually defining antonyms. In order to keep the story submission process as simple as possible, this annotation of antonyms is not done by the crowd workers in the CROSCAT authoring tool but by a special user with elevated privileges (e.g. the principal author). This user may review any submitted story and define antonyms for any number of events contained within. This approach might not necessarily scale well and could easily lead to a bottleneck when hundreds or thousands of stories need to be reviewed. The current implementation fulfils the purpose of enabling an initial assessment of whether the principle of antonym insertion

has some merit. This assessment is part of the experiment discussed in Section 8.2, that compares CROSCAT with and without antonym insertion. For the scale of this experiment, manual antonym annotation is feasible and sufficient.

7.4 Algorithm Implementation

In the CROSCAT implementation, the story graph is represented in the way that tree data structures usually are in object oriented programs, i.e. as a series of interlinked node objects. Each node object encapsulates the data describing the composition of a single comics frame (see Listing 7.1) and furthermore maintains a pointer to its parent and to a list of children nodes. For leaf nodes this list is empty. The system only retains a pointer to the root node. The back story selection algorithm is currently implemented in a simple fashion. Every time a new story is submitted it is first inserted into the graph, at which point the scores for all nodes in the graph are recalculated. As shown in Listing 7.2, this involves three stages, namely 1) recursively resetting the scores and scoring prerequisites of all nodes to zero, 2) calculating scoring prerequisites for each node and 3) determining each node's final score. After every node was assigned a new score in this fashion the highest scoring node is determined by another recursive search through the tree. The highest scoring node that is returned by this search is then chosen as a new branching points. The shortest path from the root node to that node is appended to the back story that is being served to the next crowd worker.

Listing 7.2: Top-level view of CROSCAT scoring algorithm (Java code snippet)

```
public class StoryGraph {

    /** the root node of the graph */
    private StoryNode rootNode;

    ...

    /** calculate new scores for all nodes in the graph */
    public void rescoreAll()
    {
        rootNode.resetScoreRecurse();
        rootNode.calcScoringPrerequisitesRecurse();
        rootNode.scoreRecurse();
    }
}
```

As they are too long to show here, the concrete implementations of the three methods discussed above (i.e. those called by *rescoreAll* in Listing 7.2) have been moved

into Appendix E. Here a general description of the implementation shall suffice but consulting the relevant source code in the above mentioned Appendix is recommended. The implementation for resetting the score is a straightforward recursion from the root over all nodes and thus has a runtime of complexity of $O(n)$ where n is the number of total nodes in the tree. The calculation of scoring prerequisites is more interesting. Firstly we recurse top-down through the graph starting from the root node, recording for each node its distance to a branching point in the front and the number of branching ancestors. This operation is once again of $O(n)$ complexity.

However, whenever a leaf node is reached a backwards iteration from that leaf to the root node is started, during which all nodes along this path record their distance to that particular leaf node and their distance to a branching point in the back. Therefore each non-leaf node is visited multiple times during the prerequisite calculation: once in a top-down direction, but also once in a bottom-up direction for each direct path from that node to any leaf node. This bottom-up recursion is required to determine heuristic component R_{DL} , average distance from leafs (see Section 7.2.1). The runtime for this bottom-up recursion is $O(k * l)$ where k is the number of leaf nodes (this is also the number of submitted stories) and l is the average path length from the root to a leaf (i.e. the average length of a complete story).

Finally after all prerequisites have been set, the scoring algorithm is once again a straightforward recursion over the graph starting from the root, yielding once again in $O(n)$. This final stage is where the heuristics discussed earlier in this chapter are being applied. The variables needed for the heuristics calculation of each individual node have already all been calculated as prerequisites at this stage.

It follows that the overall complexity for the algorithm is $O(n) + O(n) + O(k * l) + O(n)$, which condenses down to $O(k * l)$. We can neglect the $O(n)$ parts of the algorithm as $k * l$ will always be at least n , in the very worst case it could be close to n^2 . In practise $k * l$ should be somewhere in between n and n^2 , with the topology of the tree (i.e. the distribution of branching points) determining whether it is closer to one or the other.

7.5 Scalability

For a system that relies on crowdsourcing, scalability is a valid concern. In our prototype implementation of the CROSCAT system however, the main goal was to build something that could be used during an experiment for evaluating the effect of Crowd Task Adaptation. For the modest number of participants which we could anticipate to be able to recruit for this purpose, scalability was not an issue. In other words, for running the experiments described in the next two chapters a more efficient implementation and an improved ability of the system to concurrently serve multiple users was

not required and would not have made a noticeable difference. If the system would be employed for an actual real-world authoring effort and a medium to large user base could be tapped into, such aspects could start to matter. Therefore we briefly discuss here how the current implementation would fare when faced with thousands of users and massive story graphs.

7.5.1 Concurrent Use

One aspect that would have to be addressed for a large-scale deployment of the CROSCAT system is concurrent use, i.e. allowing multiple authors to submit stories at the same time. This is an issue as a typical transaction with the system consists of two parts, receiving a task (i.e. a backstory to continue) and submitting the continuation and a considerable amount of time (anything from a few minutes up to a few hours) elapses between these two events. Blocking access to the system entirely for others during this period is impractical, but if we allow them to logon while still waiting to receive the story of another user then which graph are we scoring in order to chose which backstory to serve the new user(s). The results of the first user's work will obviously influence the shape of the graph and thus may impact which backstory gets selected, but we need to select a backstory for the new user(s) before this information becomes available.

The current implementation of the system ignores these problems and does not allow concurrent access. Addressing this issue was not a necessity for the experiments we conducted with CROSCAT. Specifically as described in the next chapter, for our experiments multiple stories were authored in parallel so that we could let multiple participants do the experiment at the same time and still only have a single person modifying each storygraph at any one time. This is described in more detail in Section 8.2.1 in the next chapter.

There are several ways in which a more scalable version of CROSCAT could support concurrent use. The naive approach would be to not only retain the highest scoring node in each scoring round but also the second, third, etc highest scoring node. If a second, third, etc user arrives before the graph can be rescored with a first user's contribution included, then we could use the second highest scoring node to assign the backstory for the second, third, etc user. But this naive solution is not ideal. For example the first, second and third highest scoring nodes might be all situated next to each other. If we use all three of them we get three branching points next to each other, which is something that the heuristic would have most likely tried to avoid if the nodes would have been selected sequentially. Ideally the system would need a more sophisticated algorithm that generates a list of possible nodes which should be as independent of each other as possible.

7.5.2 Scoring Algorithm Performance

The current implementation of the scoring algorithm has two weaknesses: The first is the fact that the entire graph needs to be reprocessed and traversed every time in order to obtain updated scores, no matter how miniscule the change to the graph since the last scores were obtained may have been. The second weakness is the algorithm's runtime which as we have discussed above is $O(k * l)$, where k are the number of submitted stories (leaf nodes) and l is the average story length. It follows (assuming that the average story length will likely remain constant in the long term) that the algorithm's runtime increases linearly with every additional user contributing a story. For 100s or even 1000s of users this should not be a problem, but for user numbers several orders of magnitude higher this may very well turn into a problem. Running the system on more powerful hardware may alleviate some of the problems, but the fact remains, that the runtime will continually increase for every additional contributor and the system in its current form is not highly scalable.

7.6 The CROSCAT Viewer

Compared to ENIGMA, one of the main advantages of the much more simplistic branching story tree representation employed by CROSCAT is that one can quite trivially produce a runtime engine for it and subsequently an IS artefact from it. More complex story representation mechanisms may require substantial integration effort as demonstrated for example by the FearNot! case study in Chapter 2). CROSCAT's branching story tree on the other hand contains everything needed to create an interactive story experience in the "Chose your own adventure" fashion.

Interacting with the IS artefact is essentially the act of traversing the story tree. Starting at the root node (of the common back story), we present the story frames to the audience in order. Interaction consists of letting the audience choose a path whenever a branching node is encountered during the downwards traversal of the story tree. The key question then is how the choice of different paths is presented to the user. "Chose your own adventure" stories will typically present short textual descriptions of the choices on offer such as "Do you want to take the bus or walk". However, such descriptions cannot always be easily generated automatically given a set of children nodes. A simpler alternative is to simply present the first comics frame of each sub branch to the audience and let this inform their choice.

A simple IS runtime engine using the latter approach to choice presentation was implemented for CROSCAT. Figure 7.11 shows an example of how a choice is visualized and presented in this viewer application. Implemented as a web application in PHP, it accesses the branching story tree exported by the CROSCAT server as tables of nodes (containing a unique node id and comics frame image file) and edges

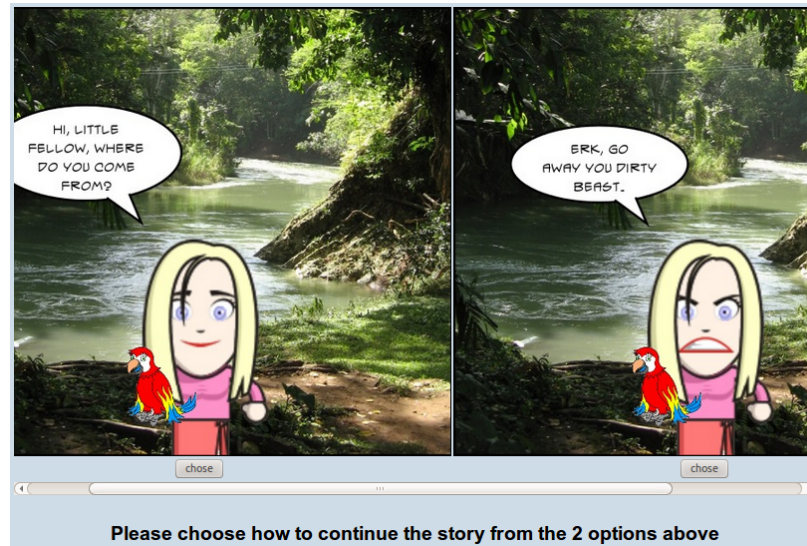


Figure 7.11: A choice in the simple web based viewer for browsing stories authored with CROSCAT.

(containing a source and destination node id). The CROSCAT story collection server can be integrated with the viewer application in such a way that without any human intervention, the IS artefact as given by the latest state of the branching story tree can be viewed.

7.7 Conclusion

After describing the ENIGMA system in the previous chapter, this chapter has shown a second, very different example of how the Crowd Task Adaptation paradigm can be applied to IS authoring. This alternative approach uses a branching story tree representation and adapts the story collection tasks that are farmed out to the crowd by changing the back stories given to workers. This essentially allows the system to decide at which point in the existing tree a new story contribution should branch off. Several heuristics for determining a suitable branching point from a given branching story tree based on its topology were discussed. We also discussed how a system's knowledge of antonyms (pairs of events with an opposite/contrary meaning) may further improve the selection of suitable branching points. This approach was implemented in a complete IS authoring system and runtime engine called CROSCAT, which forms the foundation for authoring experiments described in the next chapters. In the last chapter's conclusion we discussed how the ENIGMA system fits within the set of authoring tool requirements proposed by Medler and Magerko (2006). Much the same applies to CROSCAT, but unlike ENIGMA, it is more successful in terms of **usability**.

Part III

Authoring Experiments

Science never solves a problem without creating ten more.

George Bernard Shaw

Chapter 8

Study Descriptions

So far this thesis’ arguments for using an IS authoring process based on crowdsourcing and in particular incorporating Crowd Task Adaptation have been entirely speculative. This final part of the thesis discusses experiments that add some practical evidence to the discussion. This chapter describes the experimental design of two crowdsourced authoring studies and the research questions addressed by them.

First the “Point Nautilus” study, named after its story setting, is discussed. It was concerned with the crowdsourced collection of stories written in text form. No technology was involved in running the study and no Crowd Task Adaptation was performed. It served the purpose of establishing a baseline regarding story setting, imposed narrative boundaries and participant’s ability for creative writing free from any technological or methodological influences.

This is followed by a description of the “Seagnomes” study (again named after its story setting). This study took into account lessons learnt from the “Point Nautilus” study, which included a strong indication for using a different story setting. The “Seagnomes” study was designed to test this thesis’ hypothesis that Crowd Task Adaptation can improve the IS authoring process and result in the creation of better IS artefacts. To this end the study employs the CROSCAT tool described in the previous chapter to compare crowdsourced authoring under different Crowd Task Adaptation conditions, including one control condition where no Crowd Task Adaptation takes place. In order to evaluate the quality of the created artefacts we let users experience and rate them.

8.1 The Point Nautilus Study

In the Point Nautilus study, which was performed after the development of ENIGMA and in parallel to the development of CROSCAT, participants were given the beginning of a written story and asked to write an ending for it. All participants received exactly the same task and no Crowd Task Adaptation took place. No technology

apart from a word processor was involved in the study. This no-technology approach was chosen in order to first assess the basic potential of the style of crowdsourced authoring that this thesis advocates under simple conditions. Specifically, the study was designed to address the following questions:

- **Suitability Of Story Setting:** Does the “Point Nautilus” story setting elicit useful contributions from a crowd of authors? In particular, are example stories collected from multiple authors varied and coherent. Is the story setting a useful base for later experiments with CROSCAT?
- **Suitability Of Instructions And Boundaries:** How well are instructions being followed and imposed narrative boundaries respected by the authors contributing example stories?
- **Participant’s Capability For Creative Writing:** Are volunteering participants finding the task to contribute a story line to a dictated story setting easy or difficult? Should crowdsourced writing experiments aim to prefer recruitment of certain demographics?

Gaining insight into these aspects was deemed beneficial in establishing the foundations for the thesis’ main authoring study.

8.1.1 Story Setting

In order to perform an experiment of crowdsourced authoring, a story setting was needed. Clearly, not specifying a story setting at all and simply asking participants to write a story about anything they like would not have resulted in a collection of stories that could be combined in any meaningful way. Providing a story setting limits collected contributions by establishing story world boundaries, as was discussed in Section 4.1.2. The goal for this study was to create an open-ended narrative situation which opens up many alternative possible story paths, while also communicating clear boundaries through the following means:

- A cast of characters with established goals and motivations
- Introduction of a central conflict that requires resolution. A dilemma situation was aimed for that has no clear obvious outcome.
- The story provides a setting that naturally limits the action to a confined space.

Based on these criteria, we wrote a back story in the romance genre entitled “Point Nautilus”, which participants had to read as the first step of the experiment. The full text for “Point Nautilus” as it was provided to participants can be found in Appendix

B. Here a short summary of the story and how it addresses the above criteria is given.

Point Nautilus is the name of a small hotel on an isolated island, which is the central location in the story. The protagonist, Claire comes there in the wake of a bad breakup and ends up settling down in Point Nautilus and running the hotel. She eventually falls in love with Humphrey, a writer, who has taken up residence in the hotel to work on a novel. The couple spend a few happy weeks on the island, but eventually Humphrey has to leave for a business meeting. When he does not return to the island and cannot be located, Claire is distraught. The reader learns that unbeknown to Claire, Humphrey has an accident that causes him to lose his memory. The narrative now jumps several months forward. We learn that Claire is expecting a baby and there is no doubt that Humphrey is the father. On a stormy night, Claire is all alone on the island (her assistant Jack is on the main land and cannot return due to the storm) when she has an unexpected visitor. A woman named Susan, who had been on a sailing trip with her boyfriend shows up and asks for shelter from the storm. Claire happily agrees and Susan explains how she has until recently been a doctor and her boyfriend John a former patient suffering from Amnesia. At that moment John, who had been mooring their sailing boat enters and Claire recognizes Humphrey. He does not seem to remember her though and the back story ends with Claire feeling a sharp pain in her belly.

At the point, where this back story ends, a clear conflict is established that requires some form of resolution. The driving agent of this conflict is clearly the character of Claire as she is presented with the dilemma whether to reveal John's past. The presence of a love triangle and Claire's pregnancy add further layers and complications to this decision. The story tries to enforce narrative boundaries through the isolated island location, the storm which permits no communication with and travel to the outside world and by hinting at an imminent medical emergency, which has the potential to act as a catalytic event.

8.1.2 Execution

The study was advertised via mailing lists, social networks and word of mouth. Interested volunteers were given the link to a website, which briefly explained the purpose of the study. On the website they were then asked to read the entire Point Nautilus back story and then given the following further instructions.

- *Only use the 3 characters Claire, Susan and John/Humphrey and the island as a location. Everyone else including Jack has to stay off the island.*

- *The communications to the main land have to remain broken.*
- *Just to clarify, John/Humphrey is not faking his amnesia, he really does not remember anything.*
- *Claire has a copy of Humphrey's novel, if you want you can use that fact in your story, but you do not have to.*
- *Write in any style you like, I am interested in the plot resolution that you come up with not the language you use.*
- *Try to write at least 200 words, but you can write much more if you like.*
- *Please try to finish the story, bring it to what you consider an ending*
- *I would be very grateful if you recruit anyone you think would be interested in participating, but please don't discuss the story with them before you have both completed the experiment*

These instructions were intended to clarify the participants task and make the story world boundaries that were implicitly hinted at in the back story explicit. When finishing their story, participants were asked to email their contribution and fill out a short online questionnaire (see next Section). A time limit for completing the story was not imposed on participants.

8.1.3 Data Sources For Evaluation

Questionnaire

One source of data for the analysis of this experiment was a short two-part questionnaire that every participant had to fill out.

The first part contained questions that collected demographic information about the participant. Each participant had to declare their gender and age and using 5-point Likert items self-assess their creativity (1: very uncreative to 5: very creative) and level of English (1:basic to 5:native speaker). Furthermore we were interested in assessing a participant's previous practise in participating in creative storytelling experiences. A number of related activities were therefore identified (Write Fiction, Write Poetry, Play Video Games, Create/Design Video Games, Playing Pen & Paper or Live Action Role-Playing Games, Direct, Act and Improv) and participants were asked how often they were involved in these activities using a 5-point Likert item per activity (with the levels Never, Once or Twice, Occasionally, Regularly, Very Often). The second part of the questionnaire related to the actual activity of writing an ending for the "Point Nautilus" story. The following three questions were asked in this section:

How difficult did you find it to think of a continuation for the story? 5-point Likert item from 1:very difficult to 5:very easy.

How satisfied are you with your continuation of the story? 5-point Likert item from 1:very unsatisfied to 5:very satisfied.

Overall, did you find the constraints imposed by the original story more hindering or helping your creative process? 5-point Likert item from 1:hindering to 5:helping

Finally a box for free-form comments was made available.

Story Analysis

The second source of data is the set of written endings for the “Point Nautilus” back story. Several methods were being applied in order to analyse it. In absence of much quantifiable purely objective data being extractable from such a corpus beyond each story’s word count, manual reading and analysis of each story was being used to address the research questions. One aspect that was analysed was whether the narrative boundaries were being observed and the central narrative conflict was being resolved in each story. The corpus in its entirety was also being subjected to an analysis of variety (how many original story ideas can be found, or in other words how many stories are replicating ideas already found in other stories) and inconsistencies (how many events in one story clearly contradict those in another story). In the next chapter in Section 9.2 the importance of consistency is explained. Story consistency and variety were both evaluated through manual story analysis.

8.2 The Seagnomes Study

The “Seagnomes” study was the central experiment of this PhD thesis. Its aim was the collection of evidence in support of the Crowd Task Adaptation hypothesis. The goal of the study was to show if Crowd Task Adaptation makes a positive difference in an authoring situation. To this end it compares three versions of the CROSCAT system as its independent variable:

- **version graph:** uses the back story selection strategy without antonym insertion, i.e. with back stories selected solely based on topological features of the story graph. The back story selection algorithm uses heuristic $H_3(node)$ to score nodes with the weights set as reported in the last chapter ($w_{DB} = 2, w_{DL} = 0.5, w_{BA} = 3, w_{BF} = 4, T_{BF} = 3$).

- **version anto:** uses the back story selection adaptation strategy including antonym insertion. Applies the same heuristic as version “graph” for scoring but adds the reward $R_{an} = 100$ to each node that allows antonym insertion. The extremely high value for R_{an} was chosen to ensure antonym insertion is always carried out for this experimental condition but might not be the best choice otherwise in practice.
- **version none:** does not use any back story selection at all, i.e. always assigns the highest score to the root node of the graph, which results in always serving the same initial back story.

With the thesis’ emphasis on Crowdsourcing, readers may expect this to be a large-scale experiment with hundreds of participants, which it should be noted it is unfortunately not. While more is almost always better in the case of experimental sample size, the realities of participant recruitment and the non-trivial time investment that participants were asked to make, meant that it was not possible to arbitrarily scale up the experiment. We contend however, that a large sample size is not an absolute requirement for this experiment. Specifically, if experimental effects (i.e. differences between the above experimental groups) are being observed then our usage of statistical tests determines the significance of the effects taking into account the sample size. A low sample size can of course mean that weaker effects are more easily missed, but the effects that are found to be significant are as valid for low sample sizes as they are for higher ones. Thus we could hope to derive useful conclusions from this modest-scale experiment. As the next chapter demonstrates this hope was not unfounded and a statistically significant experimental effect was indeed found.

8.2.1 Participant Assignment

The CROSCAT server, which was setup for the experiment therefore maintains three different, collaboratively created story graphs. Apart from the differing strategies for choosing the back story presented to the client, the three versions are identical. The same story setting with the same pre-authored initial back story is used by all three versions.

