

Patterns for Story Authoring Tools

Marty Kauhanen

School of Computer Science / Carleton University / Ottawa / Canada
mkauhanen@gmail.com

Chris Eaket

Institute for Comparative Studies in Literature, Art and Culture / Carleton University / Ottawa / Canada

Robert Biddle

Human Oriented Technology Laboratory / Department of Psychology / Carleton University / Ottawa / Canada

Abstract

Interactive narratives are an important element in video games and serious games alike, however, their authoring tools have their roots in programming language and programming environments. While these end-user environments for interactive narratives provide a means for users to create custom content, they are not necessarily intuitive or easy to use for non-programmers. This paper provides the basis of a pattern language of design for authoring tools for story-tellers that are not programmers.

Key Words: Serious Games, Interactive Fiction, End-User Development, Domain Specific Languages

1.0 Introduction

In this paper we identify and discuss patterns for supporting the creation of new content for computer games. We concentrate on games that work as interactive fiction, where the creation of content therefore resembles story-telling.

In the remainder of this section we explain our motivation in more detail, especially our interest in “serious games” and our view that game scripting deserves a kind of domain specific language. We then present the patterns we have identified, beginning with an overview of their purpose and how they relate to each other, and then presenting each pattern in detail.

1.1 *Serious Games and Immersion*

Why care about games? The first and foremost reason is simple: games are big business. For example, the estimated number of active subscriptions to MMOGs (massively-multiplayer online games) is over 12 million worldwide [24]. In addition, it has been shown that people, across a wide demographic of age and gender, devote a significant amount of time ‘playing games’ [30]. The second reason is that games may play other roles than entertainment.

Our group has been studying the principles of serious game design. Barr shows that video games contain a value system that is perceived and adopted by the players, shaping the game-

play and that the study of serious games in a human computer interaction (HCI) mindset will glean insights into their triple-natures: as games, as media and as software [29]. Dormann, et al. explore the use a computer role-playing game as a means to support the informal learning of Antarctica, global warming, including its issues and controversies [23]. Using the game's authoring tools, they designed a rich environment and found that a rich use of metaphor, meaningful interactions and narratives contributed to "an increased sense of presence, level of immersion and interest within a game learning environment" [23], thus achieving the goal of informal learning support.

Yee studies and categorizes the motivations of MMOG players into three categories and ten sub-categories: Advancement (advancement, mechanics, completion), Achievement (socializing, relationship, teamwork) and Immersion (discovery, role-play, customization, escapism) [25]. Yee discovers that motivations of advancement, achievement and immersion are largely uncorrelated. For example, those players primarily motivated by advancement, like gaining experience points to improve or 'level up' their character, may also be motivated by socializing and immersive qualities of the game. Value systems in video games and their subsequent adoption by players during game-play [29] illustrate the validity of video games as tools for non-entertainment purposes. Both Dormann's and Yee's work suggest that immersion is an important part of games and that meaningful interactivity, narratives, metaphor and a rich environments are ways of achieving it.

1.2 Interactive Fiction and Interactive Drama

Interactive fiction (IF) is a literary narrative genre often employed in video games. IF systems accept textual commands from the user, in the form of verb-noun pairs or natural language statements, to influence the environment and determine the outcome of the story or game. Interactive fiction is goal-oriented. Historically, interactive fictions were heavily influenced by Dungeons and Dragons [1] and were usually adventure-themed [2] with strong elements of exploration, mystery and puzzle-solving. Others interactive fiction systems, such as Graphic StoryWriter [31], were created as a story environment for early readers and thus were influenced by children's stories.

Interactive drama (ID) is quite similar to interactive fiction in that it refers to a type of drama where the audience plays an active role, modifying the drama by changing the course of actions [3]. A contemporary example of an interactive drama is Façade [4], a first-person, 3-D virtual world that allows the player to interact with computer-controlled characters. Interactive drama is experience-oriented. For the purpose of this paper, we shall include both interactive fiction and interactive drama under the umbrella terms: interactive narrative or interactive story. A common characteristic of interactive narratives is that they have multiple, and often innumerable, outcomes.

A related area is collaborative story-telling [32] where several people collaborate on the story creation. In our present work, however, we are concentrating on tools for single story authors within the framework of an existing game structure. Although this is a kind of collaborative story-telling, the collaboration is between the new author, the game designers, and future players.

1.3 Domain Specific Languages and End-User Development

The remainder of the paper is dedicated to patterns that describe part of a pattern language used for the design of interactive narrative software systems, in particular their end-user development (EUD) tools and environments [27]. We have identified patterns suggesting that instead of relying upon the language, structure, semantics and terminology of other domains, like programming, one should instead consider the language and concepts from the domain of story-telling when designing story-authoring tools. Interactive narratives are an important element in entertaining games and serious games alike, however their authoring tools have their roots in programming language and programming environments. These tools constitute an important domain specific language [26, 28]. While these end-user environments for interactive narratives provide a means for users to create custom content, they are not necessarily intuitive or easy to use for non-programmers. This paper provides the basis of a pattern language of design for authoring tools for story-tellers that are not programmers.

2.0 The Pattern Language

STORYTELLER'S CORNER is the top level in the pattern language for design of end-user authoring tools for interactive narratives. It is the place where storytellers will create their own stories and content in an interactive narrative system. Within the STORYTELLER'S CORNER, authors should:

- (a) be comfortable and not impeded by the authoring environment,
- (b) be supported with a library of story resources
- (c) be able have their stories read back to them, even if they are not complete.

These three issues are addressed in EVERYONE HAS A STORY TO TELL, ALL STORYTELLING IS RE-TELLING and READ IT BACK TO ME, respectively.