Due to their greater statistical power, Field and Hole (2003) recommend whenever possible the usage of within-subject, i.e. repeated-measures experimental designs, where every participant is exposed to all experimental conditions in favour of between-group designs, where every participant is only exposed to one of the conditions. Nevertheless, there were several reasons why a between-group design was eventually chosen for the “Seagnomes” study. Firstly, the time required to write a single story is already substantial and stretches the goodwill of many volunteers. Doubling or tripling the

amount of work was deemed as too demanding and fatiguing and would in all likelihood have significantly reduced both the number of volunteers willing to participate in the study and the quality of contributions. Furthermore, the ordering effect for this particular task was expected to be extremely strong. We could not expect a participant to write several stories based on the same story setting and keep up their level of creativity and motivation. The only sensible way to counteract this problem would be the use of three different story settings for the three different CROSCAT versions. While this would be expected to remove most of the ordering effect, it would also introduce an additional independent variable (story setting). Observed effects could now no longer be clearly attributed to the CROSCAT version, as the story settings differ too. While this problem could be addressed through a 3x3 latin squares design, the resulting 9 experimental conditions would negate any statistical power advantages gained by using a repeated measures design in the first place.

Consequently, the server assigns participants to only one of the three versions upon initial connection. If all three versions have the same running total of participants, a random version is chosen, otherwise, in order to maintain a balanced number of participants across versions, the version with the lowest number of participants is always preferred. Additionally versions “graph” and “anto” may be locked in which case they are not eligible for selection. Version “none” is always available, so no participant would have to be rejected.

The locking of versions has two reasons. First we do not allow concurrent access by multiple clients to a single version (except version “none”). Consequently a lock is placed on versions “graph” and “anto” while a participant is connected, i.e. during the interval between logging into the system and submitting the completed story. This is a measure to ensure that the back story modification can operate under optimal conditions. When the back story for a new participant is chosen, we can guarantee the selection algorithm operates on a graph that takes into account all previous contributions. For version “none” this is not necessary as the selection algorithm always chooses the same result anyway.

Additionally, version “anto” is also locked due to the necessity of manual annotation of antonyms. Whenever a new story is submitted to version “anto”, the lock remains until the experimental operator (notified via email) had time to finish the manual antonym annotation for this story. Only then is the lock removed and the CROSCAT server ready to rescore the entire story graph and make an informed decision on which back story (most likely with an inserted antonym) should be selected for the next participant using this version.

8.2.2 Story Setting

In our discussion of the “Point Nautilus” story setting in section 8.1.1, a number of criteria for an ideal story setting were proposed. Without pre-empting a detailed analysis of the “Point Nautilus” study in the next chapter, an apparent shortcoming of the “Point Nautilus” story setting that was found during the analysis should be mentioned here: We found that the “Point Nautilus” back story did not elicit as much variation in story contributions as was hoped for. It appeared as though the central conflict established in the “Point Nautilus” back story limited the further possibilities for narrative development too much. Expressed in terms of the flying wedge model (see Figure 7.4) by Laurel (1993), the “Point Nautilus” story setting includes too many of the decisions that narrow the space of possible narrative futures already in its back story.

Based on this finding, a different story setting was sought for the main authoring study, which while imposing a similarly strict set of narrative boundaries would not already have established a narrative conflict up to the point where certain resolutions become inevitable. The eventual story setting was called “Seagnomes” and shares the use of an isolated island setting with “Point Nautilus”. The full back story for the “Seagnomes” story can be found in appendix C. As this story was served to participants within the CROSCAT authoring tool, it is written as a comic. Below a short summary of the back story is given.

The chief of the Seagnomes nation has taken his four adult children on a boat ride. When they are out on the ocean, he reveals to them his desire to retire and for one of his children to take over his leadership role. He then produces a magic bell and by ringing it summons an island that rises out of the sea. The chief explains that this is the island Badingo, a magical place that is subject of many Seagnome legends and that tradition dictates that it is Badingo that will chose the next leader. With this information the chief leaves the astonished children alone on the island without any clear task and only certain in the knowledge that they will remain there until one of them has been revealed as the leader.

While this story setting also confines a small cast of characters to an isolated island location, the initial situation is much more open ended than that of the “Point Nautilus” story setting. An already established conflict was avoided but a narrative goal to guide story development was provided (the island has to chose a leader). As this study utilises the CROSCAT authoring tool, narrative boundaries did not only have to be enforced through the story setting related via the back story but are also implicit in the graphical material (not) made available to the authors. The only usable characters were the chief’s four children. The available five locations (see

Figure 8.1) and seven objects (Banana, Chocolate Bar, Monkey, Parrot, Stick, Sun Cream, Bottle of Water) were based on generic associations with the deserted mythical island setting. While it was not a conscious choice, in hindsight the fact that Point Nautilus’ realistic setting was traded with a magical one was probably also a contributor to the more open-ended nature of the Seagnomes scenario. Importantly in their choice we deliberately tried to avoid thinking about possible plots and choosing items in service of this plot. Instead the goal was to create both through the back story and the available material library the conditions for a kind of small-scale open-world narrative playground.



Figure 8.1: Available locations for the Seagnomes Story: two beaches, a cave, a river bank and a temple

8.2.3 Execution

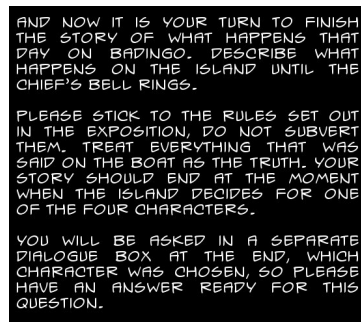
The study was performed online and allowed anonymous participation. Interested volunteers were provided with a link to a website that explained the task involved in the study. The following text was used to explain the task:

What exactly is the experiment about?

Simply speaking, you will be asked to read the beginning of a story (the back story) and “write” the ending for it. Both the first half you get to read

and the second half you write yourself are in the form of a comic. You will use a piece of software called CROSCAT (a sort of comics editor) for both reading/viewing and writing. After you have finished the story you will be asked to fill out a very short online questionnaire. The broad theme of the experiment is “collaborative creative writing”, the use of crowd-sourcing for exploring the possibilities of a story universe. A server collects and analyses all the comic stories created by the participants of this experiment. When a new experiment participant (i.e. you) logs into CROSCAT, the system will construct a new comics back story based on the stories it has collected so far. Thus the back story is different for different participants and for this reason it is really important that you complete the experiment in a single session.

The website also contained a short reference manual on how to use the CROSCAT authoring tool (see Appendix D for the entire manual shown to participants.). Reading these instructions was optional. Installation instructions for the CROSCAT client were also provided. Participants would then start the client software. This would in the first instance open the back story viewer. As reading the back story is a mandatory part of the experiment, a timer was added that ensured every comics frame of the back story was displayed for a minimum of three seconds. After the participant had seen the entire back story (initial pre-authored back story plus possible back story extension) a final frame with instructions was inserted (see Figure 8.2).



AND NOW IT IS YOUR TURN TO FINISH
THE STORY OF WHAT HAPPENS THAT
DAY ON BADINGO. DESCRIBE WHAT
HAPPENS ON THE ISLAND UNTIL THE
CHIEF'S BELL RINGS.

PLEASE STICK TO THE RULES SET OUT
IN THE EXPOSITION, DO NOT SUBVERT
THEM. TREAT EVERYTHING THAT WAS
SAID ON THE BOAT AS THE TRUTH. YOUR
STORY SHOULD END AT THE MOMENT
WHEN THE ISLAND DECIDES FOR ONE
OF THE FOUR CHARACTERS.

YOU WILL BE ASKED IN A SEPARATE
DIALOGUE BOX AT THE END, WHICH
CHARACTER WAS CHOSEN, SO PLEASE
HAVE AN ANSWER READY FOR THIS
QUESTION.

Figure 8.2: Instructions appended to back story in the Seagnomes study

As these instructions state, the CROSCAT software was modified so that upon story submission, participants had to explicitly declare in a mandatory multiple choice dialogue which character was the “chosen one”. After submission of their story, participants were asked to fill out a short online questionnaire.

Antonym Annotation

A submission in the “anto” experimental condition triggered a notification email to the experiment conductor (the author of this thesis) that manual antonym annotation was

required. The protocol for this process involved the conductor carefully reading the submitted story frame by frame, identifying any events that are of dramatic relevance and have a clear antonym (see Section 7.3 on a discussion what this entails), creating said antonym events and submitting them to the database. While this is necessarily a subjective process, we do not consider this a weakness of our evaluation: if the annotated events were not good antonyms we would only weaken our own hypothesis.

Bootstrapping

As explained in the previous chapter, due to the way CROSCAT’s back story modification algorithm calculates scores, it requires the presence of at least one branching point. This means that no matter which version of the system is used, it is guaranteed that for the first 2 submissions only the initial pre-authored back story will be shown. For the evaluation this represents a problem, as we want every participant in versions “graph” and “anto” to experience back story modification. The problem was addressed by overriding the random version assignment for the first two participants and instead assigning them to version “none”. After these first two stories were collected and a story graph with a single branching point (at the root node) existed, this story graph was copied and used to bootstrap all three versions of the system. For the “anto” version, antonyms were first manually annotated for these first two stories. Only then did the participant assignment protocol as described earlier start to apply. For evaluation purposes the data gathered about the authoring experience of these first two participants was only counted towards version “none”.

8.2.4 Data Sources For Evaluation

Logging

For every participant usage data of the CROSCAT client software was logged and collected. This includes the duration of the user’s session within the software and a timestamped list of every event created, edited, moved or deleted and every undo / redo action.

Questionnaire

Participants were asked to fill out a questionnaire, similar to that used in the “Point Nautilus” study (see Section 8.1.3). The demographic questions asked were exactly identical and the three questions about the writing experience itself were also kept with minor rewordings to reflect the fact that the medium of comics was being used. Additionally a few questions about using the software and the medium of comics were added:

Did you have problems operating the CROSCAT comics editor? 5-point Likert item from 1:lots of problems to 5:no problems at all.

Would you have preferred to continue the story by drawing a comic by hand instead of using software? 3 choices (Prefer hand drawing, prefer software, no preference).

Would you have enjoyed this experiment more if it was using only words (i.e. writing the end of a short story) as a medium instead of comics? 3 choices (I would have preferred to write, I enjoyed creating a comic more than I would have enjoyed writing, no preference)

Do you think you would have created a better story if you were writing instead of using the comics editor? 4 choices (Yes I think the story would have been better, No I think the story would have been worse if I had to write, I don't think it would have made a big difference, I don't know)

As in the “Point Nautilus” questionnaire, a free-form text box for comments was provided.

Story Analysis

Despite being in comics form, the collected corpus of stories (or more accurately the three corpora) could be subjected to a similar analysis as the collected “Point Nautilus” texts (see Section 8.1.3). Furthermore in this study we can also inspect the topology of the story graphs created by the three versions as an additional source of data.

Interactive Story Ratings

For the “Seagnomes” study, ratings of the produced IS artefacts were collected. Each of the three experimental conditions resulted in a separate story graph. Via the CROSCAT Viewer (see Section 7.6) a navigable interactive version of each of these story graphs is available. Ratings were collected by letting a separate group of subjects interact with these IS artefacts after their finalization, i.e. after the authoring part of the “Seagnomes” study was concluded.

Between-Group Design Initially an online experiment using a between-group design was used to obtain these ratings. Subjects were told their task was to “read and rate an interactive comics story”. Each subject was randomly assigned one of the three IS artefacts. Whenever a choice, i.e. a node with multiple children was encountered, the subject had to not only pick one event to proceed, but also rate

this choice on a 5-point Likert item (1:interesting choice - 5:boring choice) with the accompanying description “**Please rate this choice on a scale of 1 - 5**”. Once choices were committed they could not be reverted (In order to ensure each subject’s experience is comparable). Eventually the subject would reach a story ending, i.e. a leaf node in the graph. At this point the following two further questions were being asked to gauge each subject’s impression of the quality of plot and their sense of agency..

Are you satisfied with the way the story ended? 5-point Likert item from 1:Very Much to 5:Not At All.

Did you feel that you could influence the course of the story through your choices? 5-point Likert item from 1:Very Much to 5:Not At All.

A free form text box for comments was also provided. This concluded the story rating experiment. In order to identify replays by the same subject, which might adversely affect ratings, a cookie based browser session id mechanism was used for filtering out repeat visitors.

Within-Subject Design After conducting the between-group version of the story rating collection, we came to the realization that a within-subject design may be more suited to this particular scenario. In the between-group design, subjects had no reference grounding points to base their ratings of story lines, choices and influences on. And unlike the actual story creation part of the study, the story rating section can be easily implemented as a within-subject design. We therefore also collected within-subject story rankings as an additional source of data. Unlike the between-group design this experiment was not web-based but ran locally on a specially prepared PC. This was solely done in order to avoid having to implement the mechanisms for a robust web-based execution of the below protocol. Its adherence was instead ensured through manual preparation and supervision.

Each participant was experiencing all three created IS artefacts from beginning to end. The interaction once again used the CROSCAT viewer and was identical to the between-group design, except that ratings for individual choices were not collected. The order in which the three stories were experienced was randomized for every participant. After participants had experienced all three IS artefacts they had to rate each of them according to the following three questions.

How entertaining did you find the plot? 5-point Likert item from 1:Very Boring to 5:Very Entertaining.

How interesting were the choices you were given? 5-point Likert item from 1:Very Boring to 5:Very Interesting.

How much did you feel you were able to influence the course of the story through your decisions? 5-point Likert item from 1:No Influence At All to 5:A Lot of Influence.

By applying the same question to all three experimental conditions, the scales are contextualized and gain meaning. In effect the task of rating becomes one of ranking (with equal ranks being allowed). During the ranking stage, participants were allowed to refer back to the three stories they had read and the choices they had made in each of them. Some participants also gave some additional oral feedback after providing the story rankings.

8.3 Data Analysis Plans

8.3.1 Primary Research Question

The primary research question that was the main motivator for conducting these studies is whether Crowd Task Adaptation (in particular the back story selection strategy) can improve the quality of IS artefacts assembled from crowdsourced data. Answering this question through analysis of the “Seagnomes” study results also addresses the question if back story selection with or without antonym insertion is more effective. The primary instrument for answering this question is a statistical analysis of the story ratings. The prior hypothesis was that we would find the following:

- **Quality of plot** ratings are not expected to be statistically different across all three version. The Crowd Task Adaptation feature in CROSCAT was designed to distribute branching points, i.e. choices but should have no bearing on any individual plot line.
- **Interestingness of choices** ratings are expected to indicate the following differences across versions: “anto” (most interesting) > “graph” and “none” (both equally less interesting). We hypothesize that choices are more interesting if they are relevant and dramatic. Version “graph” does not guarantee this any more than version “none”, but in version “anto” it should be the case.
- **Level of influence** ratings are expected to obey the following ordering: “anto” (most influence) > “graph” > “none” (least influence). This expectation is based on the hypothesis that the overall number and spacing of choices is more important than the number of options provided. Version “none” would contain one gigantic choice with many options while “graph” and “anto” would both be expected to contain more frequent choices with less options. Furthermore we expect version “anto” to fare even better than version “graph” as making more

dramatically interesting choices should be beneficial to promoting a feeling of narrative influence.

The non-parametric Kruskal-Wallis (for comparing the between-group ratings) and Friedmann tests (for comparing the within-subject ratings) were employed to test for these hypotheses and their statistical significance. Non-parametric tests were chosen in order to avoid the issues surrounding an ongoing disagreement in the field of statistics; the question whether it is valid to treat individual Likert items as scale data (Field and Hole, 2003). The Kruskal-Wallis test is the non-parametric equivalent of an ANOVA (analysis of variance) that is suitable when testing one dependent variable for three or more levels of a single nominal independent variable. The Friedmann Test is the non-parametric equivalent of a Repeated Measures ANOVA.

In addition to these quantitative approaches, some additional more qualitative perspective on this question may be gained through analysis of the authored stories and the comments and feedback that were collected from the subjects, who provided the story ratings.

8.3.2 Secondary Research Questions

The data collected during the authoring studies can also shed light on several other related secondary research questions. The following questions were being addressed in our analysis.

Author Perspective on Crowd Task Adaptation

While our primary research question is concerned with the end result of Crowd Task Adaptation, i.e. the ways in which it shapes the collaboratively authored IS story worlds, we can also investigate how authors perceive Crowd Task Adaptation, specifically via back story selection. In particular we should investigate that the author's experience is not negatively affected in the "graph" and "anto" conditions compared to the "none" condition of the "Seagnomes" study. The main tool for this investigation is the authoring experience questionnaire filled out by participants of the "Seagnomes" study.

Suitable Story Settings

Comparing the "Point Nautilus" and "Seagnomes" studies allows us to draw some conclusions about the kind of story settings and situations that are important for supporting crowdsourced authoring of interactive narratives. Of particular interest are the questions of how to enforce narrative boundaries, maintain character coherence and nevertheless ensure story world variability.

Different Media: Writing vs Comics

Another aspect that differed in the two studies was the medium in which stories were expressed. Both story analysis and author opinions in the form of questionnaire results and free form comments can be employed in order to discover how the narrative medium impacts the way in which contributors tell stories.

Authoring Tool Design Aspects

Some information about the usability of the CROSCAT authoring tool can be inferred from the analysis of its usage logs and some of the questionnaire results. Especially when these results are combined with our findings of the ENIGMA usability trial (see Section 6.4), we may gain some insight into the general design for authoring tools.

8.4 Conclusion

This chapter has described the experimental designs for the “Point Nautilus” and “Seagnomes” authoring studies and explained their background. The next chapter focuses on presenting, analysing and discussing the results of these studies.

Chapter 9

Experimental Results And Analysis

We are now suitably prepared to present and analyse the results of the “Point Nautilus” and “Seagnomes” authoring studies, whose design and setup was described in detail in the previous chapter. This chapter starts by analysing the results of the “Seagnomes” study and discusses how they demonstrate some advantages gained by Crowd Task Adaptation and thus confirm the central hypothesis of this thesis. This is followed by discussions of secondary results taking into consideration both studies that highlight properties of stories suitable for crowdsourced authoring and lessons for the design of authoring tools.

All results presented in this chapter are provided as supplementary digital materials together with this thesis. This includes all the actual stories collected for the “Point Nautilus” and “Seagnomes” experiments, the questionnaire responses and the story ratings. Further information about these materials is given in Appendix F.

9.1 Evaluating Crowd Task Adaptation

9.1.1 Seagnomes Author Profiles

Forty-five volunteering participants that were equally spread across the 3 experimental conditions (i.e. 15 per version) contributed their stories to the “Seagnomes” study. The pool of authors had an average age of 32 and a gender split of 31 males and 14 females with genders and age evenly distributed across the three conditions (10 male and 5 female in version “none” with average age 32, 11 male and 4 female in version “graph” with average age 33 and 10 male and 5 female in version “anto” with average age 31). Figure 9.1 shows further background information about the participants, gathered from the questionnaire.

Summarizing this figure, the average participant’s self-assessed command of English language was very good although not at native speaker level and they rated their creativity as average. Most participants indicated some familiarity with video

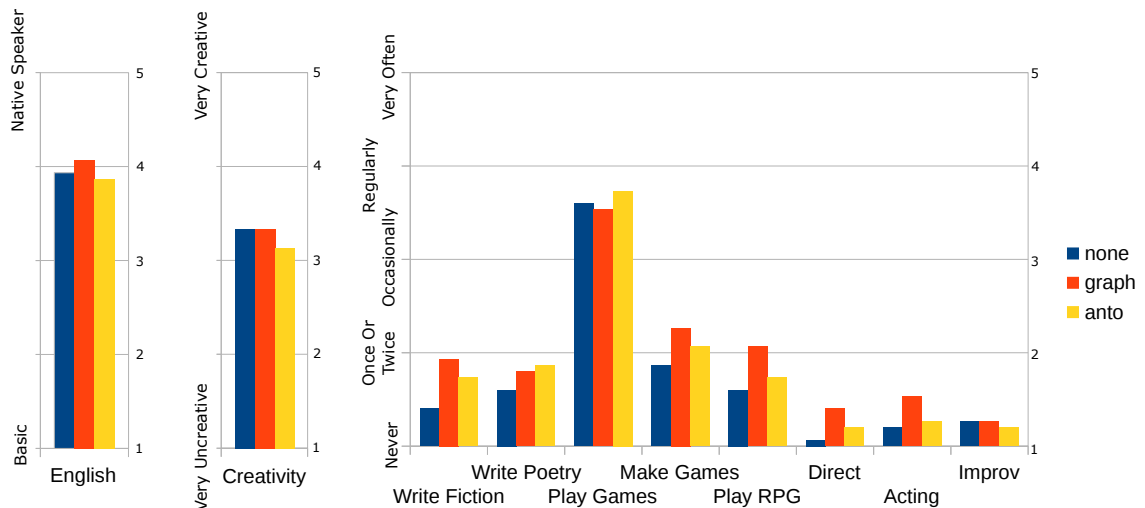


Figure 9.1: Author background questionnaire results for the “Seagnomes” study

games as players. A few participants indicated some prior exposure to writing fiction or poetry, creating games or playing role playing games. Very few participants had any directing, acting or improv experience. The graph visually indicates that there is very little variation in the author profiles across groups. Differences between groups for all these variables are only minor and not statistically significant as revealed by Kruskal-Wallis tests, which were run for all the items in Figure 9.1. This gives us more confidence that any observed differences between groups are due to the manipulated experimental condition and not attributable to e.g. a major variance in writing or English language skills across the different groups.

9.1.2 Seagnomes Story Graphs

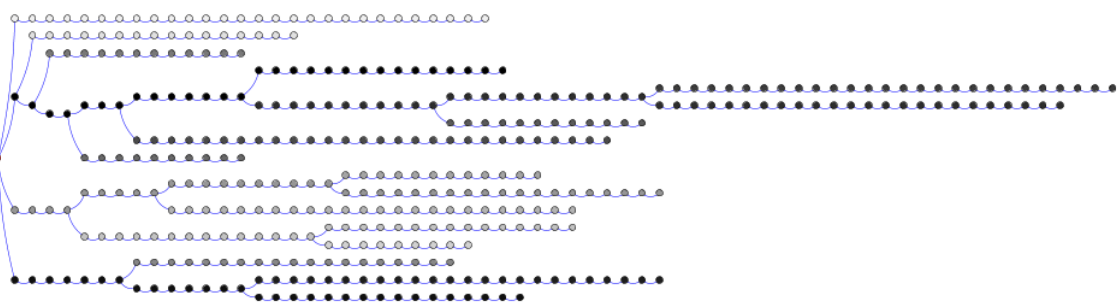


Figure 9.2: The resulting story graph for condition “graph”

Figure 9.2 shows the resulting story graph for the CROSCAT version “graph”, in which back stories were selected based on topological features. The illustration uses the same conventions as those in Chapter 7, except that the graph is displayed

sideways for readability. The root node at the left represents the entire initial pre-authored back story (see Appendix C). Every other node represents a single comics frame. For all nodes except the root node, the shade of grey indicates the temporal order of contributions. The later an author participated to the overall story graph, the brighter their contribution is displayed. The total number of leaf nodes i.e. story endings is 17 because as mentioned in the last chapter, the version was bootstrapped with a graph containing already two branches.

An informal examination of the topology shows that the back story modification strategy was applied correctly in group “graph”. Branching points are relatively evenly distributed and the result bears visual resemblance with the simulation outcome for the same heuristic, which was shown in Chapter 7’s Figure 7.7.



Figure 9.3: The resulting story graph for group “anto”

When we compare the visualisation for version “graph” to that for version “anto” shown in Figure 9.3, the weaker topological balance in version “anto” is visible in particular through the branch that has 4 consecutive branching points. A structure like that is not very likely to emerge solely based on topology. In this case it is caused by a story that has several events supporting antonym insertion clustered closely together. All branching points in version “anto” are antonym pairs, except for the branching decision at the root node (i.e. the pair of events directly following the back story), which was defined as part of the bootstrapping.



Figure 9.4: The resulting story graph for condition “none”

The graph for version “none” looks exactly as expected. Only a single branching

point exists at the root node where fifteen independent stories are split off. In fact, the only reason for it to look any different would be if several contributors by pure chance started their stories with identical events, as the system would then treat them as the same. For CROSCAT, identity would imply the exact same scene, characters, objects, facial expressions and most importantly dialogue and narration strings, which is highly unlikely.

Based purely on visual appearance then, one could conclude that the crowd task adaptation strategy of back story selection had the intended effect of placing branching points evenly in version “graph” and to a lesser degree in version “anto”, as even distribution was not the sole criterion in that case. Not using any crowd task adaptation results in a broad plot graph structure (version “none”) that only provides a single choice, independent of the scale on which we perform crowdsourced story collection.

9.1.3 Seagnomes Interactive Story Ratings

One has to admit that the conclusion above was predictable and only confirms that the system behaves as designed. It does not however address whether crowd task adaptation has improved the assembled IS artefact. We aim to answer this question by comparing the end-user ratings of the IS artefacts produced by the three versions. As described in Section 8.2.4, ratings of the three IS artefacts were initially collected in an online between-group study, where every participant experienced only one of the three IS artefacts, which was randomly assigned. 167 participants contributed to this study, of which 63 rated version “none”, 56 rated version “graph”, and 48 rated version “anto”. 140 of the 167 participants were recruited via the crowdsourcing platform Crowdfunder (51 “none”, 46 “graph”, 43 “anto”) and were being paid for their participation, while the remaining 27 subjects were unpaid volunteers.

Figure 9.5 shows the average results of these ratings. Note that while the questions regarding satisfaction with the ending and influence were asked once per participant upon reaching a story ending, while the choices rating was collected individually for every branching point and the displayed results show the average rating of all choices in each condition (63 “none”, 136 “graph”, 192 “anto”). Upon visual inspection, some of these results look promising. We can see that in version “anto”, participants on average felt they had more influence over the course of the story. We can also see that choices in version “anto” are rated on average as more interesting than for version “graph”. This is both in line with our hypothesis.

The results of Kruskal-Wallis tests (Satisfied With Ending: $H(2)=3.86, p=.15$, Choices: $H(2)=3.81, p=.15$, Influence: $H(2)=1.28, p=.53$) reveal however that none of the differences are statistically significant, taking 0.05 as the significance level. As already explained in the last chapter, we suspect that a prime reason for this unexpected

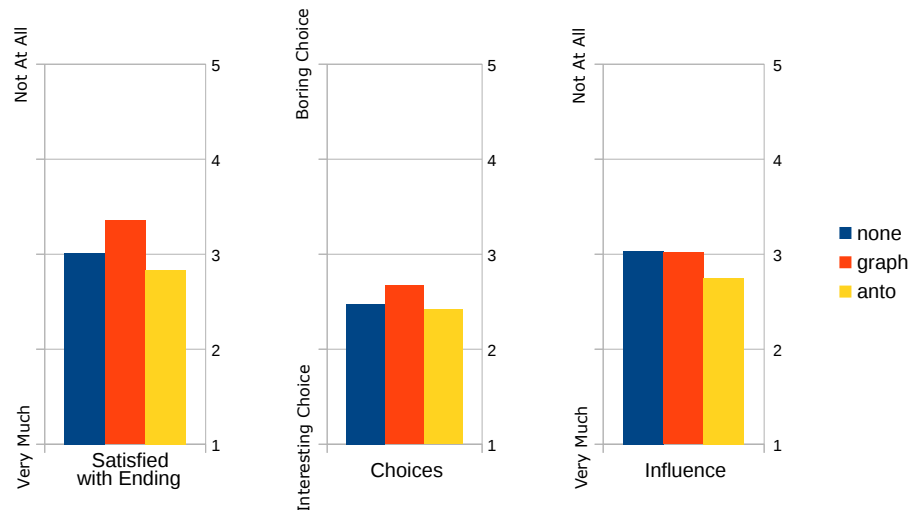


Figure 9.5: Between-Group Interactive Story Ratings

uniformity lies within the way the ratings were collected. When a participant only rates a single IS artefact, they have no reference points for the 5-point scales used in rating each story / choice.

Another collection of ratings was therefore performed using the within-subject, i.e. repeated measures design described in Section 8.2.4. 20 persons participated, each of them first experiencing all three IS artefacts in random order and subsequently rating them comparatively. Figure 9.6 shows the results. Please note that the scales are inverted compared to the results in Figure 9.5. Here the negative extreme of each scale is assigned value 1.

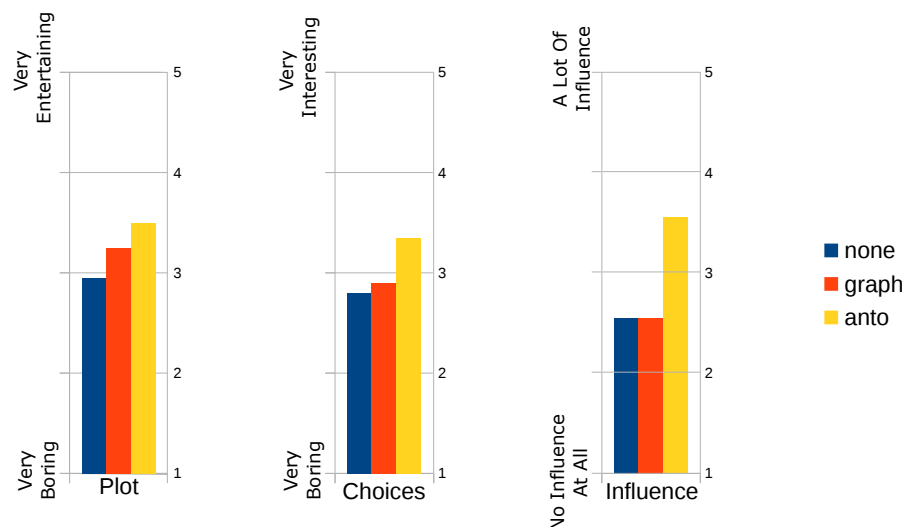


Figure 9.6: Within-Subject Interactive Story Ratings

These results show similar trends, but more pronounced. The IS artefact produced by group “anto” not only provided the highest feeling of influence over the plot

direction and the most interesting choices (both in line with our hypothesis) but also overall the most interesting plot. The IS artefact for group “graph” seems to only fare marginally better than that for group “none” in terms of choices and influence, but the two groups still exhibit some difference in story plot rating, with the plot in group “graph” deemed on average more entertaining. Applying a Friedmann test reveals that the observed differences for plot ($H(2)=1.18, p=.55$) and choice ($H(2)=2.17, p=.34$) ratings are not statistically significant, but those for influence ratings are ($H(2)=7.58, p=.02$). Again we use 0.05 as our significance threshold. The Friedman test shows that there is a statistically significant finding, but it does not show us where. Therefore the post-hoc Wilcoxon test was run on the three possible pairings of influence ratings. As expected from the bar chart, while there are no significant differences between groups “none” and “graph” ($Z=-.06, p=.95$), the differences between groups “anto” and “none” ($Z=-2.57, p=.01$) and “anto” and “graph” ($Z=-2.15, p=.03$) are both statistically significant taking into account a significance threshold of 0.05. However, in order to guard against Type I errors (i.e. false positives), it is common practise to apply a Bonferroni correction to such Wilcoxon post-hoc tests. As three pairwise comparisons were performed, the Bonferroni correction in this case adjusts the significance threshold to a third of its original value ($0.05/3 = 0.016$). Based on the adjusted threshold, the differences between groups “anto” and “graph” ($Z=-2.15, p=.03$) are no longer statistically significant, but the differences between groups “anto” and “none” ($Z=-2.57, p=.01$) remain so.

It is worth pausing for a moment at this point to emphasise the importance of this result within the context of this thesis. We have shown that a particular crowd task adaptation strategy (back story selection using topology and antonym insertion) has made a **significant positive difference** to how end users perceive the authored IS artefact. Specifically this difference manifests itself in the user’s perception of influence / control over the unfolding interactive narrative. This directly confirms the hypothesis posed in this thesis. The other two within-subject questions shown above, while not exhibiting statistically significant differences, do also point in the same direction, i.e. the “anto” group of authors having produced an overall more entertaining IS artefact than the group using no Crowd Task Adaptation. While these results are the most important outcome of this experiment, it is worth delving a bit deeper and addressing some questions that these results raise. For example, why were the differences between the groups not even greater and why did group “graph” only seem to marginally improve upon group “none”? A qualitative look at the created stories and feedback gathered from both authors and story raters can help in addressing these points.

9.1.4 Story Analysis

One feature of the collected “Seagnomes” stories that can be easily and objectively analysed is the story length, measured in number of events, e.g. story frames. Prior to running the experiment, we did not form a hypothesis on how story length may be influenced by Crowd Task Adaptation. Do contributions have a natural average length, independent on how long the back story leading up to the contribution is? Or do contributors write less when they are already given a bigger back story? The data that was collected from the experiment suggests the answer to lie somewhere in the middle. Figure 9.7 plots the length of written stories as a function of the back story length (not including the initial pre-authored back story of length 24). A linear regression fit through the data shows only a loose correlation between back story length and the length of the written continuation. Based on the collected data, the length of authored stories reduces as the back story length increases but only very slowly.

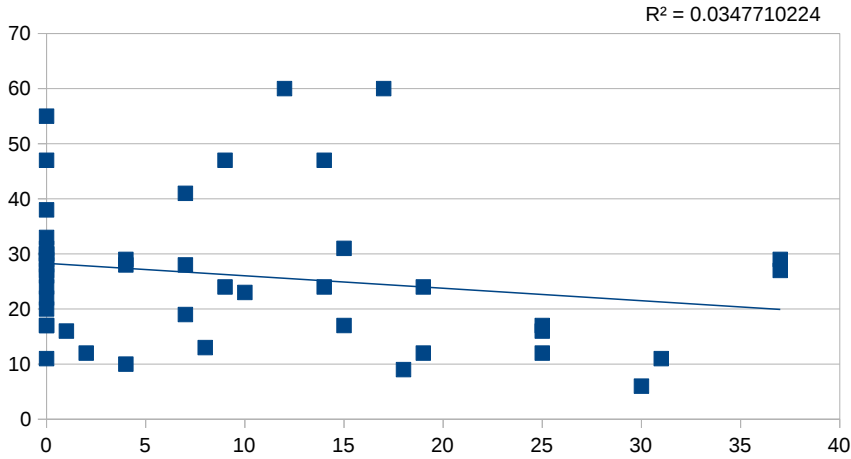


Figure 9.7: Scatter plot showing for each authored story across all 3 groups, the length (in # of events) of the back story (x-axis) and the corresponding length of the new created story continuation (y-axis). A linear fit through the data is also shown.

Table 9.1 breaks down the average length of back stories and authored stories per group, as a more accessible presentation of information also visible in the story graphs shown in Figures 9.2, 9.3 and 9.4. The data shows that while participants in group “graph” produced somewhat shorter stories than in group “none” as the result of being provided with longer back stories, the same does not apply for contributors from group “anto”, whose story contributions were as long (even slightly longer) on average than those of group “none”. As contributors in group “anto” have furthermore received the longest back stories, the average total story length (i.e. the average length from the root node to a leaf node) for the three groups seems to differ significantly. In order to verify this statistically, we first need to determine whether a parametric or non-parametric test should be applied to the data. Unlike the ordinal Likert items

we have so far statistically tested in this chapter, story length is a scale variable that can be subjected to parametric statistical tests. However, another prerequisite for applying parametric tests is a normal distribution of the tested variable. For small sample sizes as in the present experiment, the Shapiro-Wilk test is the correct statistic to test for normal distribution. Running a Shapiro-Wilk test with a standard significance threshold of 0.05 reveals that the data does not significantly deviate from a normal distribution for all three groups (“anto”: $p=.267$, “graph”: $p=.065$, “none”: $p=.153$) and can thus be further analysed using parametric statistics. A one way ANOVA ($F(2,42)=6.514$, $p=.003$) using a significance level of 0.05 shows that there was a statistically significant difference in total story length between the three groups. Post-hoc Tukey’s HSD tests at the 0.05 level of significance showed that the total story length for group “anto” is significantly higher than for both group “graph” ($p=.043$) and group “none” ($p=.003$), while the average total story length in group “graph” is only insignificantly higher than in group “none” ($p=.577$).

	none	graph	anto
back story length	0	10.733	17.6
new story length	27.933	22.333	28.2
total story length	27.933	33.066	45.8

Table 9.1: Average length of back stories, new stories (i.e. individual contributions) and total stories (sum of back stories and new stories) for all three experimental conditions.

Let us attempt to explain these numbers through some speculative qualitative analysis of the IS artefacts. A general subjective observation of the collected stories is that independent of the experimental condition many story paths initially do not contain a lot of action or drama. In fact, many stories seem to meander for a long stretch through a series of unimportant events that neither develop characters nor plot before they finally find some focus and direction. Recall that the story starts after the introductory scene on the boat, with a group of characters stranded on an island without a clear idea what to do. Given this situation, the “meandering” often manifests itself in long dialogues that only express the character’s confusion about the situation (e.g. “What should we do now”, “I don’t know”, “Neither Do I”, etc.) or in characters moving aimlessly between different locations on the island without anything of importance happening there. It is likely that the fact that stories possess this quality is a combination of three factors:

- The story setting itself promotes a long initial phase of confused orientation, which would not be very dramatically interesting.
- Authors have likely thought about the story as they went along and approached

the authoring in a real time manner. Therefore many of them may have started writing without a clear plan of where they were going and only developed this plan gradually on the fly. One of the authors in fact explicitly left a comment along these lines: *“Once I started to create the story, it went better and better. I just had to start.”*

- Finally the CROSCAT software’s interface may have had an influence as it is optimized for a linear writing style from start to end (i.e. left to right in its timeline view). While it has editing functionality that in principle allows authors to rewrite the entire beginning of the story, rewriting the beginning of a story is not as smooth an operation as appending to its end. This may have discouraged authors from editing their “meandering” story beginnings.

The “meandering” beginnings then explain the first line in table 9.1, i.e. why longer back stories were served to group “anto” than to group “graph”. As not many dramatically interesting events were happening initially, antonym annotations were not as present during these initial story sections as in later stretches. This in turn meant that events selected in group “anto” based on their support of antonym insertion were located further away from the root node than those selected in group “graph” based solely on their location within the graph. Using the same reasoning, we can also speculate why the interactivity offered by group “graph” (as expressed in ratings for choice and influence in Figures 9.5 and 9.5) was almost rated as poorly as that offered by group “none”. Given that in group “graph” branching points, whose location is chosen based on topology, are placed much closer to the root node than in group “anto”, where branching points are located in dramatically interesting places, group “graph” runs a higher risk of placing choices within the initial dramatically boring “meandering” zone. It is understandable that such branching points should not be rated as particularly interesting or giving the reader much influence over the course of the story.

We have now found possible explanations for the unexpectedly poor ratings of group “graph” (which we expected to rank better than group “none”, although not as good as group “anto”) and for the increased back story length in group “anto”. Why however, were the story continuations authored in group “anto” also longer? One possible explanation follows.

The antonym insertion mechanism in that group metaphorically speaking makes the plot take sharp 90 degree turns away from its original plot line. However in an analysis of the “anto” story graph one can find several instances, where authors resisted this steering attempt and ignored the implied change in story direction. Effectively, these authors were steering back to the original course with the antonym only having resulted in a short detour. These detours may explain the difference in length between

story continuations in group “graph” and “anto”.

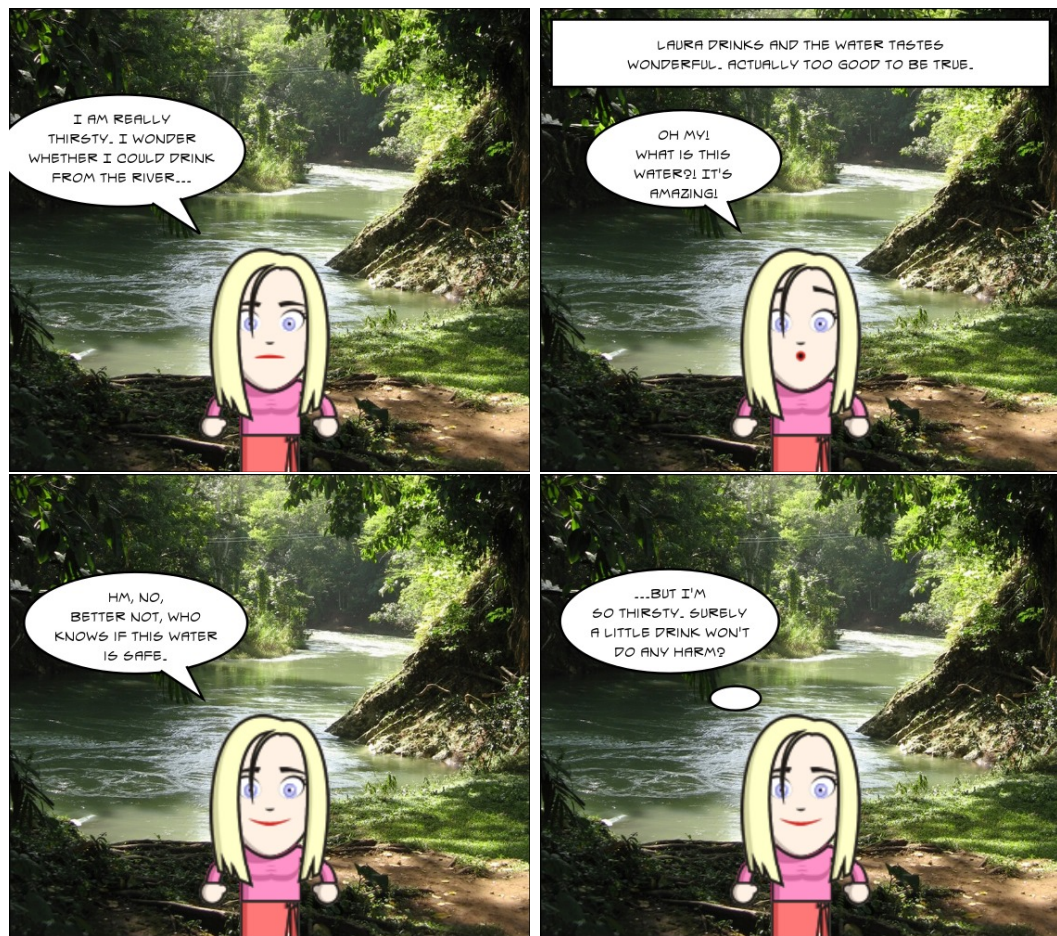


Figure 9.8: Example of an author’s resistance to being steered: top left (1): event before branching point, top right (2): event following (1) that was initially authored, bottom left (3): antonym for event (2) created by experiment conductor, bottom right (4): event following (3) written by a new author

Figure 9.8 shows an example of such a situation. In the original story the event of drinking some water, resulted in the character having hallucinations. In the antonym annotation phase this was deemed as dramatically interesting and an antonym event (the character deciding that the water looks unsafe) was created. When a subsequent author was served with a back story that ended on this antonym event, their first action was to undo it by creating an event where the character decides to drink after all. In this case there now is a 2 event long diversion in the story that has not really achieved anything.