EVERYONE HAS A STORY TO TELL is broken down into three sub-patterns: TELL THE STORY IN YOUR OWN WORDS, involving the use of natural language, BIND THE STORY, involving the use of metaphor and IT GOES SOMETHING LIKE THIS, involving the use of the terminology of stories over technical one within the authoring environment.

ALL STORYTELLING IS RE-TELLING is supported by the sub-pattern, TRADING STORIES, allowing authors to share their custom resources.

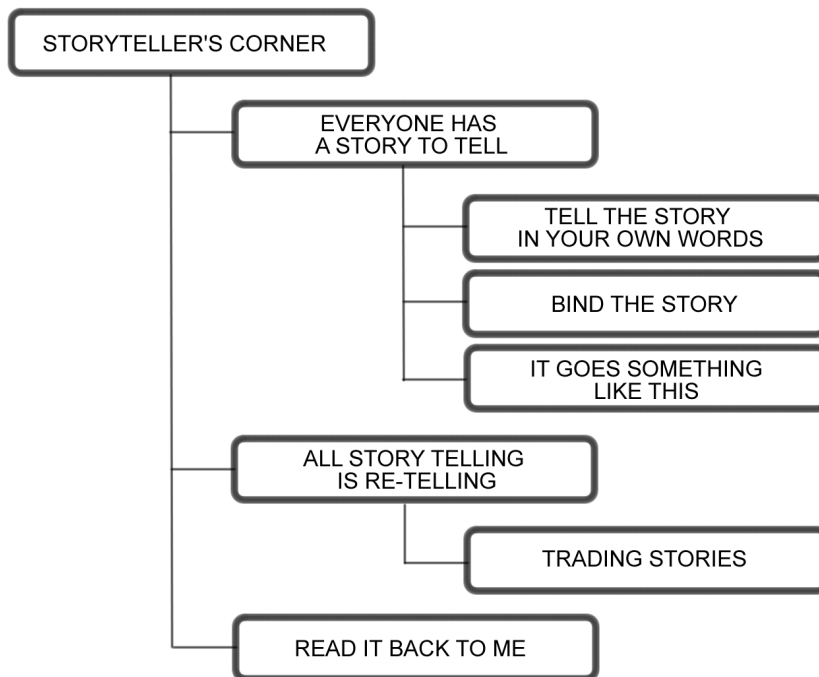


Figure 1 – The basis of a pattern language for design of end-user development environments for interactive narratives.

2.1 STORYTELLER’S CORNER

One of the challenges for video game developers is fostering a culture of commitment for the game. Another challenge is creating scale and variety within the game sufficient to justify such devotion. Both these challenges can be met by providing the means for customers to create their own custom content. Many games that have adopted this strategy have provided authoring tools or the ability for customers to create ‘mods’, or modifications, resulting in active and creative ‘mod’ communities. One only has to search the web for “Civilization 4 mods” and “Age of Empires mods” for many examples.

The success of a video game involves player commitment and variety on a large scale.

Therefore: Provide the tools necessary to create custom content for games.

There are many interactive narrative systems that provide the tools for the creation of custom content. An example of such a tool is the Neverwinter Nights (NWN) Aurora Toolset. The Aurora Toolset allows players to create their own modules for BioWare’s popular Neverwinter Nights computer role-playing game [5]. The modules may be loaded and played in the NWN game. The custom modules may then be shared with others and if set up on a server, played simultaneously by up to 64 players. Released in 2002, NWN’s popularity is aptly reflected by the amount of supplementary material produced and released by BioWare. Between 2002 and 2006, six premium add-on modules, two expansion packs and a number of editions of NWN were released. In the last quarter of 2006, a sequel, Neverwinter Nights 2, hit the shelves. It too has a tool for content creation.

While video games, NWN and NWN2 may also be considered an interactive narrative systems because the player interacts with computer characters in a coherent storyline containing elements of goal- and experience-oriented play.

The consequence of not providing story-authoring tools is the inability for users to create their own stories. The user becomes reliant upon the developers for new content. *Façade* (a first-person, real-time, AI-driven, non-linear, generative, 3-D interactive drama) is cutting edge but, at the moment, lacks an authoring tool [4]. With over half a million downloads since July, 2005, *Façade* unfortunately boasts only one act. While the act may be replayed over and over with different results, it remains limited in its appeal.

As illustrated in TELL THE STORY IN YOUR OWN WORDS, the technology behind *Façade* is complex. No doubt the complexity of creating and adding an authoring tool would be compounded by the complexities of the *Façade* architecture itself.

There are two areas of concern for programmers. First, the addition of authoring tools adds complexity and difficulty to the task of programming the entire system. For example, the game engine cannot be static; it must dynamic, capable of accepting a variety of different inputs as determined by the user. The second area of concern is that a story authoring environment must also be easy to use. The sub-patterns of STORYTELLER'S CORNER attempt to mitigate the latter, but leave the former to the domain of programming.

2.2 EVERYONE HAS A STORY TO TELL

An interactive narrative system is a piece of software and the act of scripting a story is similar to writing a program. Indeed, game systems *are* software and are created by game developers by programming. But game players and potential authors of new content are not always programmers, nor do they need to be in order to add content at the level of the story.

While interactive narrative software systems require some technical knowledge, most story-writers and authors are not programmers. This may make it difficult for authors to tell a story.

Therefore: Make the story-authoring tool accessible to non-programmers. EVERYONE HAS A STORY TO TELL and anyone who can tell a story should be able to script one at the higher level of story-telling, rather than the lower level of programming.