From the end user perspective, making such choices only to see them immediately undone, may result in an unsatisfactory experience. During the within-subject story rating, two participants gave some verbal feedback along these lines, one of them referring specifically to the water drinking choice shown above. The participant said that they preferred a meaningless non-dramatic choice to one that turns out to be an

“illusion”. This might explain too then, why group “anto” was not ranked even higher. As mentioned above there are several instances of such illusory choices resulting from an author’s resistance to being steered. But as this does not apply to all choices in group “anto”, the overall feeling of end users is still that they can influence the course of the story more strongly than for the other conditions. Nevertheless, this shortcoming highlights that there may be ways in which to improve antonym insertion. For example, authors could be made aware of the purpose and function of the system, so that they more intentionally follow the path laid out for them. This relates in some ways to the notion of offers, found in improvisational theatre. An improv actor is not supposed to discard some fact that another actor has established. Instead it is part of the etiquette of improv to go along with such offers. Establishing a similar kind of etiquette and set of rules may be required for crowdsourced authoring too.

9.1.5 End User Feedback

The validity of the speculations made above is underlined by some of the comments left by the end users, i.e. the participants of the story rating experiments. In the within-subject experiment, participants gave their feedback verbally in person. The majority of them commented on the fact that the choices in group “anto” were more meaningful and made them think harder about which one to chose. Opinions on versions “none” were divided. Specifically, there were several people that liked the quantity of choices in version “none”, even though they are mostly very similar, while several others found the plethora of nearly identical choices unappealing. Several people pointed out version “graph” in particular as the one condition where the choices don’t make much sense. As explained above, two participants commented that some choices in version “anto” can be disappointing if the intended consequences of a choice being made did not materialize and that they rated version “anto” poorly for this reason.

The feedback from the between group ratings is a bit different as participants were not comparing alternative versions or even aware of the existence of different versions. As this experiment was conducted online the comments were left in a text box and unlike for those summarized above there was no opportunity to discuss and clarify them. Many participants regardless of the version they were assigned to liked the overall experience, leaving comments along the lines of “nice”, “good story”, “interesting story”. Some comments clearly underlined the poor interactivity offered by version “none”:

“Only given one choice - didn’t feel that it made an impact on the story, and that I didn’t have enough information at the time to make a ‘sensible’ choice from so many

possible ones.”

“The only choice I got to make wasn’t dramatically interesting at all.”

“I don’t understand, so the choices we made have different story scenario? i just pick the most logical one that has nice story flow.”

The following three comments all relate to version “anto”. The first two might be indications of the “resistance to steering” symptom described earlier, as they complain about the choices having no perceived effect. The third comment shows that of course improving the quality of choices in version “anto” is not ensuring a satisfying ending.

“It seems that some people, after me having made the first choice of splitting up, would never be able to become the new leader. So the first choice seems to be the most important, whereas the later choices do not matter that much anymore. Still, I’m curious to see what all the other story branches look like!”

“The story was nice but the last few instances which was asked to rate did not add to the total theme of the story. The ending was a bit low and did not put any sense on the reader.”

“It wasn’t really clear in the first choice that there was any meaningful difference between the two options. The narrative arc didn’t conclude: the story set up the conflict to be about which of the siblings would be selected to be the next chief, but never resolved this conflict.”

9.1.6 Choice Analysis

Related to the above user-feedback is a more objective analysis of the choices made by the users during the story rating experiments. Between the within-subject and between-group ratings collections there were 66 viewers of a complete story in version “anto”, 76 in version “graph” and 82 in version “none”. The difference between the number of views in the different conditions stems from the fact that in the between-group rating experiments subjects were randomly assigned to one of the three conditions. Let us examine which choices were made during these story views.

In group “none” there was a single choice with 15 options. With all choices equally popular we would expect the 82 story views to distribute evenly, i.e. every choice should have been made either 5 or 6 times. In reality the most popular choice was

made 22 times, with the remaining choices in descending order of popularity being made 10, 9, 8, 7, 6, 4, 3, 3, 3, 3, 2, 1, 1 and 0 times. The analogous data for groups “graph” and “anto” is shown in Figures 9.9 and 9.10.

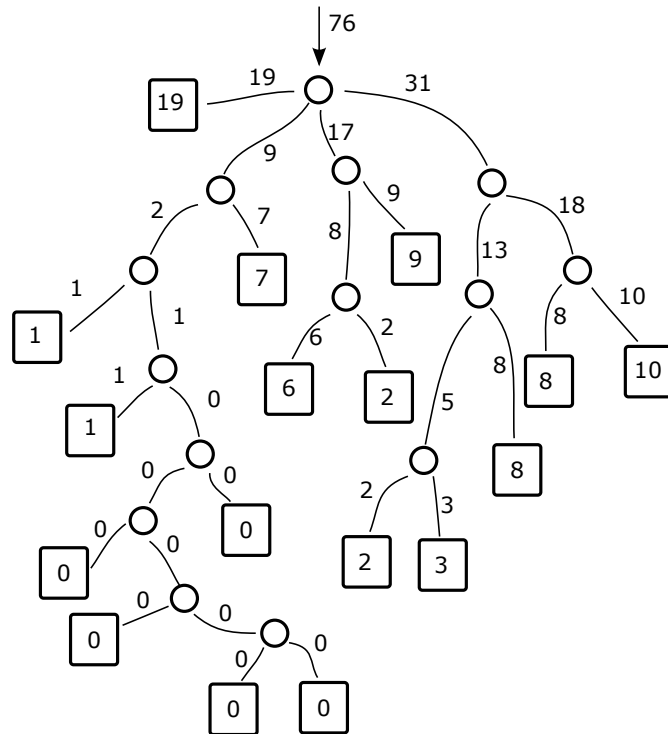


Figure 9.9: An overview of the choices made by the 76 viewers of group “graph”. The numbers indicate how many viewers have chosen a particular story path. Circles indicate choices and each box represents one of the 17 possible endings.

The first thing that stands out from this data is that in none of the conditions choices appear to have been made in a completely arbitrary, i.e. random fashion. Given that the order in which the choices were shown to viewers was randomized, if choices had indeed been made randomly, a more equal split among the available options could be expected. Instead the data clearly shows that certain favourite choice options and story paths exist in each of the three conditions. However we can also see that in none of the conditions the most favourite story path was chosen by an overall majority. In version “none” 26.8% (22/82) of viewers chose the most popular path whereas in version “graph” 25% (19/76) and in version “anto” 21.2% (14/66) did the same. We argue that in light of these numbers we can have some confidence in the results reported earlier and that they represent ratings of the complete IS artefacts rather than individual stories. An absolute majority favourite (i.e. a story path chosen by 50% or more of all viewers) on the other hand would have been more troubling. The fact that favourites would exist at all, however is not that surprising. After all the artefacts are a combination of story lines created by a number of authors with varying writing abilities. That some authors may create story frames that are more

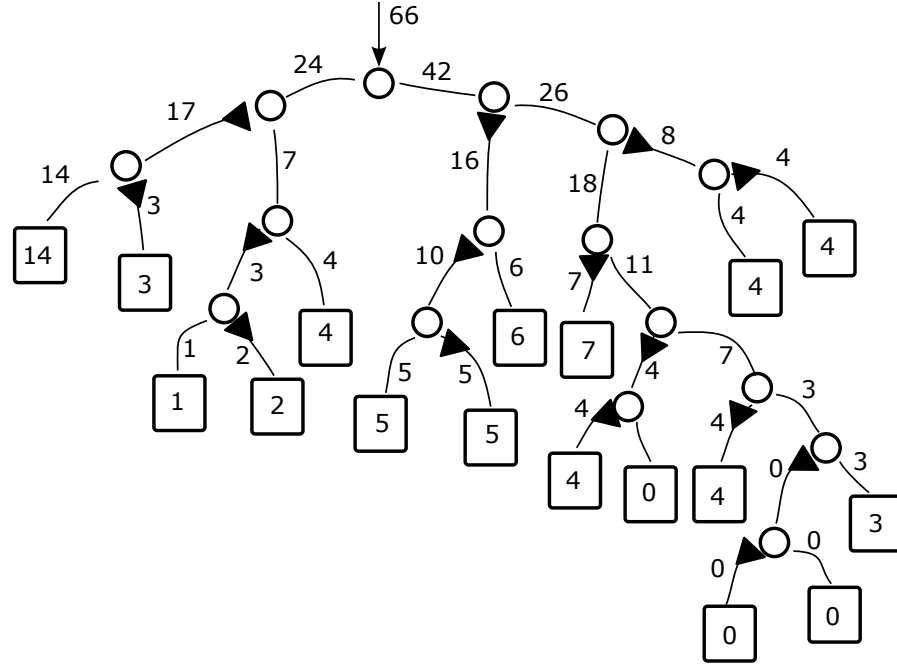


Figure 9.10: An overview of the choices made by the 66 viewers of group “anto”. The same labelling conventions as in Figure 9.9 are used. Additionally, antonyms are shown as black triangles.

captivating than others and thus preferred by viewers when presented as a choice seems only natural. Upon closer inspection, the most popular choice from the “none” condition that was chosen by 22 viewers for example clearly stands out. Most options presented at the single branching point in version “none” are similar and show a story frame where the protagonists stand puzzled and indecisive at the beach at which they have landed. The most popular option is different in that it already in the first frame shows a leader taking control and formulating a plan of action.

In the case of version “anto” the data about the choices made by viewers also allows us to confirm that the manually created antonyms were appropriate in the sense that they were neither so unappealing that they would never be selected, nor were they so appealing that viewers always chose the antonyms. We can conclude this from the fact that overall the instances where antonyms were preferably chosen (5) are fairly balanced with those where the original option was preferred (7) with both being equally popular in some instances (3). Overall 174 binary choices involving antonyms were made. The antonym option was chosen in 83 (47.7%) of these whereas the original option was chosen 91 times (52.3%).

9.1.7 Author Feedback

One aspect that we have not yet touched on is, whether the presence of Crowd Task Adaptation affected the authors in any way. E.g. was it easier / more comfortable to

create stories from the open-ended starting point of the initial back story or when one is given an already further developed back story. Authors did not leave any comments that would shed light on this question, which is perhaps not so surprising, considering that they were not fully aware of the back story manipulation and how it works. The other data source relevant for this question are some items from the questionnaire filled out by every author, specifically how difficult it was to think of a continuation and how satisfied authors were with their continuation.

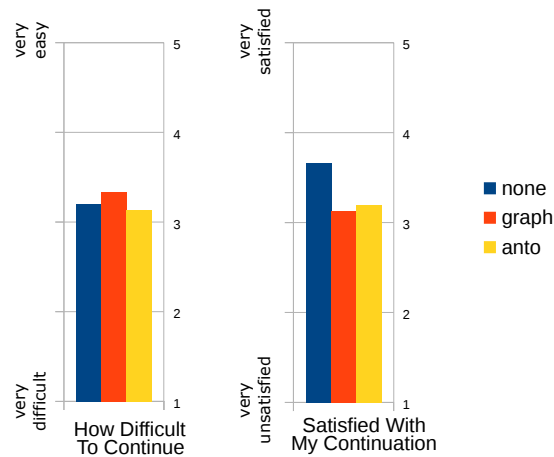


Figure 9.11: Feedback On The Authoring Experience

Figure 9.11 shows the results for these items broken down into the three different experimental groups. The result most standing out is that participants in group “none” overall seem to have been more satisfied with the stories they created, although they did not find writing any easier. However, we have also seen earlier that the separate group of raters did not rate stories in group “none” more positively than those produced by the other groups. So the group “none” stories do not seem to be of an inherently better quality than the stories of the other groups, but their authors experienced higher levels of satisfaction during their creation. This makes sense if we consider that only the participants in group “none” had full ownership over their stories (not considering the introductory boat scene). In any case, a Kruskal-Wallis test shows that these differences between groups are not statistically significant (How difficult to continue: $H(2)=.32$, $p=.85$, Satisfied with my continuation: $H(2)=3.56$, $p=.17$). Crowd Task Adaptation seemed to have had no adverse effect on the author’s experience, which is a positive outcome.

9.2 Story World Properties

9.2.1 Results of The Point Nautilus Study

Demographics

Twenty-one participants (8 female, 13 male) with an average age of 37.714 contributed their stories to the “Point Nautilus” study. Their average self-assessed creativity on a scale from 1(Very Uncreative) to 5(Very Creative) was 3.333, comparable to that of the “Seagnomes” participants, while the self-assessed level of English proficiency, 3.238 on a scale from 1(basic) to 5 (native speaker) was slightly lower than for the “Seagnomes” participants. Figure 9.12 shows the participants self-assessed familiarity with activities related to IS. This profile is almost identical to that of the “Seagnomes” participants (see Figure 9.1), the only outstanding difference being that the “Point Nautilus” participant pool was significantly less accustomed to playing video games (Mann-Whitney-U test results: $U = 328.50$, $p=.04$). This is probably explained by the older demographic (average age more than 6 years higher) of the “Point Nautilus” authors. In section 9.3 we will return to this difference.

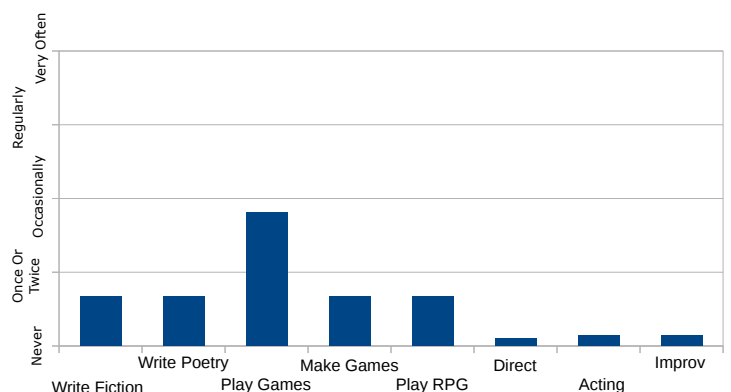


Figure 9.12: “Point Nautilus” Authors’ Familiarity With Related Activities (averages)

Story Properties

The average word count of the twenty-one stories is 566 with quite noticeable variation (std. deviation = 406) between the shortest (166) and the longest (1858) stories. The difference in length is also reflected by a difference in writing styles: while some authors used the bare minimum of words to convey the key plot points (three stories were written in bullet points), others described situations and character’s feelings in great detail. Unlike in the “Seagnomes” experiment, where a limited set of characters, locations and props was built into the authoring tool, for the “Point Nautilus” experiment it was not possible to enforce any narrative constraints and the cooperation of authors was necessary to achieve them. Authors mostly followed the instructions

given to them, such as limiting themselves to the three characters and the island location that were provided by the back story.

The main purpose of the experiment was to find out if, given the back story and the constraints given by the instructions, a corpus of stories could be collected that exhibits both variety and consistency. By this point in the thesis it should be clear, why we care about variety: the goal is to construct an entire story landscape by sampling as many individual story paths as possible and that requires the samples to differ from each other. It is, however, worth explaining the consistency requirement further and what we mean by it. All the stories that were collected in the “Point Nautilus” experiment start at the same moment in (narrative) time, the point at which the back story ended, i.e. the point when Claire recognizes John as Humphrey. We consider a set of collected stories to be consistent with each other if they believably depict parallel universes that separated only at that forking moment. This means for example, that what is revealed of a character’s past in one story needs to be consistent with what is revealed in another one. Characters should also behave consistently, and if they exhibit a different personality in one story this must be explained by previous events in that same story that believably led to a personality change. There are two reasons for desiring such a form of consistency across the collected story corpus. Firstly it enables a special kind of rereadability (Mitchell and McGee, 2012) of the final IS artefact that allows a reader / player to experience a story landscape multiple times, making different decisions along the way and playing with “what if” scenarios. This possibility of exploring the consequences of choices, is undoubtedly attractive, as various movies such as “Sliding Doors” or “Butterfly Effect”, derive their entire plot from presenting alternative (consistent) universes created by a simple choice. IS offers the unique possibility to explore such scenarios freely. Secondly, a degree of character consistency in the input stories is required for some IS story representations, such as that employed by FATiMA. The learning component integrated into the ENIGMA system (see Section 6.3) for example assumes that there is a single personality model that can be learned for a character, which would not be the case if in different input stories the same character exhibits differing personalities.

So, were variety and consistency achieved in the “Point Nautilus” corpus? A subjective analysis of the corpus shows that many stories are very similar. The plots of several contributions only differ in minor details or in the ordering of some events. One of the authors, described the story landscape fairly accurately in their free form comment:

*When I read the initial story, I could envisage a state machine of possibilities for the ending and so spent time wondering about which transition sequence to follow:
Claire lives/dies*

Baby lives/dies

John realises/doesn't realise he's Humphrey

Claire does/doesn't tell John/Susan he's Humphrey

Claire reacts well/badly to news

John reacts well/badly to news

John & Claire do/don't stay together

John & Susan do/don't stay together

The differences between the majority of collected stories are mostly captured by these alternative states. This lack of variety and the obviousness of most of the collected stories is likely a direct result of the strong constraints and conflict set up by the back story and instructions. Several of the other authors also left comments that indicate this is the case:

The last sentence seems like a cliffhanger and may influence the continuation more than intended.

Many coincidences in the story, but that's not a criticism, its a story after all.

By isolating totally the island, and co-incidentally putting a doctor and a heavily pregnant woman together the natural thought would be that Claire 'should' be giving birth in the presence of a doctor.

I started by making a short list of possible scenarios and then discounted those which were dull, unbelievable, far fetched, stereotypical (mills and boon!).

While the majority of the corpus has this obvious nature, there were two stories in particular that defied expectations. In both these cases an unexpected shift of perspective took place: in one most of the back story was revealed to be a made-up bedtime story, whereas in the other one all events turned out to be a daydream. Not surprisingly then, these two stories are not consistent with any of the other ones. But even discarding these two outliers, consistency was not achieved across most of the corpus. Stories that were not consistent had characters making different decisions without any explanation. For example, in one case Susan is jealous when John's true identity is revealed, while in another story she is not and no events in either story could explain this divergence. Similarly in some stories John is still in love with Susan after regaining his memory, while in others he is in love with Claire. Claire on the other hand is angry at John in some stories, while in others she is immediately in love with him again and in a third set she is indifferent and happy to leave those events

behind her altogether.

We can conclude then that in the “Point Nautilus” experiment, the attempt to enforce consistency only via the means of back story and instructions failed and that furthermore the severity of these constraints has significantly limited the creative variety of the collected stories.

9.2.2 Comparison of Point Nautilus and Seagnomes stories

As mentioned in the last chapter, the “Seagnomes” story setting was designed to prevent the shortcomings that were found in the “Point Nautilus” experiment, as discussed above. A scenario was needed that allows for more variety in the produced stories. With the “Seagnomes” setting this was attempted by avoiding a back story ending on a cliffhanger as mentioned above. Only a goal for the story but no central conflict was established and characters were only minimally developed during the back story, thus effectively being blank sheets.

Whether this has paid off then in terms of variety is not an easy question to answer. As the scenarios and modes of authoring are so radically different, and not even the same amount of stories were collected for both of them, a comparison between the “Point Nautilus” and “Seagnomes” experiments can not be conclusive and decisive, but only provide some cautious indication. This indication does however point to the answer that yes, indeed there is more variety in the “Seagnomes” corpus than in the “Point Nautilus” corpus, also after taking into account the corpus relative sizes. While in “Point Nautilus”, nearly every story revolved around Claire’s pregnancy and Humphrey’s amnesia, “Seagnomes” stories turned out very differently from each other. Here are just a few one-sentence summaries of some collected “Seagnomes” stories, illustrating the variety of plots:

- The island creates various situations that test which of the siblings has the purest heart.
- The island is full of deadly traps and a test of survival, with the last man/woman standing winning the contest (in a variant of this, everyone is alive again, once the test is over).
- The island is a test of intelligence and the person that solves some riddle wins.
- The siblings need to work out their differences in order to pass the test.
- One of the male siblings has always been a secret agent, serving the deity that rules the island and betrays his siblings once on the island.
- Some of the siblings turn to murder in order to “win” the contest.

At the same time, however, as should also become clear from the above summaries, the “Seagnomes” story corpus exhibits even less consistency across stories than the “Point Nautilus” corpus¹. Given that the characters are blank sheets this is not surprising as an author has to invent personalities, motivations and personal histories for them and it would be a coincidence if they were matching up in different stories. The choice of sacrificing consistency in the “Seagnomes” experiment was a conscious and voluntary one. As the “Point Nautilus” results have shown, achieving the strict kind of consistency we discussed earlier is difficult, even when exhaustive back stories for all characters are provided. For the “Seagnomes” experiment, consistency was neither a requirement for the underlying story representation, nor was the “what-if” style of rereadability a necessity.