While story-writing and programming are not mutually exclusive, it does not make sense to burden writers with learning to program in order to create interactive narratives. The solution is to provide a higher level representation of the programming.

As discussed in the parent pattern, the increased complexity of programming the system and the tool itself is a tradeoff for an increase in the tool's ease of use. This is a natural progression in technology, however. Few people still program in Assembly code, a low-level language that deals directly with memory addresses, registers and stacks, but choose to program in more abstract, high-level languages, such as C++ or Java that deal in terms of Boolean expressions, classes, functions and variables.

The Aurora Toolset, for example, is an even higher-level language still. It is very graphical in nature, allowing authors to drag objects, such as monsters, buildings and trees, onto an area map. The objects are editable through property screens that consist of fields of parameters (see Figure 6).

The Adrift Generator, the authoring tool for the Adrift text adventure software, takes a similar approach with respect to programming [21]:

Instead of having to learn a new adventure programming language, ADRIFT Generator takes all the difficulty away leaving you with a simple, yet powerful game designer. Adventures are built up by adding rooms, objects, tasks, events and characters. All you have to do is type in the descriptions, and select how everything interacts with each other from pull down menus and lists.

The creators of *Façade* realize the advantage of a higher level language for authoring, stating [10]:

Our current tools and authoring techniques require programming expertise; in fact, even expert programmers familiar with language such as C++ and Java will have to learn new ways of thinking about programming to program in ABL. For this reason, we are not publicly releasing the authoring tools at this time, though we will be working towards the future release of higher-level authoring tools that enable writers and artists to create Façade-like content.

Their custom language, A Behaviour Language (ABL), is a behaviour language that supports sequential, parallel and joint behaviours and is “effectively a multi-threaded programming language [and] challenging to program in, even for experienced coders” [11]. Quite obviously the designers of *Façade* realize that due to the complexity of their system, a STORYTELLER’S CORNER that adopts the pattern of EVERYONE HAS A STORY TO TELL is required.

There are several sub-patterns that one may employ to create a tool that limits the amount of programming knowledge required to script a story. The three sub-patterns of EVERYONE HAS A STORY TO TELL are: TELL THE STORY IN YOUR OWN WORDS, BIND THE STORY and IT GOES SOMETHING LIKE THIS.

2.2.1 TELL THE STORY IN YOUR OWN WORDS

Programming languages are used to write software. This means that for most video game systems and interactive narrative systems, the ability to create custom content is tied to the author’s ability to program. This is especially true if a video game system does not provide an authoring tool. STORYTELLER’S CORNER, however, provides the authoring tool. EVERYONE HAS A STORY TO TELL tells us the pattern to employ to reduce technical knowledge requirements: give the author a higher-level environment to work in. But how do you provide a way to script the logic of a story and the story itself that is both easy to read yet sufficiently powerful?

The narrative structure necessary for interactive story-telling requires management of sequences, conditions, repetition, and remembering details that happen along the way. Programming languages can do all this but they are difficult to learn and difficult to read. In the end, the program's code neither reads nor looks like the story that it represents. Poor readability results in a longer time to learn the system and problems with the maintainability of the code. In other words, an ordinary programming language will not reveal the author's intent.

Therefore: Use a limited natural language as a programming language.

Instead of relying upon programming languages, a story-authoring system could adopt rules for semantics and English sentences making it much closer to the structure of work in creating stories. Instead of objects, classes and methods, the author could concentrate on writing more natural sentences and dialogue. Natural language is used to script the story itself.

We see the evolution from object-oriented code to natural language-based code in the *Inform* [6] interactive fiction design system. In Inform 6, the language used in story creation is essentially an object-oriented programming language that is compiled to a file which is then interpreted in the user's machine. Below are two code snippets of Inform 6 code from the 'Hello World' of interactive fiction stories called: *Cloak of Darkness* [7].

```
Object cloak "velvet cloak"
with name 'handsome' 'dark' 'black' 'velvet' 'satin' 'cloak',
description
  "A handsome cloak, of velvet trimmed with satin, and slightly
  spattered with raindrops. Its blackness is so deep that it
  almost seems to suck light from the room.",
before [;
  Drop, PutOn :
    if (location == cloakroom) {
      give bar light;
      if (action == ##PutOn && self has general) {
        give self ~general;
        score++;
      }
    }
  else
    "This isn't the best place to leave a smart cloak
    lying around.";
],
after [;
  Take: give bar ~light;
],
has clothing general;
```

Figure 2– Code snippet from the Inform 6 version of *Cloak of Darkness*, describing the behaviour of the velvet cloak object. This code is from the middle of the file.

```
[ Initialise;
  location = foyer;
  move cloak to player;
  give cloak worn;
  "^^Hurrying through the rainswept November night, you're glad to see the
  bright lights of the Opera House. It's surprising that there aren't more
  people about but, hey, what do you expect in a cheap demo game...?^^";
];
```

Figure 3– A second code snippet from *Cloak of Darkness*, written in Inform 6. This snippet is located near the end of the file and is the code that instantiates the game-play.

Notice that the above code does not look anything like a story and reads very poorly. The “Object cloak” is the class definition of a cloak that describes the behaviours of a cloak object the game and the “Initialize” portion, in the second snippet, is the part of the text that the player sees at the beginning of the interactive narrative. Clearly, one must understand classes, methods, instances, if-then-else statements and variables to understand (and write) the above code.

In Inform 7, the objects, class and methods are replaced by a rule-oriented design and natural language where “the activity of programming IF is a form of dialogue between programmer and computer to reach a state with which both are content, [...] not unlike the activity of playing IF” [8].

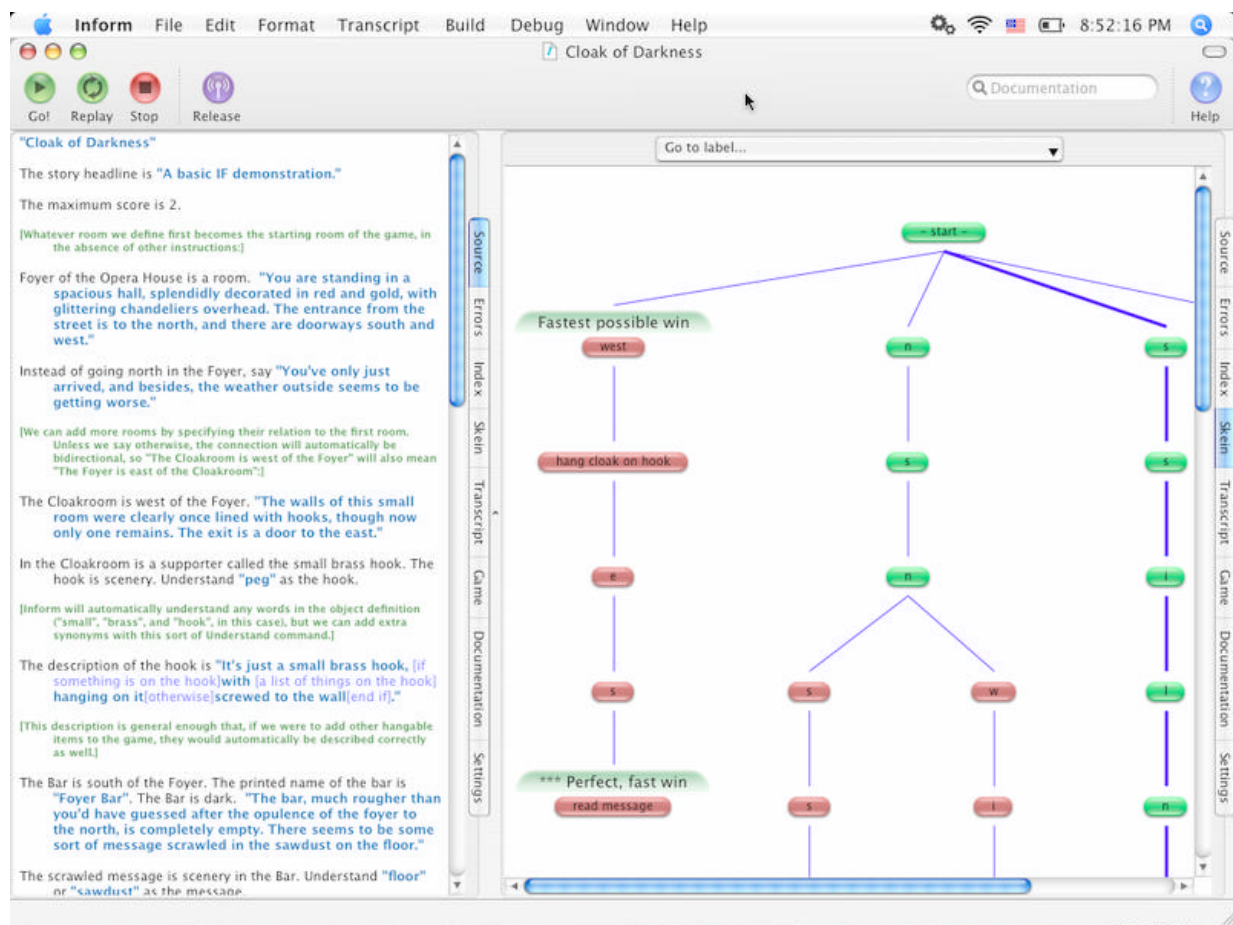


Figure 4– Screenshot of Inform 7 showing the natural language “source code” of the *Cloak of Darkness* [7].

Daly summarizes the advantages and disadvantages of using natural language in Inform 7 [9]. Advantages include: the ease of reading, the elimination of IF-specific errors, the limiting of loop structures and the lack programming concepts like procedure names and variables.

There are some danger spots with using natural language. Daly notes that the disadvantages of using natural language “include occasional awkwardness of natural language and the possible penchant for writers to misunderstand the meaning of paragraph of code when skimming” [9]. Also, when writing natural language-based rules, one must consider the

alternate verbs that a player may use to perform an action. For example, the following rule appears in the Inform 7 Cloak of Darkness code [7]:

```
Understand "hang [something held] on [something]" as putting it on.
```

Basically people *put on* cloaks to wear them, but *hang* cloaks on brass hooks. This potentially could be time consuming to write rules for all the alternatives, but consider the Inform 6 version of the above rule [7]:

```
Include "Grammar";  
  
Verb 'hang'      * held 'on' noun    -> PutOn;
```

While some diehard programmers may consider natural language programming too verbose and less powerful than an object-oriented approach, they would be happy to know that Inform 7 supports the ability use Inform 6 code in its projects [9].

TELL THE STORY IN YOUR OWN WORDS approach is one of three sub-patterns in EVERYONE HAS A STORY TO TELL. The other two, possibly complimentary, patterns in design are: BIND THE STORY and IT GOES SOMETHING LIKE THIS.

2.2.2 BIND THE STORY

Discussed in EVERYONE HAS A STORY TO TELL and TELL THE STORY IN YOUR OWN WORDS was the need for a higher-level representation of the code and how natural language may be used in lieu of programming languages, respectively. The organization of the code and environment is addressed in BIND THE STORY.