Having provided a back story without a clear conflict on the other hand was likely a contributing factor to the observed initial “meandering” in many of the collected “Seagnomes” stories. There thus seems to be a difficult trade-off between too much and too little established conflict in the story setting, but further research outside the scope of this PhD is needed to make definite conclusions on this topic.

9.3 Authoring Tool Design Lessons

9.3.1 User Interface

Judging by the fact that all participants of the “Seagnomes” experiment managed to create stories that incorporate most of the available software features (narration boxes, speech bubbles, different locations, characters, objects and facial expressions), one may conclude that the user interface coupled with the provided tutorial have been sufficiently intuitive. The same cannot be said for the ENIGMA system, as mentioned in the previous discussion of the ENIGMA usability trial in Section 6.4. The simplifications made in transforming the ENIGMA to the CROSCAT system, e.g. allowing the storytelling to focus on visuals and natural language, without having to deal with symbolic representations, seemed to have paid off then.

However, the “Seagnomes” experiment also highlights areas for further improvements, with participants answering the question about whether they encountered any problems operating the CROSCAT software (5-point Likert item from 1:lots of problems to 5: no problems at all) with a mean rating of only 3.35. Fortunately many participants elaborated on this point in their free-text comments and suggested features for improving usability. The most requested of these features (mentioned by 7 participants) was to allow multiple speech bubbles in a single frame. For future authoring systems

¹To clarify, this is a different discussion from the crowd task adaptation one. In the “Seagnomes” story graph, each individual story path is in itself a consistent story, consistency in the current discussion is concerned with the differences between alternative story paths.

based on comics this should be a relatively simple feature to add that apparently contributes positively to the user experience.

Related to this is a request for a finer grained, more direct and realtime control of the visual appearance of the final frames (mentioned by 5 participants). In the CROSCAT system all layouting of the comics frames is handled by Alves et al. (2008)’s comic generation system. For example, one participant mentioned: “I was annoyed that I cannot see the picture I am currently creating.”. The other 4 participants were more specific and specifically mentioned positioning of characters and items and in one case determining the camera angle in which the scene is displayed. When designing CROSCAT, we thought that relieving the user of low level layout decisions would be a useful abstraction, but possibly did not give enough consideration to the advantages of controlling the visual appearance details. For future systems, a hybrid solution that applies a default layout but allows interested user to change the frame’s appearance should be considered.

Four users commented on the fact that story-level editing (i.e. inserting or moving frames) was too cumbersome. This is certainly a valid criticism, as in CROSCAT new events are always appended at the end of the story and then have to be moved manually to the front, one position at a time. Therefore inserting a single event may be achieved but requires several clicks. Moving an entire section of story quickly becomes infeasible, as movement operations can only be applied to individual events. This feature may not only have had cosmetic consequences, but is a possible contributor to the above mentioned meandering of the created stories. It may have simply been too much work to edit stories, trimming out overlong passages. Therefore better story-level editing facilities would be the highest-priority recommendation for future development of the CROSCAT system. Finally, the integration of a spell checker was recommended twice. Indeed several of the collected stories contained spelling mistakes, so we acknowledge that a spell checker would be a useful additional feature.

9.3.2 Content Library

The by far most frequent comment of the CROSCAT users was that they would have liked more graphical content to assemble comic frames from. This wish for a bigger library of the CROSCAT system’s story building blocks (specifically mentioned were more facial expressions, alternative body postures, more items and more locations) was mentioned by 12 of the 45 users.

Several users not only expressed a wish for more content but also were interested in adding their own content. The advantage of the comics medium is that such a feature can be realized technically without too much effort, especially for items and locations. A minimal implementation would allow users to upload new image files,

resize and crop them and after giving them a name they could be incorporated into the pool of available content. Supporting user generated characters would be slightly more difficult though, as character resources consist of a single body and multiple matching heads (one for each facial expression).

Allowing user generated graphical content might introduce several risks. For one, the story constraints (e.g. in the case of “Seagnomes” remaining on the deserted island) can no longer be enforced as rigidly. This may not be a problem though, if one can rely on the cooperation of the crowd. Similarly uploaded images may be inappropriate and offensive or violating copyright law, so some additional filtering would be needed, although again this may not be an issue with a fully cooperative crowd. Finally the principal author, i.e. the initiator of the story collection will lose control over the visual appearance of the created story world. The clash of various visual styles may not lead to aesthetically pleasing results. Given these potential problems, for future systems it may be worth to experiment with larger pre-selected content libraries first, before resorting to support user-generated graphical content.

9.3.3 Accommodating Deliberation

In hindsight, the restriction imposed on the “Seagnomes” experiment participants to create the entire story in one continuous session may have affected their creativity in some cases. The spontaneity required to finish the story on the spot may not come natural to everyone. Several experiment participants related that they would have preferred to “sleep on it”. It is possible that the overall story quality would have been improved if such deliberation would not have been explicitly discouraged by the experiment. The meandering that was observed in many of the stories may also be partially attributable to insufficient deliberation time.

The reason for requiring completion of a story in a single session was a side effect of optimising the experimental conditions so as to achieve the best back story selection results as explained in the last chapter. By locking the story graph for each experimental condition after each login we avoided concurrent effects to influence the back story selection algorithm.

In practice, such a restriction may not be required and unnecessarily restrictive. By implementing a more advanced session management including the saving and restoring of a user’s session state, the system could allow users to return to their story later, without having to select a new back story for them. That their story does not contribute to the overall graph and thus does not affect the back story selection algorithm until it is completed should not cause any problems when crowdsourcing on a larger scale. Accommodating deliberation is thus a definite recommendation for future authoring systems similar to ours.

9.3.4 Storytelling Modality

The majority of users of the CROSCAT system (30) indicated that they found the use of the software comics editor preferable to drawing the comics themselves, but 8 users expressed a preference for hand drawing, while 7 indicated no preference for either. It is possible that those users who did not prefer the software were missing some of the expressive freedom offered by hand drawing. A future system that takes into account the user feedback regarding more detailed control of the comic frames visual appearance and that possibly also offers users to incorporate their own visuals may possibly help change the minds of some of this group of users.

The question regarding the use of comics versus written text produced similar results, with 31 users preferring the offered storytelling modality of comics, while 8 users would prefer writing and 6 had no preference for either modality. These results suggest that comics overall are a suitable storytelling medium for authoring tools that should be easily accessible by the masses. However, an indication that the preference for comics may be different across demographics, is given by the observed demographics differences between the authors for the “Point Nautilus” and “Seagnomes” experiment. As mentioned above, the “Point Nautilus” authors were more than 6 years older on average than the “Seagnomes” authors. We already discussed that this might explain the difference in video game familiarity between the two groups, but the fact that this age difference exists in the first place, suggests that there may be some aspects of the two different experimental tasks that appealed to a different demographic. Possibly the prospect of writing a story seemed overall more appealing to a slightly older demographic, while the use of a comics generator software appealed to a younger participant profile. However, this hypothesis could not be confirmed when comparing the mean age of CROSCAT users that indicated they would prefer comics (32.38) with that of those who prefer writing (33.37), as they only differ by one year. Whether there exist demographic-specific storytelling modality preferences for authoring tools, remains an open question that may be worth pursuing further in future work in order to understand how to tailor tools for crowdsourced authoring to the particular audience of authors that are targeted.

Finally, CROSCAT authors were also asked if they thought the story would have turned out better, if they would have been allowed to write, for which the results are presented in Table 9.2.

Note that this is a different question from the storytelling modality preference, as it addresses the quality of the results and not the experience of creating them and interestingly the answers to the two questions are not identical. They are consistent in that all 11 users that selected option 2 (story would have been worse if I had to write) were part of the 31-strong majority that preferred the comics modality. But surprisingly, the belief that one could create a better or equally good story using

Option	Count
1) Yes, I think the story would have been better	14
2) No, I think the story would have been worse if I had to write	11
3) I don't think it would have made a big difference	13
4) I don't know	7

Table 9.2: Answers to the multiple choice question “*Do you think you would have created a better story if you were writing instead of using the comics editor?*” presented as participant count per available option.

writing instead of CROSCAT was a lot stronger than the preference expressed for writing. This conflict may be explained by the constraints that are enforced when writing a story with CROSCAT. Participants may have felt that the small selection of characters / items / locations restricted their ability to express themselves creatively. Therefore, several participants while having enjoyed using the comics editor, at the same time may have had several ideas that they could not realize with the offered content and thus felt a written story, where no such constraints are imposed, would have been superior.

9.4 Conclusion

In this chapter the results of the experiments performed for this thesis were discussed. Most importantly it has presented some evidence for this thesis’ claim that incorporating Crowd Task Adaptation into a crowdsourced authoring process can make a positive difference. Specifically, it was found that subjects being exposed to both IS artefacts created using Crowd Task Adaptation (in particular a strategy incorporating Back Story Modification with Antonym Selection) and created with the equivalent process without Crowd Task Adaptation found they had significantly more influence over the course of the story in the former compared to the latter.

Besides this main result, an analysis of the experimental data also revealed several other interesting facts. For example, it was found that many stories collected during the CROSCAT experiment were at times meandering aimlessly through stretches of very little action. Several potential reasons for this were suggested, e.g. the story setting itself, the lack of initial conflict in the back story, the difficulty of editing stories in the CROSCAT software and the experimental setup that may not have allowed for enough deliberation time. We also found that several authors resisted the attempts of being steered along a certain story path by the back story. This implies that maybe participants should be educated so as to be more cooperative, similarly to actors in improvisational theatre, who follow a behavioural codex to accept “offers” from their peers.

Properties of story settings suitable for crowdsourced authoring were also discussed. In particular the amount of background information provided and of established conflict may influence the degree of variety and consistency in the collected results. We did however not manage to elicit entirely consistent contributions in any of our experiments. This is in line with the observation of Tapscott et al. (2013) that "producing consistent stories that share a common narrative space when multiple authors are involved is not a trivial task." A solution similar to their work on formal models for imposing consistency constraints may be needed to elicit entirely consistent stories. Finally, several lessons learnt regarding the design of authoring tools for crowdsourced story collection were discussed. The results suggest that comics are a suitable story telling medium for these type of applications. Given the use of comics, lower-level visual editing facilities than what is currently offered by CROSCAT seem to be important to at least some users and an extensive library of graphical materials or the ability to add custom visuals is deemed important by many. Story-level, i.e. event flow editing facilities are essential: moving, deleting and inserting events or sections of events, should be made simple and effective.

Chapter 10

Reflection

This final short chapter attempts to reflect on the work presented in this thesis. We first summarize the contents of the preceding chapters and restate the contributions made by this work. We then discuss the practicality of authoring interactive stories in the way advocated throughout this thesis. Finally, some ideas for future work are presented.

10.1 Summary of Thesis

The difficulty of authoring IS story-worlds that provide a true sense of agency and are also dramatically engaging (referred to here as fully realized IS story-worlds) was the central motivating problem of this work. Wardrip-Fruin (2008) has coined the term “Tale-Spin Effect” to describe an interactive AI system that despite great internal complexity produces outputs that are relatively underwhelming for the end-user. Wardrip-Fruin contrasts this to the “Eliza-Effect” observed during user interaction with the early chat bot system Eliza (Weizenbaum, 1966), which despite its simple, almost trivial internal implementation was regarded as possessing almost human-level intelligence by many end users.

We experienced the Tale-Spin effect first hand during the creation of the educational interactive drama FearNot!, the creation process of which was related in Chapter 2 as a detailed IS authoring case study. Despite its complex agent architecture (FAtiMA) and spending of considerable authoring effort and resources, the end-result “FearNot!” was a far cry from the goal of producing a fully-realized IS artefact. Frustration with this state of affairs was the main motivator for focussing on the authoring bottleneck as a topic for this PhD. As is shown in Chapter 3, the problem is universal and applies to both explicitly specified story graphs and a wide spectrum of generative AI based solutions. While the latter reduce the overall amount of content that has to be authored through recombination of content and more efficient representation, the former explicit representation provides much more authorial intent. In practise,

achieving fully-realized IS with either approach is always primarily a scalability problem. Chapter 4 discussed existing IS authoring tools and found that scalability has not been a design criterion for them. Existing data-driven authoring efforts on the other hand seem to offer a solution to the scalability aspect, especially if input data is produced by an online crowd.

In Chapter 5 we discussed different ways in which authoring data could be collected online and identified a crowdsourced collection of example stories as the most promising approach. We then suggested the concept of Crowd Task Adaptation as an improvement of the typical crowdsourcing process, that allows the elicitation of more relevant contributions by adapting the task for a crowd worker in realtime, based on the current collection state. Chapter 6 describes how this idea was applied to the design of an authoring tool called ENIGMA, which aims to derive story-world representations compatible with the FATiMA architecture from collected example stories. However during implementation and early testing it became apparent that the envisaged design and necessary usability improvements were outwith the scope of what could be achieved within this PhD. The focus of the authoring tool therefore changed to a simpler story representation based on explicit branching. Chapter 7 described this second incarnation of the tool called CROSCAT. CROSCAT can adapt authoring crowd tasks by modifying the back story contributors are asked to continue. Chapter 8 explained how, armed with the CROSCAT tool, a collaborative authoring experiment could be designed that would be able to shed some light on the validity of our Crowd Task Adaptation hypothesis. A preparatory experiment for piloting a story scenario using hand-written stories was also described. Finally Chapter 9 presented the results of these experiments and most importantly found that the use of Crowd Task Adaptation significantly improved how end users rated the authored IS artefact.

By completing this work the following contributions were made (listed in order of estimated significance).

1. The novel content creation method of Crowd Task Adaptation was introduced and shown experimentally to be beneficial for authoring IS story worlds in at least one concrete realization of the idea.
2. Two exemplary designs for how to apply this method in an authoring tool are provided, one of which is fully and one of which is partially implemented.
3. Our experiments of running a crowdsourced collection of dramatically improvised stories are novel in many aspects compared to similar studies found in the literature and thus provide useful authoring case studies. Their novel aspects are listed in detail in Section 4.3.3.

4. As a side product of our experiments we are able to provide some recommendations for the design of future IS authoring tools.
5. This thesis includes a case-study of the major IS system FearNot! that includes novel results of an analysis of the authoring process and produced artefact.
6. The authoring tools and stories produced for this work are available to interested parties and may be used in future research.

10.2 Practicality of Crowdsourced Authoring

Throughout this thesis we have worked under the assumption that creative human labour is readily available through the internet. Of course the reality is somewhat more complicated. While the technical possibilities for reaching out to millions of possible collaborators exist, one has to compete with an almost infinite amount of other online activities for their attention. Realizing a crowdsourced authoring effort in practise therefore also poses the additional challenge of recruitment. The way in which subjects for the authoring experiments presented in this thesis were recruited is not necessarily a good sustainable model for carrying out such activities in earnest, when the primary goal is to produce a real IS artefact and not just as in our case to study the authoring process itself. For our studies, we recruited subjects through word of mouth and social media. Thus our contributors were primarily friends, family, co-workers and acquaintances. Their primary motivation for participation was goodwill and altruism, helping a friend with their research project. A weaker secondary motivation was probably curiosity, but this again was probably mostly based on knowing this thesis' author personally and wanting to find out what that PhD work was all about. A strength of social media is that they allow sharing content and several people reposted our call for participation. Through this, we managed to recruit a number of second-degree acquaintances, who were probably again motivated by altruism, doing a favour to a first-degree acquaintance by helping a second-degree acquaintance. Goodwill or altruism alone however are not good enough incentives to achieve a viral effect. That is not to say that crowdsourcing tasks with mainly altruistic incentives cannot be spread virally through social media, on the contrary. But the altruistic motivation for helping in our case a PhD student with their research or more generally an IS author with creating an IS artefact is arguably weak, compared to for example helping to locate a plane crash site by reviewing satellite data. A direct or at least indirect connection to the person orchestrating the effort is needed for people to care enough, which limits the scalability of this approach.

If in practise altruism is not a good enough incentive, which other ones could then be employed to recruit a pool of authors? Curiosity is certainly an option, as was shown

by e.g. “the Restaurant” project (Orkin and Roy, 2007), whose subjects were largely recruited from the readers of media news stories that covered and “advertised” the experiment as an innovative new type of video game. The problem with this approach is of course that it is not sustainable in the long term. Curiosity of both the media and its readers wears off quickly and once the novelty of this new approach to story creation is gone, a better incentive will be needed once again.

There are of course crowdsourcing platforms such as Amazon’s Mechanical Turk or Crowdfunder, whose workers main incentive is of a financial nature. If a budget is available paying participants may be a viable option. Participants for the Scheherazade system Li et al. (2012) are for example recruited this way. However, if creativity in the collected stories is important, the average quality of contributions gathered this way is questionable as most workers on crowd platforms just want to get the job done and move on to the next one quickly. This is less of an issue for the Scheherazade system, which aims to construct primarily situational scripts that require less creative input than the type of authoring we have introduced in this thesis. Another possible problem with using paid workers from crowdsourcing platforms may be a lack of common cultural understanding among the workers, which is arguably a necessary prerequisite for jointly creating a story. As was mentioned in the previous chapter, we also utilised the platform Crowdfunder for our initial collection of story ratings, but not for authoring. Nevertheless some of the comments left by these workers after rating the Seagnomes artefact reveal a very different cultural understanding of the provided story, which suggests that such issues would also arise, if such a platform was used for the recruitment of authors. The following two comments exemplify this:

“I don’t like the ending by the way mostly because if i to choose a leader i won’t pick a man who wears trouser with pink heart on it.”

“In my opinion, the brief story does not appeal to kids (equal gender rights, spirituality, this are mind impressionable motives and should be avoided).”

We see the most promising sustainable way for recruiting subjects in connecting authoring efforts as described in this thesis with online communities centered around a common (narrative) interest. Fan communities for popular fictional universes are a perfect example of this. At the time of writing the website <http://www.harrypotterfanfiction.com/> has for example claimed to have more than 90.000 members, more than 37.000 of which have been actively authoring a total of more than 81.000 Harry Potter fan fiction stories. The members of such communities clearly possess the required motivation and their love of the fictional universe is a better incentive than any of the ones mentioned above. Of course involving fan fiction communities means restricting oneself to a particular pre-existing fictional universe

and there may be copyright issues to consider so if a similar community could be built from the ground up on the creator’s own terms this would be even better.

10.3 Future Work

Like any piece of research, this work opens up ample opportunities for follow up investigations. Below future work is listed grouped into potential follow-up work to specific software and experiments discussed in this thesis and more general future investigations into the Crowd Task Adaptation Concept.

10.3.1 Specific Follow-ups

As discussed in the previous chapter, we derived a list of possible improvements for the CROSCAT system from user feedback. An obvious next step in the continuation of this work would be an implementation of these items, followed by a repetition of the “Seagnomes” experiment using the improved CROSCAT software. By keeping the same story setting, the results of this repeated experiment could be directly compared with the previous ones. Particularly interesting would be whether the addition of more user friendly story editing facilities would lead to less “meandering” stories as we hypothesized.

In general, however, none of these additions would make a fundamental difference to Crowd Task Adaptation, which we have already shown to work in the CROSCAT setting. One specific idea for a more substantial change to the CROSCAT system, is to present choices to end-users in more intriguing ways. For example, when using a foreshadowing technique similar to that described by Bae and Young (2008), choices might appear less arbitrary and Crowd Task Adaptation, which produces more and higher quality choices may be valued even more.

With regards to CROSCAT, we have already mentioned (in Section 7.5) the scalability issues our naive implementation would probably encounter with a contributor pool some orders of magnitude larger than during our experimental studies. Further work would be required to solve these issues. For example one could imagine several ways for improving the scoring algorithm, either by reducing its runtime, making it iterative (so that rescoring does not need to process the entire graph) or by changing the heuristics in some way. Realistically though, if someone should be willing to expend the effort and resources required for such an endeavour, they would probably want to make many other changes to the system too. In this hypothetical situation where the CROSCAT system needs to scale up way beyond its current possibilities we would recommend a complete re-design and a fresh implementation as such a system would have very different requirements since CROSCAT was designed as a proof of

concept research prototype.

A large scale future work project would be to have another attempt at realizing the design plans we had for ENIGMA with more resources. This is of course unlikely to happen in reality, but if it were to happen then we would advocate one important change based on what was learned in this thesis: as was discussed in the last chapter, based on the two experiments we conducted it seems difficult to collect a corpus of entirely consistent stories even when imposing strong narrative constraints. In either experiment, character behaviour in different stories did not always match up in ways that it could be attributed to the same personality. Therefore we would suggest to drop the consistency requirement in future attempts of crowdsourced IS authoring. In order to build sensible story world representations from inconsistent input stories a future version of ENIGMA (or indeed any system that makes similar attempts of learning character-centric story representations from example stories) should not target the limited story representation of the “vanilla” FAtiMA system. This is because the FAtiMA story representation has no way of expressing contradicting behaviours. For example if in one story a wife forgives her husband after she finds him cheating, whereas in another one she attempts to murder him, we cannot reconcile both behaviours and easily express both behavioural variants in the same character. What is needed is something like the late commitment strategy of the Virtual Storyteller system (Swartjes et al., 2008). Unknown character traits should not be fixed at the start of the simulation but instead be chosen at runtime. With such a scheme we could allow characters to be different across collected stories.