While the work of creating an interactive narrative is similar to that of writing a program, the motivations and criteria for success are quite different for stories than for programs. Just as the language of the story should be more like a story and less like a program, so too must the STORYTELLER'S CORNER support the *process* followed by the content author in a way more suitable to the work and space of story-telling. The visibility and understandability of the environment and story language within are important.

Therefore: Employ story metaphors to help with the structure of work in story design and the organization of story elements in the authoring tool.

By leveraging concepts in the domain of the stories and drama, the creation of interactive narratives can be less arduous and more intuitive for the author. In the interface of Inform 7, Nelson intentionally uses a book metaphor, where a project is a single book and the file system view of resources has been removed entirely [8]. In Figure 4, we see a book with the source code on the left, written in a natural language divided into paragraphs, with the results of the natural language code displayed on the right. Few programming concepts are required to understand the interface. The paragraphs provide a story-like way of breaking up the natural language code.

The use of metaphor can be seen in end-user development environments for other domains. The canonical example is LabVIEW, a diagram-based programming environment used to create control programs for electronic equipment. LabVIEW, employs a wiring metaphor where block diagrams *are* the basis of the source code, leveraging the skills already possessed by engineers and scientists in that domain [18]. Another example of metaphor use is found in a software system that uses robotic agents on virtual tracks to program the behaviour of actual robots [17]. In addition to metaphor, these end-user environments are employing visual programming techniques, a much higher-level representation of the underlying code.

A conceivable problem area is that the use of metaphor in a design tool may seem contrived. For example, the designers of Inform 7 notably did not rename the menu item “Debug”, a technical term, as something that fit perfectly with the book metaphor. Debugging is essentially the act of finding and correcting the errors, or bugs, in a program. Debugging is different than reading through a story to see if it makes sense or even play-testing an interactive story (the Run command in Inform 7). A debugging tool finds errors in logic and syntax automatically and informs the user of the errors. Implicitly, the designers of Inform 7 say: adopt aspects of other metaphors when such aspects are needed but cannot be accurately portrayed in the main story-book metaphor.

2.2.3 IT GOES SOMETHING LIKE THIS

Often the terminology used in an authoring environment is borrowed from programming environments. Would-be authors of interactive narratives may not be familiar with technical terms, error codes and programmatic commands. The intent behind the terms, codes and commands is not to confuse the author but to help.

The supporting environment for creation of interactive content is similar in nature to that for supporting software development in that one manages static “text” that is stored and later used to produce dynamic behaviour. But the vocabulary for managing all this that as grown up within software development is unfamiliar to story-tellers.

Therefore: In the authoring tool, use terminology that either relates to story creation or that is less technical in nature.

By incorporating terminology from the domain of the stories and drama, the creation of interactive narratives may be more intuitive for the author. The author will be more inclined to use commands that sound less technically intimidating and make better use of the error messages.

The use of technical terms and commands is indicative of the evolution of authoring tools from programming environments to higher-level story-creation environments, as previously illustrated with Inform 6 and Inform 7. While Inform 7 does keep the term ‘Debug’ as it doesn’t seem to have a story-related equivalent, it does minimize the technical nature of error messages, themselves. Inform 7 provides a different type of error message, called problems, that are not like traditional programming compiler errors. Problems make reference to sections and chapters, not lines numbers, and make liberal use of quotations in order to make suggestions as why the problem occurred. In addition, the compile-run command in Inform 7 is labeled simply ‘Go’, a much more friendly term to a non-programmer [8].

In the Façade architecture (in addition to the ABL language, an animation engine and an interpreter for natural language input) there is a drama manager that “sequences dramatic beats” [12] where beats are a basic unit of measurement in a story. The use of beats is both a use of IT GOES SOMETHING LIKE THIS, dealing with beats as a unit of measurement, also an example of BIND THE STORY, as it adopts the beat as a drama metaphor.

In the Adrift architecture [21], the Adrift Generator and the Adrift Runner to a programmer are essentially the ‘compiler’ and ‘interpreter’, respectively. ‘Generator’ and ‘runner’ are much more understandable and operational terms for non-programmers.

An example of a system that employs all three sub-patterns of EVERYONE HAS A STORY TO TELL is the Aurora Toolset for NWN. NWN is a fantasy-themed game set in a popular Dungeons & Dragons [14] campaign world called the Forgotten Realms [15]. The game mechanic of the game is based on the d20 System [13], the same as the pencil-and-paper version of Dungeons & Dragons. A game mechanic is a set of rules used to determine the outcomes of actions in the game, where skills and abilities of the characters and their opponents along with environmental factors influence the probabilistic outcome.

NWN makes use of the d20 System for character advancement and statistics and as a basis for conflict resolution. Following suit, the Aurora Toolset allows authors to create ‘encounters’ and creatures whose attributes are similar to the pen-and-pencil version. Creating and editing the ‘statistics’ of a creature or character in the Aurora Toolset will be a familiar exercise for those knowledgeable in d20 System and D&D domains. The similarity is illustrated in Figures 5 and 6.

DUNGEONS & DRAGONS
CHARACTER RECORD SHEET

CHARACTER NAME _____ PLAYER _____
 CLASS AND LEVEL _____ RACE _____ ALIGNMENT _____ DEITY _____
 SIZE _____ AGE _____ GENDER _____ HEIGHT _____ WEIGHT _____ EYES _____ HAIR _____ SKIN _____

ABILITY NAME	ABILITY SCORE	ABILITY MODIFIER	TEMPORARY SCORE	TEMPORARY MODIFIER	TOTAL
STR STRENGTH					
DEX DEXTERITY					
CON CONSTITUTION					
INT INTELLIGENCE					
WIS WISDOM					
CHA CHARISMA					

SAVING THROWS

	TOTAL	BASE SAVE	ABILITY MODIFIER	MAGIC MODIFIER	MISC. MODIFIER	TEMPORARY MODIFIER	CONDITIONAL MODIFIERS
FORTITUDE (CONSTITUTION)							
REFLEX (DEXTERITY)							
WILL (WISDOM)							

HP HIT POINTS _____

AC ARMOR CLASS TOTAL = 10 + _____ + _____ + _____ + _____ + _____ + _____ + _____

TOUCH ARMOR CLASS _____

FLAT-FOOTED ARMOR CLASS _____

INITIATIVE MODIFIER TOTAL = _____ + _____ + _____

SKILLS

SKILL NAME	KEY ABILITY	SKILL MODIFIER	ABILITY MODIFIER	RANKS	MISC. MODIFIER
<input type="checkbox"/> APPRAISE	INT				
<input type="checkbox"/> BALANCE	DEX				
<input type="checkbox"/> BLUFF	CHA				
<input type="checkbox"/> CLIMB	STR				
<input type="checkbox"/> CONCENTRATION	CON				
<input type="checkbox"/> CRAFT (_____)	INT				
<input type="checkbox"/> CRAFT (_____)	INT				
<input type="checkbox"/> CRAFT (_____)	INT				

Figure 5 – Top portion of the first page of a pen-and-paper D&D Character Sheet [16]

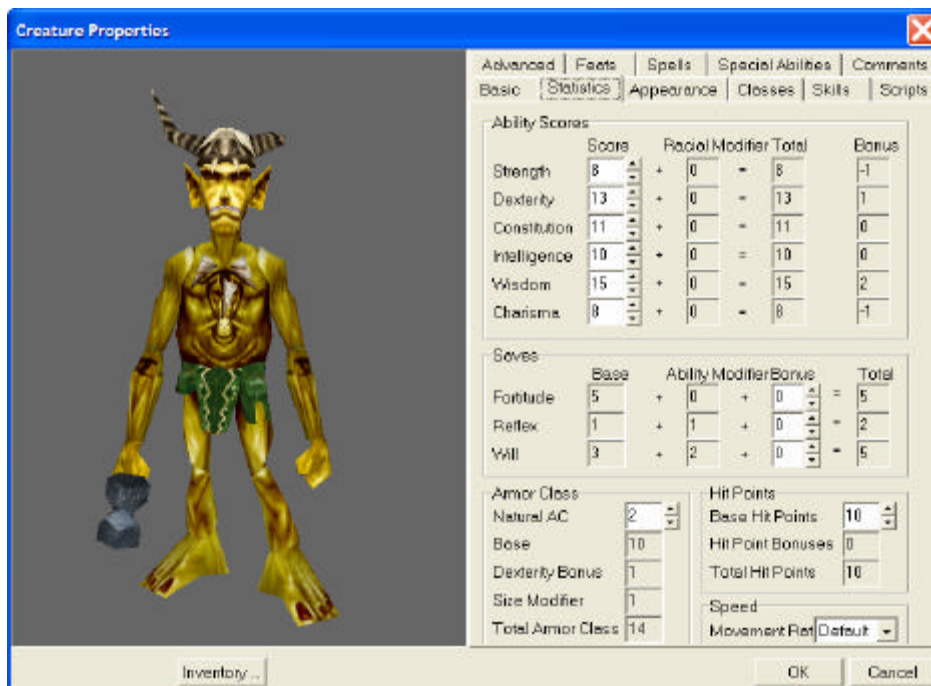


Figure 6 – Screenshot of a Creature Properties screen from the Aurora Toolset

The Aurora Toolset employs the pattern BIND THE STORY in its use of a pen-and-paper style D&D character sheet as a metaphor for creature and character creation. Closely related is the use of the pattern IT GOES SOMETHING LIKE THIS in the Aurora Toolset. An ‘encounter’ is the terminology used for a group creatures or characters that a player will meet and interact with in the game. In the toolset, such a grouping of creatures is still called an ‘encounter’. The use of IT GOES SOMETHING LIKE THIS helps keep the pen-and-paper metaphor intact. While the Aurora toolset does have elements of programmatic scripting, the dialogues themselves are written in the form of statements and responses as they would appear in the game, organized in a branching tree structure. This is a step in the direction of implementing TELL THE STORY IN YOUR OWN WORDS.

2.3 ALL STORY TELLING IS RE-TELLING

It is common for authors to re-use the components, characters and resources in their stories. Both ALL STORY TELLING IS RE-TELLING and TRADING STORIES address the organization of story components and their portability, respectively.

It is a difficult task to keep track of story resources, such as characters, components, text, titles, names and scripts. The problem is compounded by keeping track of multiple versions of the same resource, their locations and all the relevant details an author needs to know to effectively re-use. This places a mental burden upon the author or leads to the adoption of an informal secondary notation.

Therefore: Provide a common library of resources, preferably one that allows the addition of custom or edited content.

If you provide the environment for authors to create their own custom stories, then the environment should also provide a means of accessing common content and a place to store, access and organize custom content. Libraries allow authors to quickly access and add the common components to their interactive narratives.

Libraries are common in programming environments and are used as a part of common and custom resource organization environments like NetBeans and Visual Studio.