10.3.2 General Follow-ups

There are also many possible lines of investigation following on from the work discussed in this thesis that need not be directly based on the specific tools and experiments we produced. Implementing the Crowd Task Adaptation paradigm in a different IS authoring context (e.g. for producing a different story representation or using a different user interface for creating stories) would help in further understanding how to effectively employ it. And with such future work hopefully resulting in a bigger and more convincing body of evidence for our hypothesis that Crowd Task Adaptation is a useful “design pattern”, it would help to pave the way towards its adoption in producing real (i.e. non-trivial) IS artefacts.

A general open question about the crowdsourced authoring style following from the work in this thesis was how to define a story context so as to elicit contributions that are both varied and consistent. Investigating this question would be valuable for crowdsourced authoring in general, even if Crowd Task Adaptation is not used. On the other hand, however, it might be interesting to investigate how Crowd Task Adaptation may be employed to counteract the consistency problems we have found, e.g. by

continually modifying or adding to initially incomplete character profile sheets given to authors, based on character profile information gathered from already collected example stories. This is similar to CROSCAT's backstory modification approach, but technically more challenging, as a way to reliably extract this information from collected stories is needed.

Another question which we did not need to address for the small scale studies performed for this PhD but which may become relevant for more large scale authoring projects is how to perform quality control and filter out inappropriate or low quality contributions which may be caused by vandalism, technical problems or unqualified contributors. The unique challenge for Crowd Task Adaptation is that such filtering would ideally have to occur in realtime. Filtering the corpus only after all data has been collected may be too late, as by then invalid contributions will probably have influenced the tasks handed out to subsequent contributors in unwanted ways. Related to the question of how to perform quality control is that of recruitment, as the quality of contributions will to a large degree depend on who the contributors are. We have discussed above some possibilities and challenges for recruiting collaborators through e.g. social networks, crowdsourcing platforms or fan communities. A useful strand of future work would be to investigate what effects employing these different pools of workers have on the results of crowdsourced authoring tasks and to develop guidelines for effective recruitment.

10.4 Concluding Remarks

We hope to have made a convincing case for strongly considering the use of crowdsourcing in future IS endeavours in the first half of the thesis, while the second half demonstrates that if crowdsourcing is indeed being used, there are many ways in which the process can be further enhanced through Crowd Task Adaptation. The long term impact we would hope this work to have is for the creators of future IS systems to be aware of the Crowd Task Adaptation paradigm. We believe the concept to be strong, convincing and versatile enough to fit into many different contexts, some of which we have demonstrated here. Thus, awareness may really be all that stands in the way of its adoption, which has the potential of being a further step along the road towards fully-realized Interactive Stories.

Glossary

CBR	Case-Based Reasoning
eCIRCUS	Education through Characters with emotional Intelligence and Role-playing Capabilities that Understand Social Interaction
FAtiMA	FearNot AffecTive Mind Architecture
HSP	Heuristic Search Planning
HTN	Hierarchical Task Network
IS	Interactive Storytelling
OCC	Ortony, Clore and Collin's Taxonomy of Emotions
STRIPS	Stanford Research Institute Problem Solver
VICTEC	Virtual ICT with Empathic Characters

References

- Alves, T., Simoes, A., Figueiredo, R., Vala, M., Paiva, A., and Aylett, R. (2008). So tell me what happened: turning agent-based interactive drama into comics. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 3*, pages 1269–1272.
- Arinbjarnar, M. and Kudenko, D. (2008). Schemas in directed emergent drama. In *First Joint International Conference on Interactive Digital Storytelling, ICIDS 2008, Erfurt, Germany*, page 180–185. Springer.
- Arinbjarnar, M. and Kudenko, D. (2009). Duality of actor and character goals in virtual drama. In *Intelligent Virtual Agents*, page 386–392. Springer.
- Aristotle (330 BC). *The Poetics*. Mineola, New York: Dover, 1997.
- Aylett, R. (1999). Narrative in virtual environments: Towards emergent narrative. In Mateas, M. and Sengers, P., editors, *AAAI Fall Symposium, Technical report FS-99-01*, pages 83–86. AAAI Press.
- Aylett, R. (2000). Emergent narrative, social immersion and storyfication. In *Proceedings, Narrative Interaction for Learning Environments*, Edinburgh, UK.
- Aylett, R., Dias, J., and Paiva, A. (2006a). An affectively driven planner for synthetic characters. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 2–10. AAAI press.
- Aylett, R., Louchart, S., Dias, J., Paiva, A., Vala, M., Woods, S., and Hall, L. (2006b). Unscripted narrative for affectively driven characters. *IEEE Computer Graphics and Applications*, 26(3):42–52.
- Aylett, R., Vala, M., Sequeira, P., and Paiva, A. (2007). FearNot! - an emergent narrative approach to virtual dramas for anti-bullying education. In *Virtual Storytelling, 4th International Conference, ICVS 2007*, pages 202–205, Saint-Malo, France.
- Aylett, R. S., Louchart, S., Dias, J., Paiva, A., and Vala, M. (2005). FearNot! - an experiment in emergent narrative. In *Intelligent Virtual Agents*, pages 305–316.

REFERENCES

- Aylett, R. S., Louchart, S., Tychsen, A., Hitchens, M., Figueiredo, R., and Delgado Mata, C. (2008). Managing emergent character-based narrative. In *Proceedings of the 2nd international conference on INtelligent TEchnologies for interactive enterTAINment, INTETAIN*.
- Bae, B.-C. and Young, R. M. (2008). A use of flashback and foreshadowing for surprise arousal in narrative using a plan-based approach. In *First Joint International Conference on Interactive Digital Storytelling, ICIDS 2008, Erfurt, Germany*, page 156–167. Springer.
- Baldwin, C. Y. and Clark, K. B. (2006). The architecture of participation: Does code architecture mitigate free riding in the open source development model? *Manage. Sci.*, 52(7):1116–1127.
- Balet, O. (2007). Inscape: An authoring platform for interactive storytelling. In *Virtual Storytelling, 4th International Conference, ICVS 2007*, page 176–177, Saint-Malo, France. Springer.
- Barber, H. and Kudenko, D. (2007a). Dynamic generation of dilemma-based interactive narratives. In *Proceedings Of The Third Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE '07)*, pages 2–7, Stanford, CA.
- Barber, H. and Kudenko, D. (2007b). A user model for the generation of dilemma-based interactive narratives. In *AIIDE'07 Workshop on Optimising Player Satisfaction*, pages 13–18.
- Benkler, Y. and Nissenbaum, H. (2006). Commons-based peer production and virtue. *Journal of Political Philosophy*, 14(4):394–419.
- Boal, A. (2000). *Theatre of the Oppressed*. Pluto, London.
- Bond, S. (2007). Storytron review. Retrieved 20th February 2014. Available from: <http://plover.net/~bonds/storytron.html>.
- Bonet, B. and Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, 129(1–2):5–33.
- Bremond, C. (1974). *Logique du récit*. Seuil.
- Brom, C., Bída, M., Gemrot, J., Kadlec, R., and Plch, T. (2009). Emohawk: Searching for a “Good” emergent narrative. In *Second Joint International Conference on Interactive Digital Storytelling, ICIDS 2009, Guimaraes, Portugal*, pages 86–91.
- Campbell, J. (1972). *The Hero With A Thousand Faces*. Princeton University Press.

REFERENCES

- Cardona-Rivera, R. E., Robertson, J., Ware, S. G., Harrison, B., Roberts, D. L., and Young, R. M. (2014). Foreseeing meaningful choices. In *Proceedings of the Tenth Annual AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2014)*.
- Cavazza, M., Charles, F., and Mead, S. (2002). Character-based interactive storytelling. *IEEE Intelligent Systems*, 17(4):17–24.
- Cavazza, M., Charles, F., and Mead, S. J. (2003). Generation of humorous situations in cartoons through plan-based formalisations. In *CHI-2003 Workshop: Humor Modeling in the Interface*.
- Cavazza, M., Charles, F., and Mead, S. J. (2004). Developing re-usable interactive storytelling technologies. In *IFIP World Computer Congress, Toulouse, France*, pages 39–44.
- Chernov, S., Iofciu, T., Nejdl, W., and Zhou, X. (2006). Extracting semantics relationships between wikipedia categories. *Proceedings of 1st International Workshop: "SemWiki2006 - From Wiki to Semantics" (SemWiki 2006), co-located with the ESWC2006 in Budva*.
- Costikyan, G. (2007). Games, storytelling, and breaking the string. In *Second Person: role-playing and story in games and playable media*, pages 5–14. MIT Press.
- Crawford, C. (1999). Assumptions underlying the Erasmatron interactive storytelling engine. In *Proc. AAAI Fall Symposium on Narrative Intelligence*, North Falmouth MA. AAAI Press.
- Crawford, C. (2005). *Chris Crawford on Interactive Storytelling*. New Riders.
- Crawford, C. (2007). Storytron - interactive storytelling. Retrieved 12th July 2007. Available from: <http://www.storytron.com>.
- Crawford, C. (2008). Deikto: An application of the weak Sapir-Whorf hypothesis. In *Proceedings of the hypertext 2008 workshop on Creating out of the machine: hypertext, hypermedia, and web artists explore the craft*, Creating '08, page 1–4, New York, NY, USA. ACM.
- Crawford, C. (2012a). Storytron: Plans for the future. Retrieved 17th February 2014. Available from: <http://www.storytron.com/PlansForFuture.html>.
- Crawford, C. (2012b). Storytron: What went wrong with the previous effort. Retrieved 26th January 2014. Available from: <http://www.storytron.com/WhatWentWrong.html>.

REFERENCES

- Cypher, A., editor (1993). *Watch What I Do: Programming by Demonstration*. MIT Press.
- Dang, K. D., Hoffmann, S., Champagnat, R., and Spierling, U. (2011). How authors benefit from linear logic in the authoring process of interactive storyworlds. In *Fourth International Conference on Interactive Digital Storytelling, ICIDS 2011*, pages 249–260, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Dias, J. and Paiva, A. (2005). Feeling and reasoning: A computational model for emotional characters. In *EPIA 2005*, pages 127–140.
- Dias, J. and Paiva, A. (2011). Agents with emotional intelligence for storytelling. In *Proceedings of the 4th International Conference on Affective Computing and Intelligent Interaction, ACII 2011*, Lecture Notes in Computer Science, pages 77–86, Memphis, TN, USA. Springer Berlin Heidelberg.
- Doan, A., Ramakrishnan, R., and Halevy, A. (2010). Mass collaboration systems on the world wide web. *Communications of the ACM*, pages 86–96.
- Doerner, D. (2005). The mathematics of emotions. In *Fifth International Conference on Cognitive Modelling*, pages 75–80.
- Donikian, S. and Portugal, J. N. (2004). Writing interactive fiction scenarii with DraMachina. In *Technologies for Interactive Digital Storytelling and Entertainment (TIDSE)*, pages 101–112. Springer.
- Engel, R. (2006). SPIN: a semantic parser for spoken dialog systems. In *Proceedings of the 5th Slovenian and First International Language Technology Conference (IS-LTC 2006)*.
- Evans, R. and Short, E. (2014). Versu - a simulationist storytelling system. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(2):113–130.
- Fairclough, C. (2004). *Story Games and the OPIATE System*. PhD thesis, University of Dublin - Trinity College.
- Fendt, M. W., Harrison, B., Ware, S. G., Cardona-Rivera, R. E., and Roberts, D. L. (2012). Achieving the illusion of agency. In *Proceedings of the 5th International Conference on Interactive Storytelling, ICIDS’12*, page 114–125, Berlin, Heidelberg. Springer-Verlag.
- Field, A. and Hole, G. (2003). *How to design and report experiments*. Sage publications Ltd.

REFERENCES

- Figueiredo, R., Brisson, A., Aylett, R., and Paiva, A. (2008). Emergent stories facilitated. In *First Joint International Conference on Interactive Digital Storytelling, ICIDS 2008, Erfurt, Germany*, pages 218–229. Springer.
- Figueiredo, R. and Paiva, A. (2010). "I want to slay that dragon!": influencing choice in interactive storytelling. In *Proceedings of the Third joint conference on Interactive digital storytelling, ICIDS'10*, page 26–37, Berlin, Heidelberg. Springer-Verlag.
- Fikes, R. and Nilsson, N. J. (1971). STRIPS: a new approach to the application of theorem proving to problem solving. *Artif. Intell.*, 2(3/4):189–208.
- Freytag, G. (1863). *Technique of the Drama*. Benjamin Blom.
- Gerke, J. (2010). *Plot versus character a balanced approach to writing great fiction*. Writer's Digest Books, Cincinnati, Ohio.
- Gervás, P., Díaz-Agudo, B., Peinado, F., and Hervás, R. (2005). Story plot generation based on CBR. *Know.-Based Syst.*, 18(4-5):235–242.
- Grasbon, D. and Braun, N. (2001). A morphological approach to interactive storytelling. In *Proc. CAST01, Living in Mixed Realities. Special issue of Netzspannung.org/journal, the Magazine for Media Production and Inter-media Research*, page 337–340.
- Guibas, L. J. and Sedgewick, R. (1978). A dichromatic framework for balanced trees. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, page 8–21, Washington, DC, USA. IEEE Computer Society.
- Góczya, Z. (2010). Myth #23: Choices should always be limited to 7+/-2 - UX myths. Retrieved 5th July 2013. Available from: <http://uxmyths.com/post/931925744/myth-23-choices-should-always-be-limited-to-seven>.
- Hoffmann, S., Spierling, U., and Struck, G. (2011). A practical approach to introduce story designers to planning. In *Proceedings of GET 2011, IADIS International Conference Game and Entertainment Technologies*,, pages 22–24, Rome, Italy.
- Hofstede, G., Hofstede, G. J., and Minkov, M. (1997). *Cultures and organizations*. McGraw-Hill New York.
- Howe, J. (2006). The rise of crowdsourcing. *Wired Magazine*, 06(14):1–4.
- Iurgel, I. A. (2006). Cyranus-an authoring tool for interactive edutainment applications. In *Proceedings of Edutainment 2006, Int. Conf on E-Learning and Games, Zhejiang*, pages 577–580.

REFERENCES

- Kelleher, C. (2009). Supporting storytelling in a programming environment for middle school children. In *Second Joint International Conference on Interactive Digital Storytelling, ICIDS 2009, Guimaraes, Portugal*, pages 1–4.
- Koenitz, H. (2011a). Extensible tools for practical experiments in IDN – the Advanced Stories Authoring and Presentation System. In *Fourth International Conference on Interactive Digital Storytelling, ICIDS 2011*, pages 79–84, Vancouver, Canada. Springer Berlin Heidelberg.
- Koenitz, H. (2011b). An iterative approach towards interactive digital narrative – early results with the Advanced Stories Authoring and Presentation System. In *Proceedings Of The 10th International Conference on Web-based Learning (ICWL 2011)*, pages 59–68, Hong Kong.
- Koenitz, H. and Chen, K.-J. (2012). Genres, structures and strategies in interactive digital narratives: analyzing a body of works created in ASAPS. In *Proceedings of the 5th international conference on Interactive Storytelling, ICIDS’12*, page 84–95, Berlin, Heidelberg. Springer-Verlag.
- Kolodner, J. L. (1992). An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1):3–34.
- Kriegel, M. and Aylett, R. (2007). A mixed initiative authoring environment for emergent narrative planning domains. In *Proceedings of the AISB Annual Convention*, pages 453–456.
- Kriegel, M. and Aylett, R. (2008). Emergent narrative as a novel framework for massively collaborative authoring. In *Intelligent Virtual Agents, 8th International Conference, IVA 2008, Tokyo, Japan*, volume 5208, pages 73–80.
- Kriegel, M. and Aylett, R. (2010). Crowd-sourced AI authoring with ENIGMA. In *Proceedings of the Third joint conference on Interactive digital storytelling, ICIDS’10*, page 275–278, Berlin, Heidelberg. Springer-Verlag.
- Kriegel, M., Aylett, R., Dias, J., and Paiva, A. (2007). An authoring tool for an emergent narrative storytelling system. In *Papers from the AAAI Fall Symposium on Intelligent Narrative Technologies, Technical Report FS-07-05*, pages 55–62.
- Kriegel, M., Lim, M. Y., Nazir, A., Aylett, R., Cawsey, A., Enz, S., Rizzo, P., and Hall, L. (2008). ORIENT: an inter-cultural role-play game. In *5th International Conference on Narrative and Interactive Learning Environments Edinburgh, Scotland*.

REFERENCES

- Larkey, L. B. and Love, B. C. (2003). CAB: connectionist analogy builder. *Cognitive Science*, 27(5):781–794.
- Laurel, B. (1993). *Computers as theatre*. Addison-Wesley Pub. Co., Reading, Mass.
- Law, E. L. M., von Ahn, L., Dannenberg, R. B., and Crawford, M. (2007). Tagatune: A game for music and sound annotation. In *Proc. Intl. Symp. Music Information Retrieval*, page 361–364.
- Lazarus, R. S. and Folkman, S. (1984). *Stress, appraisal and coping*. New York: Springer.
- Lebowitz, M. (1985). Story-telling as planning and learning. *Poetics*, 14(6):483–502.
- Lenat, D. B. (1995). CYC: a large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):33–38.
- Li, B. (2012). Narrative intelligence without (domain) boundaries. In *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*.
- Li, B., Lee-Urban, S., and Riedl, M. O. (2012). Toward autonomous crowd-powered creation of interactive narratives. In *5th Workshop on Intelligent Narrative Technologies*, volume 8, pages 25–52, Palo Alto, CA.
- Li, B., Thakkar, M., Wang, Y., and Riedl, M. O. (2014a). Data-driven alibi story telling for social believability. In *Proceedings of the 2014 Foundations of Digital Games Workshop on Social Behavior in Games*, Ft. Lauderdale, Florida, USA.
- Li, B., Thakkar, M., Wang, Y., and Riedl, M. O. (2014b). Storytelling with adjustable narrator styles and sentiments. In *Proceedings of the 2014 International Conference on Interactive Digital Storytelling*, pages 1–12, Singapore.
- Lieberman, H., Smith, D., and Teeters, A. (2007). Common Consensus: a web-based game for collecting commonsense goals. In *Workshop on Common Sense for Intelligent Interfaces, ACM International Conference on Intelligent User Interfaces (IUI-07)*.
- Lim, M. Y., Dias, J., Aylett, R., and Paiva, A. (2008). Improving adaptiveness in autonomous characters. In *Intelligent Virtual Agents*, page 348–355. Springer.
- Lin, G. and Walker, M. (2011). All the world’s a stage: Learning character models from film. In *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*.

REFERENCES

- Louchart, S., Aylett, R., and Dias, J. (2007a). Double appraisal for synthetic characters. In *Proceedings of the 7th International Conference on Intelligent Virtual Agents*, IVA '07, page 393–394, Berlin, Heidelberg. Springer-Verlag.
- Louchart, S., Aylett, R., Kriegel, M., Dias, J., Figueiredo, R., and Paiva, A. (2007b). Authoring emergent narrative-based games. *Journal of Game Development*, 3(1):19–37.
- Louchart, S., Swartjes, I., Kriegel, M., and Aylett, R. (2008). Purposeful authoring for emergent narrative. In *First Joint International Conference on Interactive Digital Storytelling, ICIDS 2008, Erfurt, Germany*, pages 273–284. Springer.
- Macvean, A., Hajarnis, S., Headrick, B., Ferguson, A., Barve, C., Karnik, D., and Riedl, M. O. (2011). WeQuest: scalable alternate reality games through end-user content authoring. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, page 22. ACM.
- Magerko, B. (2002). A proposal for an interactive drama architecture. In *AAAI 2002 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment*.
- Marsella, S. C., Pynadath, D. V., and Read, S. J. (2004). PsychSim: agent-based modeling of social interactions and influence. In *Proceedings of the international conference on cognitive modeling*, volume 36, page 243–248.
- Mascarenhas, S., Dias, J., Afonso, N., Enz, S., and Paiva, A. (2009). Using rituals to express cultural differences in synthetic characters. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, page 305–312.
- Mason, B. and Thomas, S. (2008). A million penguins research report. Technical report, Institute of Creative Technologies, De Montfort University, Leicester, UK.
- Mateas, M. and Stern, A. (2003). Façade: An experiment in building a fully-realized interactive drama. In *Proceedings of Game Developer's Conference: Game Design Track*, San Jose, California, USA.
- Mateas, M. and Stern, A. (2005). Structuring content in the facade interactive drama architecture. In *Artificial Intelligence and Interactive Digital Entertainment (AI-IDE)*, pages 93–98. AAAI press.
- McCoy, J., Treanor, M., Samuel, B., Reed, A., Mateas, M., and Wardrip-Fruin, N. (2014). Social story worlds with comme il faut. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(2):97–112.

REFERENCES

- McCoy, J., Treanor, M., Samuel, B., Tearse, B., Mateas, M., and Wardrip-Fruin, N. (2010a). Authoring game-based interactive narrative using social games and comme il faut. In *Proceedings of the 4th International Conference & Festival of the Electronic Literature Organization: Archive & Innovate (ELO 2010)*.
- McCoy, J., Treanor, M., Samuel, B., Tearse, B., Mateas, M., and Wardrip-Fruin, N. (2010b). Comme il faut 2: A fully realized model for socially-oriented gameplay. In *INT3 2010: Proceedings of the Intelligent Narrative Technologies III Workshop*, New York, NY, USA.
- Medler, B. and Magerko, B. (2006). Scribe: A tool for authoring event driven interactive drama. In *3rd International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE 2006)*, pages 139–150, Darmstadt (Germany). Springer.
- Mehta, M., Corradini, A., Ontañón, S., and Henrichsen, P. J. (2010). Textual vs. graphical interaction in an interactive fiction game. In *Proceedings of the Third Joint Conference on Interactive Digital Storytelling, ICIDS'10*, page 228–231, Berlin, Heidelberg. Springer-Verlag.
- Miller, G. A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63(2):81–97.
- Miller, G. A. (1995). WordNet: a lexical database for english. *Commun. ACM*, 38(11):39–41.
- Mitchell, A. and McGee, K. (2012). Reading again for the first time: a model of rereading in interactive stories. In *Proceedings of the 5th international conference on Interactive Storytelling, ICIDS'12*, page 202–213, Berlin, Heidelberg. Springer-Verlag.
- Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill, New York, NY [u.a.
- Murray, J. H. (1998). *Hamlet on the Holodeck*. The MIT Press.
- Nazir, A., Ritter, C., Aylett, R., Krumhuber, E., Swiderska, A., Degens, N., Endrass, B., Hume, C., Hodgson, J., and Mascarenhas, S. (2012). eCute: difference is good. In *Proc. of the IADIS International Conference e-Learning*, page 425–429.
- Orkin, J. (2011). Using online games to capture, generate, and understand natural language. In *Proceedings of the 13th European Workshop on Natural Language Generation, ENLG '11*, page 71–71, Stroudsburg, PA, USA. Association for Computational Linguistics.

REFERENCES

- Orkin, J. and Roy, D. (2007). The restaurant game: Learning social behavior and language from thousands of players online. *Journal Of Game Development*, 3(1):39–60.
- Orkin, J. and Roy, D. (2009). Automatic learning and generation of social behavior from collective human gameplay. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 385–392.
- Orkin, J., Smith, T., Reckman, H., and Roy, D. (2010). Semi-automatic task recognition for interactive narratives with EAT & RUN. In *Proceedings of the 3rd Intelligent Narrative Technologies Workshop (INT3)*.
- Ortony, A., Clore, G. L., and Collins, A. (1988). *The cognitive structure of emotions*. Cambridge University Press.
- Pizzi, D. and Cavazza, M. (2008). From debugging to authoring: Adapting productivity tools to narrative content description. In *First Joint International Conference on Interactive Digital Storytelling, ICIDS 2008, Erfurt, Germany*, pages 285–296. Springer.
- Pizzi, D., Charles, F., Lugin, J.-L., and Cavazza, M. (2007). Interactive storytelling with literary feelings. In *Proceedings of the Second International Conference on Affective Computing and Intelligent Interaction (ACII '07)*, page 630–641. Springer.
- Polti, G. (1921). *The thirty-six dramatic situations*. J.K. Reeve, Franklin, O.
- Propp, V. (1968). *Morphology of the Folktale*. University Of Texas Press.
- Pynadath, D. V. and Marsella, S. C. (2005). PsychSim: modeling theory of mind with decision-theoretic agents. In *IJCAI*, volume 5, page 1181–1186.
- Rank, S. and Petta, P. (2012). Backstory authoring for affective agents. In *Proceedings of the 5th international conference on Interactive Storytelling, ICIDS'12*, page 144–149, Berlin, Heidelberg. Springer-Verlag.
- Rao, A. S., Georgeff, M. P., et al. (1995). BDI agents: From theory to practice. In *Proceedings of the first international conference on multi-agent systems (ICMAS-95)*, pages 312–319.
- Riedl, M. (2004). *Narrative Generation: Balancing Plot and Character*. PhD thesis, Department of Computer Science, North Carolina State University.
- Riedl, M., Saretto, C. J., and Young, R. M. (2003). Managing interaction between users and agents in a multi-agent storytelling environment. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '03*, page 741–748, New York, NY, USA. ACM.

- Riedl, M. O. (2009). Incorporating authorial intent into generative narrative systems. In *Intelligent Narrative Technologies II, Papers from the 2009 AAAI Spring Symposium*, pages 91—94. AAAI Press.
- Riedl, M. O. and Bulitko, V. (2013). Interactive narrative: An intelligent systems approach. *AI Magazine*, 34(1).
- Riedl, M. O. and Stern, A. (2006). Failing believably: Toward drama management with autonomous actors in interactive narratives. In *3rd International Conference on. Technologies for Interactive Digital Storytelling and Entertainment (TIDSE 2006)*, page 195–206, Darmstadt (Germany). Springer.
- Riedl, M. O. and Sugandh, N. (2008). Story planning with vignettes: Toward overcoming the content production bottleneck. In *First Joint International Conference on Interactive Digital Storytelling, ICIDS 2008, Erfurt, Germany*, pages 168–179. Springer.
- Riedl, M. O. and Young, R. M. (2010). Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39(1):217–268.
- Robertson, J. and Nicholson, K. (2007). Adventure author: a learning environment to support creative design. In *Proceedings of the 6th international conference on Interaction design and children*, page 37–44. ACM.
- Russell, S. and Norvig, P. (2003). *Artificial Intelligence, A modern Approach*. Prentice Hall, 2nd edition edition.
- Sapouna, M., Wolke, D., Vannini, N., Watson, S., Woods, S., Schneider, W., Enz, S., Hall, L., Paiva, A., André, E., Andre, E., Dautenhahn, K., and Aylett, R. (2010). Virtual learning intervention to reduce bullying victimization in primary school: a controlled trial. *Journal of child psychology and psychiatry, and allied disciplines*, 51(1):104–112.
- Sauer, S., Osswald, K., Wielemans, X., and Stifter, M. (2006). U-create: Creative authoring tools for edutainment applications. In *3rd International Conference on. Technologies for Interactive Digital Storytelling and Entertainment (TIDSE 2006)*, pages 163–168, Darmstadt (Germany). Springer.
- Schank, R. C. and Abelson, R. P. (1977). *Scripts, Plans, Goals, and Understanding*. Psychology Press.
- Seif El-Nasr, M. (2007). Interaction, narrative, and drama: Creating an adaptive interactive narrative using performance arts theories. *Interaction Studies*, 8(2):209–240.

REFERENCES

- Si, M., Marsella, S., and Pynadath, D. (2009). Directorial control in a decision-theoretic framework for interactive narrative. In *Second Joint International Conference on Interactive Digital Storytelling, ICIDS 2009, Guimaraes, Portugal*, pages 221–233.
- Si, M. and Marsella, S. C. (2010). Modeling rich characters in interactive narrative games. *GAMEON-ASIA, Shanghai, China*.
- Si, M., Marsella, S. C., and Pynadath, D. V. (2005). Thespian: Using MultiAgent fitting to craft interactive drama. In *Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 21–28. IEEE Computer Society.
- Si, M., Marsella, S. C., and Pynadath, D. V. (2006). Social norms models in thespian: Using decision theoretical framework for interactive drama. In *AISB 06 Proceedings, vol. 3*.
- Si, M., Marsella, S. C., and Riedl, M. O. (2008). Integrating story-centric and character-centric processes for authoring interactive drama. In *AIIDE*.
- Sina, S., Rosenfeld, A., and Kraus, S. (2013). Social narrative adaptation using crowdsourcing. In *2013 Workshop on Computational Models of Narrative*, pages 238–256, Hamburg, Germany.
- Singh, P. (2002). The open mind common sense project. Retrieved 22nd Oct 2008. Available from: <http://web.media.mit.edu/~push/Kurzweil.html>.
- Skorupski, J., Jayapalan, L., Marquez, S., and Mateas, M. (2007). Wide ruled: A friendly interface to author-goal based story generation. In *Virtual Storytelling, 4th International Conference, ICVS 2007, Saint-Malo, France*, pages 26–37.
- Skorupski, J. and Mateas, M. (2009). Interactive story generation for writers: Lessons learned from the wide ruled authoring tool. In *Proceedings of the 8th Digital Art and Culture Conference (DAC 2009), Irvine, CA*.
- Skorupski, J. and Mateas, M. (2010). Novice-friendly authoring of plan-based interactive storyboards. In *AIIDE*.
- Spierling, U. (2007). Adding aspects of “Implicit creation” to the authoring process in interactive storytelling. In *International Conference on Virtual Storytelling*, pages 13–25, Saint Malo, France.
- Spierling, U. and Iurgel, I. (2008). Workshop and panel: The authoring process in interactive storytelling. In *First Joint International Conference on Interactive Digital Storytelling, ICIDS 2008, Erfurt, Germany*, page 331. Springer.

REFERENCES

- Spierling, U., Iurgel, I., Richle, U., and Szilas, N. (2009). Workshop on authoring methods and conception in interactive storytelling. In *Second Joint International Conference on Interactive Digital Storytelling, ICIDS 2009, Guimaraes, Portugal*, pages 356–357.
- Spierling, U. and Szilas, N. (2009). Authoring issues beyond tools. In *Second Joint International Conference on Interactive Digital Storytelling, ICIDS 2009, Guimaraes, Portugal*, pages 50–61.
- Stern, A. (2008). Embracing the combinatorial explosion: A brief prescription for interactive story R&D. In *First Joint International Conference on Interactive Digital Storytelling, ICIDS 2008, Erfurt, Germany*, pages 1–5. Springer.
- Struck, H.-G. (2005). Telling stories knowing nothing: Tackling the lack of common sense knowledge in story generation systems. In *Proceedings of the International Conference on Virtual Storytelling 2005*, page 189–198. Springer.
- Swanson, R. and Gordon, A. S. (2008). Say anything: A massively collaborative open domain story writing companion. In *First Joint International Conference on Interactive Digital Storytelling, ICIDS 2008, Erfurt, Germany*, pages 32–40. Springer.
- Swanson, R. and Gordon, A. S. (2009). A comparison of retrieval models for open domain story generation. In *AAAI 2009 Spring Symposium on Intelligent Narrative Technologies II, Stanford, CA*.
- Swanson, R. and Gordon, A. S. (2010). A data-driven case-based reasoning approach to interactive storytelling. In *Third International Conference on Interactive Digital Storytelling, ICIDS 2010*, pages 186–197, Edinburgh, UK. Springer Berlin Heidelberg.
- Swartjes, I. (2007). Using narrative cases to author interactive story content. In *Proceedings of the Sixth International Conference on Entertainment Computing (ICEC 2007)*, pages 205–210.
- Swartjes, I. (2010). *Whose story is it anyway? How improv informs agency and authorship of emergent narrative*. PhD thesis, University of Twente.
- Swartjes, I., Kruizinga, E., and Theune, M. (2008). Let’s pretend I had a sword: Late commitment in emergent narrative. In *First Joint International Conference on Interactive Digital Storytelling, ICIDS 2008, Erfurt, Germany*, pages 264–267. Springer.

REFERENCES

- Swartjes, I. and Theune, M. (2009). Iterative authoring using story generation feedback: Debugging or co-creation? In *Second Joint International Conference on Interactive Digital Storytelling, ICIDS 2009, Guimaraes, Portugal*, pages 62–73.
- Swartjes, I. M. T. and Theune, M. (2008). The Virtual Storyteller: Story generation by simulation. In *Proceedings 20th Belgian-Netherlands Conference on Artificial Intelligence*, pages 257–264.
- Szilas, N. (2002). Structural models for interactive drama. In *Proceedings of the 2nd International Conference on Computational Semiotics for Games and New Media*.
- Szilas, N. (2003). IDtension: a narrative engine for interactive drama. In *1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE 2003)*, pages 187–203.
- Tapscott, A., Colas, J., Moghnieh, A., and Blat, J. (2013). Writing consistent stories based on structured multi-authored narrative spaces. In *2013 Workshop on Computational Models of Narrative*, pages 277–292, Hamburg, Germany.
- Theune, M., Faas, S., Heylen, D. K. J., and Nijholt, A. (2003). The Virtual Storyteller: Story creation by intelligent agents. In *1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE 2003)*, pages 204–215, Darmstadt (Germany).
- Thomas, J. and Young, R. M. (2006). Author in the loop: Using mixed-initiative planning to improve interactive narrative. In *ICAPS 2006 Workshop on AI Planning for Computer Games and Synthetic Characters*.
- Thompson, S. (1955). *Motif Index of Folk Literature*. Indiana University Press, Bloomington.
- Thue, D., Bulitko, V., Spetch, M., and Wasylishen, E. (2007). Interactive storytelling: A player modelling approach. In *Proceedings Of The Third Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE '07)*. Stanford, CA., pages 43–48.
- Todorov, T. (1970). Les transformations narratives. *Poétiques*, 3:322–333.
- Tomaszewski, Z. and Binsted, K. (2007). The limitations of a propp-based approach to interactive drama. In *Intelligent Narrative Technologies: Papers from the AAAI Fall Symposium*, volume 7, page 05.
- Turner, S. R. (1993). *Minstrel: A Computer Model of Creativity and Storytelling*. PhD thesis, University of California at Los Angeles.

- von Ahn, L. and Dabbish, L. (2004). Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM New York, NY, USA.
- von Ahn, L., Kedia, M., and Blum, M. (2006a). Verbosity: a game for collecting common-sense facts. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 75–78. ACM New York, NY, USA.
- von Ahn, L., Liu, R., and Blum, M. (2006b). Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 55–64. ACM New York, NY, USA.
- Wardrip-Fruin, N. (2008). The tale-spin effect: Toward an acknowledgement of process in digital literature. *Media-Space Journal*, 1(1).
- Watson, S. E., Vannini, N., Woods, S., Dautenhahn, K., Sapouna, M., Enz, S., Schneider, W., Wolke, D., Hall, L., Paiva, A., André, E., and Aylett, R. (2010). Intercultural differences in response to a computer-based anti-bullying intervention. *Educational Research*, 52(1):61–80.
- Weallans, A., Louchart, S., and Aylett, R. (2012). Distributed drama management: beyond double appraisal in emergent narrative. In *Proceedings of the 5th international conference on Interactive Storytelling, ICIDS’12*, page 132–143, Berlin, Heidelberg. Springer-Verlag.
- Wei, H. (2011). Structuring narrative interaction: What we can learn from Heavy Rain. In *Fourth International Conference on Interactive Digital Storytelling, ICIDS 2011*, pages 338–341, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Weiss, C., Oliveira, L. C., Paulo, S., Mendes, C., Figueira, L., Vala, M., Sequeira, P., Paiva, A., Vogt, T., and André, E. (2007). ECIRCUS: building voices for autonomous speaking agents. In *Proceedings of the 6th ISCA Workshop on Speech Synthesis, SSW-6*, pages 300–303, Bonn, Germany.
- Weiss, S., Müller, W., Spierling, U., and Steimle, F. (2005). Scenejo—an interactive storytelling platform. In *Proceedings of the International Conference on Virtual Storytelling 2005*, page 77–80. Springer.
- Weizenbaum, J. (1966). ELIZA - a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.
- Young, R. M. (2001). An overview of the mimesis architecture: Integrating intelligent narrative control into an existing gaming environment. In *The Working Notes of the*

REFERENCES

- AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, page 78–81.
- Young, R. M., Riedl, M. O., Branly, M., Jhala, A., Martin, R. J., and Saretto, C. J. (2004). An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Development*, 1(1):51–70.
- Zancanaro, M., Cappelletti, A., Signorini, C., and Strapparava, C. (2001). An authoring tool for intelligent educational games. In *Virtual Storytelling: Using Virtual Reality Technologies for Storytelling, International Conference, ICVS 2001*, page 61–68. Springer.
- Zucker-Scharff, A. (2011). Crowd-sourced television with Bar Karma and Storymaker | hack text. Retrieved 15th Aug 2013. Available from: <http://hacktext.com/2011/02/crowd-sourced-television-goes-to-the-next-level-with-bar-karma-and-storymaker-477/>.

Appendices

Appendix A

Complete List of FearNot! agent goals

The following table lists all 83 active pursuit FAtiMA goals that were authored for the emergent episodes in FearNot! The victim's goals for controlling the dialogue with the user in the interaction episodes are excluded from this list. Besides the goal names the table also lists for each goal the number of possible unique partially ordered plans (i.e. where the ordering of some actions is flexible, different orderings are not counted as different plans) and the average length (number of actions) of these plans. Only rational plans that the planner would actually construct are considered.

Goal Name	Unique POPs	Avg. Plan Length
ReplyPositively	1	1
ReplyNegatively	1	1
ReplyPositivelyNoChoice	1	1
AssistBully	1	1
AssistBullyJoinGroupNegativeAnswer	1	1
DefendVictimFrom	1	2
DefendFriendFrom	1	2
JoinGroup	3	5.67
ConvinceOtherNoJoinGroupOf2	1	3
ConvinceOtherJoinGroupOf2	1	3
JoinGroupAccept	1	1
JoinGroupRefuse	1	1
ConvinceGroupAccept	1	1
ConvinceGroupRefuse	1	1
ConvinceOtherNoJoinGroupOf2Accept	1	1
ConvinceOtherNoJoinGroupOf2Refuse	1	1
ConvinceOtherJoinGroupOf2Accept	1	1

ConvinceOtherJoinGroupOf2Refuse	1	1
JoinGroupThreatLeaveAccept	1	1
JoinGroupThreatLeaveRefuse	1	1
ConfrontVictimTellTeacher	1	1
IgnoreBully	1	1
StandUpToBully	1	1
MakeNewFriend	1	3
TellFriend	2	3
RunAway	1	1
WalkAway	1	1
LaughOff	1	1
FightBack	1	3
Insult	1	1
InsultBack	1	1
Fight	1	2
FightSuccess	1	1
GloatVictory	1	1
HelpInvite	1	2
DeceiveVictim	1	3
InviteToParty	1	2
AcceptInvitation	1	1
CancelPartyInvitation	1	1
RefuseParty	1	1
SabotagePartyInvitation	1	2
Bully	1	1
Tease	1	1
TeaseDefend	1	1
PickFromFloor	1	1
BullyObject	1	1
StealItem	1	1
ClaimBackItem	1	2
ClaimItem	1	2
DestroyItem	1	1
ItemDestroyedComplain	1	1
AskNotToThrowObject	1	1
WarnVictimBeforeThrowingObject	1	2
GotHitComplain	1	1
OrderToLeave	1	1
Attack	1	1

AskPity	1	1
WalkAwayVictim	1	1
GloatVictory	1	1
AnnoyVictim	2	1
LeaveFrom	1	1
AggressiveQuestion	1	1
Humiliate	1	1
RespondToHumiliationPositive	1	1
RespondToHumiliationNegative	1	1
HumiliateThreat	1	1
RespondToHumiliationThreatPositive	1	1
HumiliateVictory	1	1
Follow	1	1
FollowAskWhy	1	1
FollowDontCare	1	1
FollowLeave	1	1
FollowLeaveFollow	1	1
Gossip	1	1
GossipLeave	1	1
GossipReinforce	1	1
GossipObject	1	1
GossipObjectAgree	1	1
GossipObjectDisagree	1	1
GossipReport	1	1
GossipReportAngryAnswer	1	1
GossipReportDontCareAnswer	1	1
GossipReportHelplessAnswer	1	1

Appendix B

The Point Nautilus Story

The back story that was given to participants of the Point Nautilus Experiment is provided below for reference.

Point Nautilus

About 2 miles off the west coast of Canada lies Seagull Island, a tiny rocky island that one can walk round in about 2 hours. The only building on Seagull Island is the lighthouse Point Nautilus, which since the 70s has also doubled as a hotel. The hotel guests are typically people that need a break from the stresses of modern life without losing certain comforts. It is a cosy place where guests sit warming themselves by the fireplace with a cup of tea or glass of whisky in one hand and a good book in the other.

A lot of the recent popularity of Point Nautilus is due to the hard work of Claire, the current owner. Claire practically runs the place by herself and is determined to make every guest feel as comfortable as possible. She prides herself on her cooking skills and prepares wonderful meals for her guests, mostly using the fresh catch of the day. All this kitchen work together with managing and cleaning the hotel keep Claire extremely busy. Her only assistance and the only other permanent resident of Seagull Island is Jack the old lighthouse keeper who helps by ferrying guests to and from the mainland. Jack's boat is the only way on and off Seagull Island. On his daily boat trips Jack also brings supplies, such as fish caught by the local fishermen.

Claire became the owner of Point Nautilus 5 years ago. Originally a city-dweller, 10 years ago she found out her fiance had cheated on her and felt she had to get away from everything. She ended up taking refuge from the world in Point Nautilus. She fell in love with the landscape, the solitude, the fresh air and the sounds of nature, unspoilt by the noise of civilisation. The owner back then was an old lady, Mrs. Miller, who like Jack the lighthouse keeper immediately took a liking to this

sad young woman. Claire ended up helping Mrs. Miller out and became the cook of Point Nautilus. After 5 years Mrs. Miller died and it came as a big surprise to Claire to hear that she had inherited the place.

With the inheritance came more responsibility and while she loved this new challenge in her life, something was still missing. While she got over the sadness about her fiancé's betrayal, she often felt lonely. She had always imagined that by now, well into her 30s, she would have a little family, a handsome husband, 2 children and maybe a dog. She felt that it might be time to move on and get back to the city. But who would take care of the hotel then?

The answer seemed to arrive one day when a young writer from Calgary named Humphrey began a long term residency at Point Nautilus in search of a quiet place to finish his novel. It was love at first sight. Claire and Humphrey took long walks on the beach together, he would read to her what he had written in the evenings at the fireplace and on clear nights they would hold hands and watch the stars. Their romance had gone on for 3 months when he finished his book. For the first time since he arrived he had to leave the island to talk to publishers in Calgary.