In the NWN Aurora Toolset and the Obsidian NWN2 Toolset, there is a library that authors may use to access the blueprints for objects, creatures and map features that they may wish to use in their project. The blueprints make adding a dwarf to their environment quick and simple, as opposed to programming 100 lines of code for the “threatening little dwarf” as done in the Inform 6 implementation of the classic interactive fiction, Advent [19]. A programmer may recognize a blueprint as a class and the use of a blueprint to create a dwarf in the environment as an instance of a class (an object). This domain-specific language is not required to convey the meaning of what is stored in libraries. The term ‘library’ seems to be adequately domain-neutral. The blueprint is a simple enough concept to understand that two buildings may be built from the same blueprint. The Aurora Toolset is more similar to Alexander’s pattern 205 STRUCTURE FOLLOWS SOCIAL SPACES [20], as it allows the user to edit existing blueprints to create objects and creatures that are unique to the story.

Possible areas of concern include keeping track of versions of resources and enforcing the distinction between a template (a class or blueprint) and an instance of the template (an object). The ‘edit properties’ screen, for example, in the Aurora Toolset for a template is almost exactly the same as the ‘edit properties’ screen for an instance of the template. There is no visual indication, once in the ‘edit properties’ screen, of whether the user is editing a template for the creature or an actual placement of a specific creature within the game world. Tracking versions of templates is left up to the user and is achieved by copying old templates and renaming them as new ones. Because characters and objects in a story change throughout the course of the story, it would be most helpful to provide a means not only store these characters and objects but to also help keep track the changes or versions.

2.3.1 TRADING STORIES

While ALL STORY TELLING IS RE-TELLING addresses the organization of story components for re-use, the sub-pattern TRADING STORIES addresses how story components are shared.

Authors of interactive narratives want to share their stories and the components of their stories with one another. Locating, importing from other computers and exporting story resources to other locations may be difficult.

Therefore: Provide a means to import and export resources, thus supporting re-use, portability and the ability to share components of stories.

This is a logical extension of ALL STORY TELLING IS RE-TELLING, populating libraries with the content of others, and is illustrated by the large communities of interactive narrative players and writers that support resource libraries on the web.

The NWN community is an example of this pattern [22], given that NWN's Aurora Toolset also supports the importing and exporting of custom objects, encounters and creatures.

Adrift supports the import and export of modules and object scripts from one adventure to another. *Wizards* control the importing of objects to ensure that the scripting code matches that of the target adventure, and offers corrections to ensure that everything works as planned.

Designers should consider including a means to keep track of the components inside the exported resource. This could take the form of a secondary notation system that is attached to an exported resource file or the ability to browse and peek inside resources. The danger of sharing resources, especially over the web, is that with any file-based sharing system: you never really know what you are going to get. The risks are mitigated through the use of anti-virus software and by the nature of the self-policing communities that support the tool.

2.4 READ IT BACK TO ME

During the design process, authors commonly check their stories for errors, read through their creations to refresh their memories or to see if it reads properly. This is done continuously throughout the creation of the story.

Reading and re-reading a story throughout creation is common and often time-consuming.

Therefore: Provide a means to run the story within the STORYTELLER'S CORNER.

Similar to the code-test cycle performed by programmers of software, authors of interactive narratives will desire to run their creations during the design process for a variety of reasons.

The most common reasons include testing (does my 'source code' actually do what I want it to do?), debugging (are there any system or syntax errors when I run it?), refreshing one's memory (where did I leave off last?), exploratory (what happens when I do this?) or simply wishing to view the latest part of their creation as a player in the story.

As seen in Figure 1, Inform 7 has a Debug menu as well as Go buttons that allow the user to readily play through the narrative created from their natural language source code. The Go catches and notifies any syntax errors, as previously discussed. The *Replay* and *Play to Here* buttons give the authors the flexibility to continue play from different points in the storyline. These are not unlike the debugging, steps, watches and testing performed in programming environments like NetBeans or Visual Studio.

The Aurora Toolset allows authors to run the module that they are working on, even if it is not complete. The unfortunate aspect of the tool is that authors must play through the entire module from the beginning. This is cumbersome when dealing with large maps and long and complicated story-lines with many quests and goals. The poor concept of global state in the NWN system is the culprit. While running a module is a necessary feature, it does not solve the time-consuming nature of the task.