They could hardly stand being separated and Claire eagerly waited for his return. After he had left for only a few hours she needed to hear his voice again. By now he should have arrived in Calgary. When the mobile phone number he had given her did not work, she was not overly worried at first. The plane must be delayed, she thought. But as the hours passed, she grew uneasy. Eventually Claire's worry turned into panic. The agreed date of Humphrey's return passed and there was still no word from him. The police were of little help. They told her that no airline had any records of a passenger by his name travelling into Calgary.

Claire had a million questions going through her head: Had something happened to him? Or could he have abandoned her? She couldn't believe that but then again, she had misplaced her trust in men before. Was everything he told her a lie including his name? That could explain why his phone number didn't work and why he was not listed on any of the flights? He hadn't paid for his 3 month stay, was he just trying to avoid the hotel bill? Was he on his way to another remote hotel by now, charming his way into the heart of another lonely hotel keeper with his romantic poet routine? Claire was conflicted and didn't know what to think but as the weeks passed, she knew one thing with certainty: she was expecting his child.

Humphrey did not, in fact, want to leave Claire and his love for her was genuine.

When he left Seagull Island he felt really bad that he hadn't told her the truth about his phobia of flying, but it was something he was inexplicably ashamed of. He had told her he was taking the plane when he was really travelling by train, because the truth would have undoubtedly led to more questions and eventually have exposed his phobia. But the choice of transportation turned out to have fateful consequences.

At his change in Vancouver where he had a few hours to kill until boarding the next train, he ventured into town to find something to eat. After his meal he was on his way back to the train station when he realised that he had left his bag, containing his wallet, his tickets, his phone and his novel, in the restaurant. Worried that someone might have stolen it, he rushed back to the restaurant. Crossing a street without looking, he ran directly into an oncoming bus. The mobile phone in his bag had been switched off, after all he had to pretend to be on a plane.

7 months later

It is a stormy day. The work at Nautilus Point has become increasingly difficult for Claire to manage as her pregnancy progresses and as a result the lighthouse hotel is temporarily closed. Claire is sitting in front of the fireplace, imagining her future with a baby and wondering where Humphrey is right now. The telephone rings. It is Jack calling from the main land telling her that the storm is worsening and that he can't make it to the island in this weather. She can only barely make out what he is saying and before they can finish their conversation the phone goes dead. This happens regularly out here in bad weather but Claire is a bit worried about her isolation given her pregnancy. The due date is still 4 weeks away, but she would feel better if she knew that she was not alone on an island without any means of communication.

Just as she thinks this, there is a knock on the door and she hears a female voice shouting:

"Hello, is anybody there?"

Surprised, Claire opens the door and finds a woman in a soaking wet rain coat standing in front of her.

"Oh thank God, someone is here. We were sailing but the weather was so bad, I thought we might be shipwrecked, but then we saw this lighthouse and made it to the dock. My boyfriend is sailing, he is mooring the boat right now. I'm so terribly sorry to intrude on you like this but could we please come in?" the woman blurts out.

"Of course" Claire answers.

"This is normally a hotel, when I'm not..." she points at her belly *"...like this."*

The woman smiles at her.

"So we have plenty of space here and this is no weather for you to be out sailing. Please come in. Oh and my name is Claire."

"Thank you so much and nice to meet you Claire, I am Susan and my boyfriend's name is John." the woman replies.

"He should be here in a minute."

As the two women enter the corridor Susan says:

"What a beautiful place you have Claire, I really like it."

She pauses awkwardly.

"Oh, before I forget, there is something about John I should tell you, in case you find his behaviour a bit peculiar."

Claire raises her eyebrows, curious what this is supposed to mean and wondering whether she may have made the wrong decision letting this woman in. Susan doesn't seem to notice as her gaze trails off into the distance and her face takes on a sad look.

"John suffers from amnesia and we have no idea who he really is. I am a doctor at a hospital in Vancouver and he was brought in one day run over by a bus. He is very lucky to even be alive. When he came out of his coma it soon became clear he had completely lost his memory. He couldn't remember his name or where he came from. He didn't have any ID on him when he was brought in, so we gave him a name and started to call him John. The hospital staff were the only people he knew in the whole world."

"But I thought you said you were his girlfriend, not his doctor?" Claire says in confusion.

Susan blushes

"We fell in love. I couldn't help myself. But a doctor can't be romantically involved with a patient - I had to resign. But I didn't care. I'd had enough of my job at the hospital anyway, too long hours and no time for a private life. So we decided to start a new life somewhere together and here we are, sailing the world. Oh, I'm sorry, I just keep on talking."

"Oh no, don't worry" Claire interrupts.

"What an extraordinary story, I thought this only happens in films. Is there a chance he can remember who he is?"

"Hard to say." Susan replies. *"There is a lot about how the human brain works that we don't understand. But you know, sometimes I hope he won't remember. We are so happy right now and who knows what the memories could destroy."*

As they talk, the two women hear footsteps approaching through the rain.

"John, we were very lucky, we can stay here for the night, isn't this great news? And it's such a cosy place." Susan excitedly calls out.

"Come and meet Claire."

As the man steps into the light, Claire cannot believe her eyes as she sees Humphrey coming in, but it suddenly all makes sense. All these months that she thought he left her like she was left before. He hadn't betrayed her after all. Or did he? What was he doing in Vancouver? Was he running away from her after all and this was how fate decided to punish him? Or was it all a tragic accident? Did it even matter? Was he not a totally different man now anyway? He patiently holds out his hand and with a friendly smile repeats:

"Nice to meet you Claire."

"Claire, are you alright?" Susan asks.

Claire feels a sharp bolt of pain in her belly and thinks oh no, the baby. His baby.

...

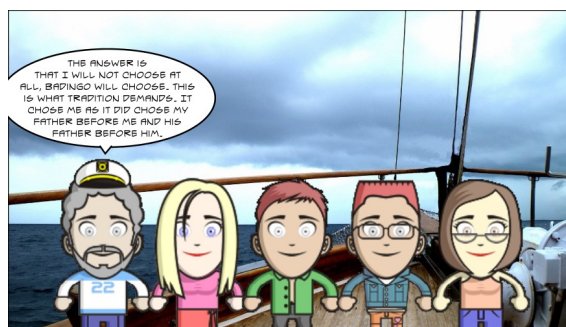
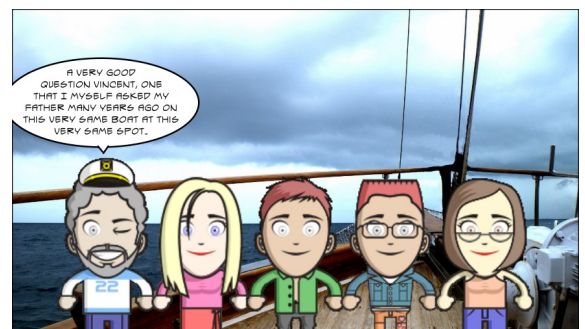
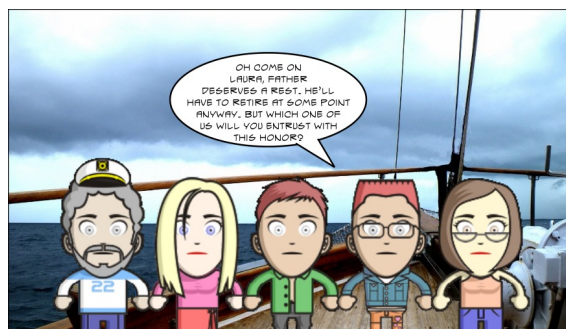
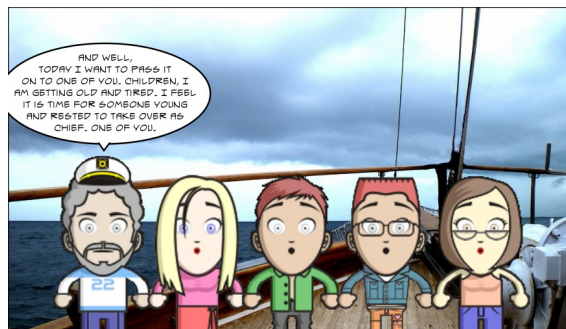
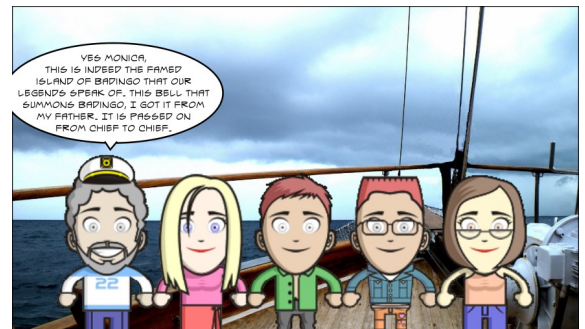
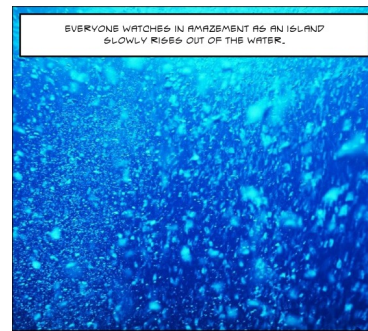
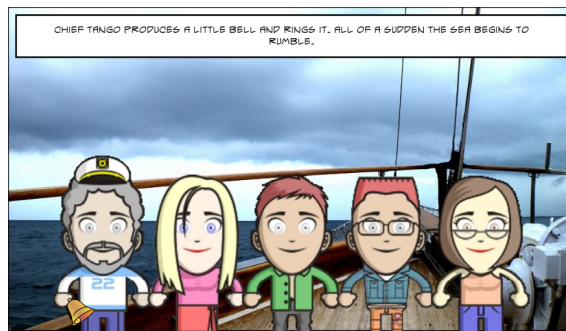
Appendix C

The Seagnomes Story

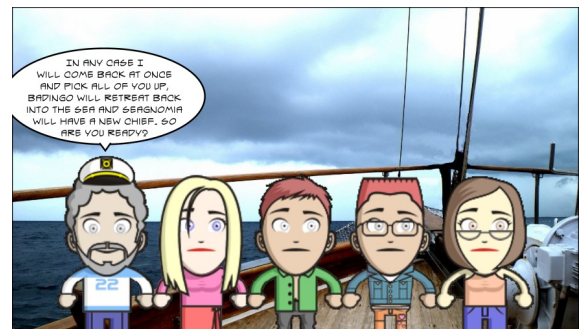
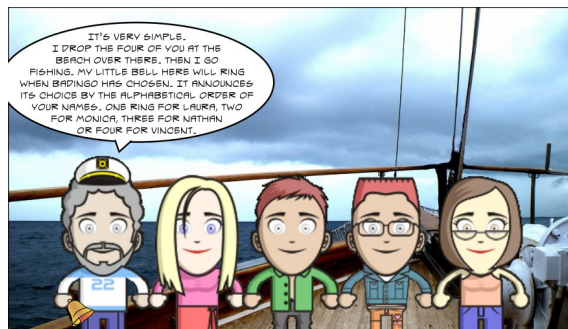
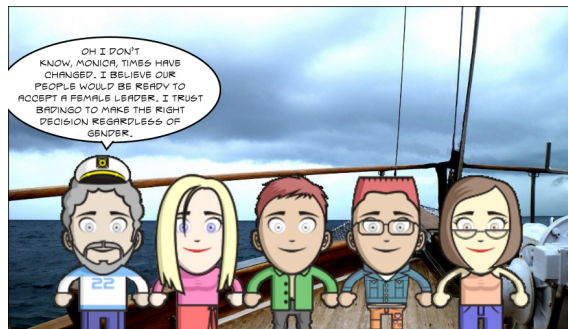
The initial pre-authored comics back story provided to the participants of the Seagnomes experiment is shown below.



Chapter C: The Seagnomes Story



Chapter C: The Seagnomes Story



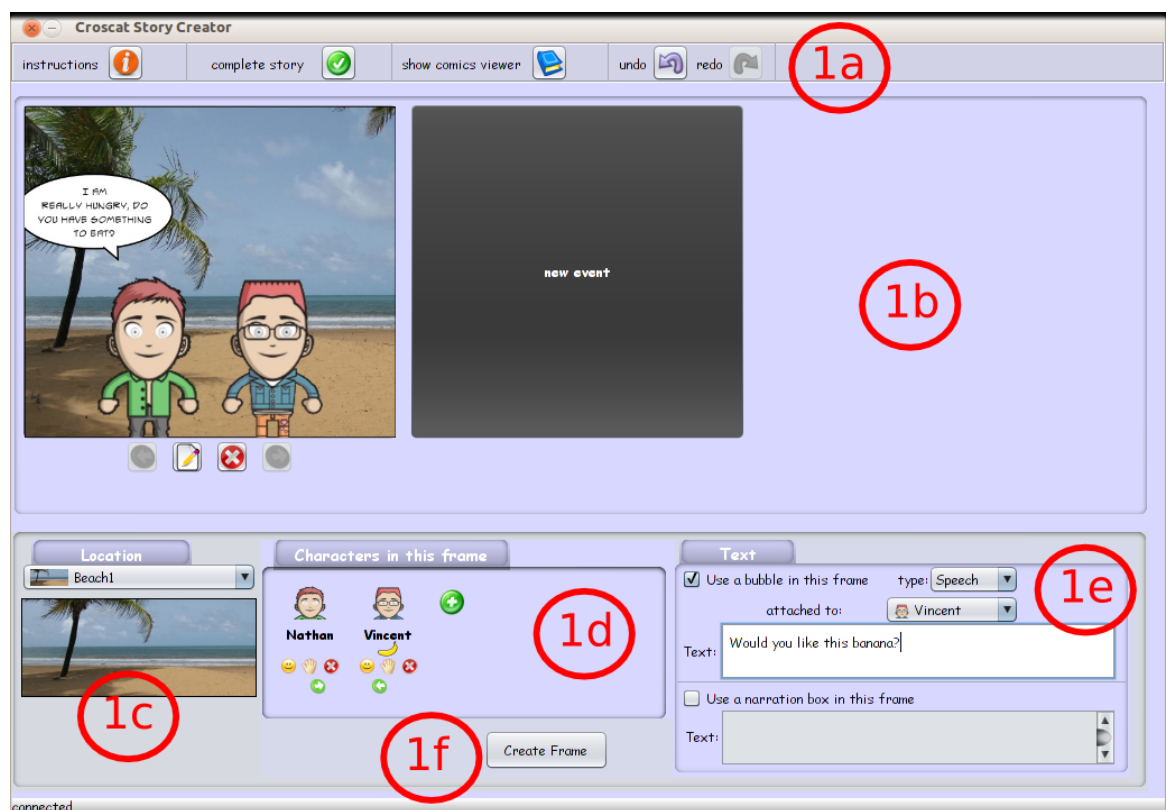
Appendix D

CROSCAT usage instructions

The following usage instructions were given to participants of the CROSCAT study.

How to use the CROSCAT software

The screenshot below shows what CROSCAT looks like.



Toolbar (1a): There are five buttons on the toolbar. The instructions button will open a browser and bring you back to this website. Use the "complete story" button to finish the experiment. "show comics viewer" opens a dialog which displays all the back story comic plus all the frames you have created so far. Finally undo and redo allow you to correct mistakes (e.g. when you accidentally deleted a frame).

Timeline **(1b)**: The time line shows all the comics frames that you have created so far in order (from left to right). After a few frames have been created it will display a scrollbar at the bottom. The timeline does not show the back story (to view that, you need to use the comics viewer). Double click on any frame to display it in full size in the comics viewer. Underneath each frame there are buttons that allow you to change the order of frames, delete or edit a frame. The right most frame of the time line is always called "new event". This is a placeholder for the next frame that will be created. New frames are therefore always inserted at the end (right) but can be moved around afterwards. If the frame editor does not allow you to edit a new frame, click on the "new event" frame once to select it.

The bottom third of the screen is the frame editor. It consists of:

Location Picker **(1c)**: Allows you to change the background / location of the comics frame.

Character Panel **(1d)**: Lists all the characters (in order) that will appear in this frame. Press the green plus button to add a character and the red x underneath a character to remove it. The green arrows (min 2 characters) allow you to change the location of the characters (in which order from left to right are they shown in the frame). The smiley icon allows you to change the facial expression of a character and the hand item allows you to put a single item into a character's hand (or to remove an item from a character's hand).

Text Panel **(1e)**: This allows you to place a bubble and /or narration box in the frame. You can chose between speech and thought bubbles. For bubbles you also have to chose a speaker / thinker (from one of the characters listed in the character panel obviously).

Create Frame Button **(1f)**: When you have selected a location, characters and some text you can press the create frame button. You can then directly proceed to edit the next frame. If you want to change something you can use the edit button in the time line to make minor changes to your frame. When editing instead of "Create Frame" you will see 2 buttons to "Update Frame" and "Cancel Editing". After editing you need to manually click on the "new event" in the time line to continue making new frames.

Appendix E

CROSCAT Graph Scoring Implementation

The following listing provides implementation details of the CROSCAT graph scoring implementation, specifically, the methods of the StoryNode class used for recursively resetting the score, calculating scoring prerequisites and calculating the final score. This appendix is an extension to Section 7.4, which describes the algorithm at a high level.

Listing E.1: Implementation details of the CROSCAT graph scoring implementation

```
public class StoryNode {

    /** children of this node */
    protected List<StoryNode> children;

    /** parent node or null if this is the root node*/
    private StoryNode parent;

    /** this will contain the final score for the node */
    private double score;

    /** what is the distance (in nodes) to a branching point in front */
    private int distBranchFront;

    /** what is the distance (in nodes) to a branching point at the back */
    private int distBranchBack;

    /** how many of this node's ancestors are branching points */
    private int noOfAncestorsThatBranch;

    /** a list with the distances from this node to each descendant leaf node */
    private List<Integer> distsEnd;
```

...

```

public void resetScoreRecurse() {
    score = 0;
    distBranchFront = 0;
    distBranchBack = 0;
    distsEnd.clear();
    noOfAncestorsThatBranch = 0;
    for (StoryNode child : children)
        child.resetScoreRecurse();
}

public void calcScoringPrerequisitesRecurse() {

    // calc dist front
    if (children.size() > 1)
        distBranchFront = 0;
    else if (parent==null) // root node with only 1 child,
        // set distBranchFront to 0 (treat beggining like a branching point)
        distBranchFront = 0;
    else // any other node that is not root and has not more than one child
        distBranchFront = parent.distBranchFront + 1;

    if (parent==null)
        noOfAncestorsThatBranch = 0;
    else if (parent.children.size() > 1)
        noOfAncestorsThatBranch = parent.noOfAncestorsThatBranch + 1;
    else
        noOfAncestorsThatBranch = parent.noOfAncestorsThatBranch;

    for (StoryNodeBase child : children)
        child.calcScoringPrerequisitesRecurse();

    // if we reach an end node start a linear backward recursion up the tree
    // note: we will traverse some nodes (e.g.) root node multiple times this way,
    // but that doesn't matter: the result will always be the same
    if (children.size()==0)
    {
        // set our distBranchBack rating to 0 (treat end like a branching point)
        distBranchBack = 0;
        // same for distance to end
        distsEnd.add(0);
        parent.calcScoringPrerequisitesRecurseBackwards(this,0);
    }
}

private void calcScoringPrerequisitesRecurseBackwards(

```

```

StoryNode origin,
int distEnd) {

    // score distance from back branching point
    if (children.size() > 1)
        distBranchBack = 0;
    else
        distBranchBack = origin.distBranchBack + 1;

    // score distance from this end (bonus points for being far away from end)
    distsEnd.add(distEnd+1);

    // move along in backwards direction
    if (parent!=null)
        parent.calcScoringPrerequisitesRecurseBackwards(this,distEnd+1);
}

public void scoreRecurse()
{
    // assign branch distance bonus
    score += Math.min(distBranchFront, distBranchBack) * BRANCH_DISTANCE_BONUS_FACTOR;

    // calculate and assign end distance bonus (avg of all end distances)
    double avgDistEnd = 0;
    for (int distEnd : distsEnd)
        avgDistEnd += distEnd;

    avgDistEnd /= distsEnd.size();
    score += avgDistEnd * END_DISTANCE_BONUS_FACTOR;

    // determine if antonym bonus is due and award it
    if (allowsAntonymExtension()) score+= ANTONYM_BONUS;

    // add branch extension penalty

    if (children.size()>MAX_BRANCH_FACTOR)
        score -= EXTEND_BRANCH_ABOVE_MAX_PENALTY * (children.size() - MAX_BRANCH_FACTOR);

    // penalty points for having branching ancestors
    score -= this.noOfAncestorsThatBranch * BRANCHING_ANCESTOR_PENALTY;

    // recursively score all children
    for (StoryNode child : children)
        child.scoreRecurse();
}
}

```

Appendix F

Supplementary Digital Materials

Some materials that have been produced during the course of this PhD, may be of interest to the reader but do not fit reasonably within this appendices section. This includes the source code for the two authoring systems implemented in the course of this PhD (ENGIMA and CROSCAT) and the experimental data collected during the course of conducting the “Point Nautilus” and “Seagnomes” experiments. These materials are collected and provided on the accompanying DVD. Please refer to the README.txt file in the DVD’s root directory for more information.