3 References

- [1] *History of Infocom*, Infocom, Accessed on January 22, 2007 on the World Wide Web: <http://www.infocom-if.org/company/company.html>
- [2] Packard, Edward, and Paul Granger. (1979) *The Cave of Time*. Toronto; London ; New York ; Sharon, CT ; Englewood Cliffs, NJ : Bantam ; Grey Castle Press ; Prentice-Hall.
- [3] Szilas N. (1999) Interactive drama on the computer: beyond linear narrative, In: AAAI 1999 Fall Symposium on Narrative Intelligence.
- [4] (2005) *Façade: A One-Act Drama* Procedural Arts, Accessed on January 22, 2007 on the World Wide Web: <http://www.interactivestory.net/>
- [5] Neverwinter Nights Community Site. BioWare Corporation. Accessed December 8, 2006 on the World Wide Web: <http://nwn.bioware.com/>
- [6] The Six Standard Examples, Inform. Accessed on January 22, 2007 on the World Wide Web: <http://www.inform-fiction.org/examples/index.html>
- [7] Firth, R. Cloak of Darkness: Inform. Accessed on January 22, 2007 on the World Wide Web: <http://www.firthworks.com/roger/cloak/inform/index.html>
- [8] Nelson, G. (2006) *Natural Language, Semantics Analysis and Interactive Fiction*. St. Annes's College, Oxford. Available on the World Wide Web: <http://www.inform-fiction.org/I7Downloads/Documents/WhitePaper.pdf>
- [9] Daly, L. (2006) *Natural Language Game Programming with Inform 7*. O'Reilly ONLamp.com, LAMP: The Open Source Web Platform. Accessed on the World Wide Web: <http://www.onlamp.com/pub/a/onlamp/2006/06/08/inside-inform-7.html?page=1>
- [10] *Façade* Technology (2005-6). *Façade: a one-act interactive drama*. Accessed on January 22, 2007, on the World Wide Web: <http://www.interactivestory.net/technology/>
- [11] Mateas, M. and Stern, A. (2003) *Façade: An Experiment in Building a Fully-Realized Interactive Drama*. Game Developers Conference, Game Design track, March 2003. Accessed on January 22, 2007 on the World Wide Web: <http://www.interactivestory.net/papers/MateasSternDAC05.pdf>
- [12] Mateas, M. and Stern, A. (2005) *Structuring Content in the Façade Interactive Drama Architecture*. Artificial Intelligence and Interactive Digital Entertainment (AIIDE), Los Angeles, June 2005. Accessed on January 22, 2007 on the World Wide Web: <http://www.interactivestory.net/papers/MateasSternAIIDE05.pdf>
- [13] *d20 System*. Wizards of the Coast Inc., Renton, WA. Accessed on January 23, 2007, on the World Wide Web: <http://www.wizards.com/default.asp?x=d20/welcome>
- [14] *What is D&D?* Wizards of the Coast Inc., Renton, WA. Accessed on January 23, 2007, on the World Wide Web: <http://www.wizards.com/default.asp?x=dnd/whatisdnd>

- [15] *Forgotten Realms Campaign Setting Official Home Page*. Wizards of the Coast Inc., Renton, WA. Accessed on January 23, 2007, on the World Wide Web: <http://www.wizards.com/default.asp?x=dnd/fr/welcome>
- [16] (2003) D&D Character Sheet, *Dungeons and Dragons Player's Handbook, Core Rulebook I*, v3.5, Wizards of the Coast Inc., Renton, WA. Available for download at: <http://www.wizards.com/default.asp?x=dnd/dnd/charactersheets>
- [17] Cox, P. and Smedley, T. (2001) Experiences with Visual Programming Languages for End-Users and Specific Domains, Proc. OOPSLA Workshop on Domain-Specific Visual Languages, Univ. of Jyvaskyla, Dept. of Computer Science TR-26, Tampa Bay FL (October 2001), 87-96.
- [18] National Instruments, Inc. (2003) *Introduction to LabVIEW: A Six-Hour Course*. National Instruments Inc.
- [19] 27. *Dwarves! (lines 3045-3148)*. Browsing Advent.inf. Inform. Accessed on January 23, 2007 on the World Wide Web: http://www.inform-fiction.org/examples/Advent/Advent_2_27.html
- [20] Alexander, Christopher (1977). *A Pattern Language*. New York: Oxford University Press.
- [21] Campbell Wild (2006) ADRIFT: Adventure Development & Runner – Interactive Fiction Toolkit. Accessed on January 23, 2007, on the World Wide Web: <http://www.adrift.org.uk/cgi/new/adrift.cgi>
- [22] Neverwinter Nights Community Site. BioWare Corporation. Accessed December 8, 2006 on the World Wide Web: <http://nwn.bioware.com/>
- [23] Claire Dormann, Jean-Pierre Fiset, Sebastien Caquard, Birgit Woods, Aida Hadziomerovic, Elizabeth Whitworth, Amos Hayes, and Robert Biddle (2005) *Repurposing a computer role playing game for engaging learning*. In Proceedings of Ed-Media: World Conference on Educational Multimedia, Hypermedia, and Telecommunications, pages 4430-4435, Montreal, Canada, 2005. Association for the Advancement of Computing in Education.
- [24] An Analysis of MMOG Subscription Growth - Version 21.0. Accessed on January 23, 2007, on the World Wide Web: <http://www.mmogchart.com/>
- [25] Yee, N. (2005), Motivations of play in MMORPGs: Results from a factor analytic approach, Available online at <http://www.nickyee.com/daedalus/motivations.pdf>
- [26] Diomidis Spinellis (2001) *Notable design patterns for domain specific languages*. Journal of Systems and Software, 56(1):91–99, February 2001. Available at: <http://www.spinellis.gr/pubs/jrnl/2000-JSS-DSLPatterns/html/dslpat.html>

- [27] *End-user development: tools that empower users to create their own software solutions* (2004) Source Communications of the ACM archive Volume 47, Issue 9, SPECIAL ISSUE: End-user development, ACM Press, New York, NY, USA
- [28] *Domain-specific languages: an annotated bibliography* (2000) ACM SIGPLAN Notices archive, Volume 35, Issue 6, June 2000, ACM Press New York, NY, USA
- [29] Pippin Barr, James Noble, and Robert Biddle (2006) *Videogame values: Human-computer interaction and games*. *Interacting with Computers*, 19(2), 180-195.
- [30] Pratchett, R. (2005) *Gamers in the UK: Digital Play: Digital Lifestyles*. BBC.
- [31] Steiner, K.E. and Moher, T.G. (1992) *Graphic StoryWriter: an interactive environment for emergent storytelling*. CHI '92: Proceedings of the SIGCHI conference in Human factors in computing systems. ACM Press: New York, NY, USA, pp. 357-364.
- [32] Schafer, L., Valle, C. and Prinz, W. (2004) Group storytelling for team awareness and entertainment. NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction. ACM Press: New York, NY, USA, pp. 441-444.