

Using Social Agents to Visualize Software Scenarios

Thomas A. Alspaugh, Bill Tomlinson, and Eric Baumer*
Department of Informatics
Donald Bren School of Information and Computer Sciences
University of California, Irvine

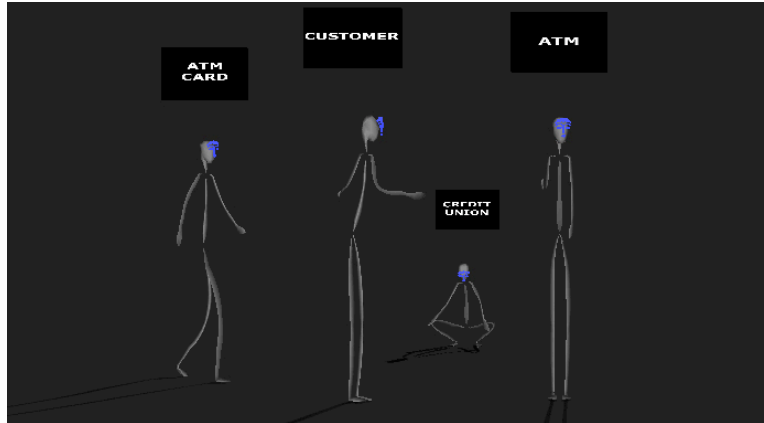


Figure 1: Visual model of an ATM scenario

Abstract

Enabling nonexperts to understand a software system and the scenarios of usage of that system can be challenging. Visually modeling a collection of scenarios as social interactions can provide quicker and more intuitive understanding of the system described by those scenarios. This project combines a scenario language with formal structure and automated tool support (ScenarioML) and an interactive graphical game engine featuring social autonomous characters and text-to-speech capabilities. We map scenarios to social interactions by assigning a character to each actor and entity in the scenarios, and animate the interactions among these as social interactions among the corresponding characters. The social interactions can help bring out these important aspects: interactions of multiple agents, pattern and timing of interactions, non-local inconsistencies within and among scenarios, and gaps and missing information in the scenario collection. An exploratory study of this modeling's effectiveness is presented.

CR Categories: D.2.1 [Requirements]: Languages and Tools

Keywords: scenario analysis, ScenarioML, social autonomous characters, interactive animation

1 Introduction

*e-mail: {alspaugh,wmt,ebaumer}@ics.uci.edu

Scenarios and use cases are widely used in a number of ways during the development process, and by a variety of participants [Alexander 2004; Benner et al. 1992]. These participants frequently include stakeholders and users who are not experts in the use and analysis of scenarios. It is essential that these nonexperts be able to understand the scenarios of usage that describe a software system. Scenarios are an effective communication medium between stakeholders and developers. Their narrative form and use of natural language take advantage of people's natural ability to understand stories. However, scenarios also involve challenges, especially when more than one scenario must be considered at once. Other researchers have noted that people are better at identifying errors of commission than errors of omission: they are more successful at finding individual statements that are incorrect, than at identifying missing information, non-local inconsistencies between two or more scenarios, unstated assumptions about the world or the system, or ambiguous or ill-defined terminology. Visually modeling a collection of scenarios as social interactions (see Figure 1) provides an additional path to achieve better understanding of the systems described by those scenarios. If the modelling preserves the interconnections and dependencies of the scenarios, it can take advantage of people's basic competences with social interactions to give viewers insight into the interactions described by the scenarios. The research presented here uses the metaphor of social interaction to improve nonexperts' comprehension of sets of scenarios.

This project combines a scenario language with structure, semantics, and automated tool support (ScenarioML) with an interactive graphical game engine featuring social autonomous characters and text-to-speech capabilities. The animated social interactions of the autonomous characters and the text which they speak is driven by the collection of scenarios used as input to the visualization. Visualization of another collection of scenarios is accomplished simply by giving the new scenarios as input to the visualization software.

Other researchers have developed a 3D representation for visualizing large software systems and representing high dimensional data [Marcus et al. 2003]. This research demonstrates the value that may

be gained by utilizing the human ability to process spatial information.

Related work in requirements engineering has found that rich media enhance the effectiveness of scenarios for walkthrough, analysis, and elicitation of further requirements. However, the cost of producing the rich media to do this is substantial, and of course it is difficult to produce rich media for systems that do not yet exist. Our approach produces a visualization of a scenario that is produced directly from the scenario itself, a far less costly approach. And as each scenario evolves, its visualization evolves with it with minimal cost and effort.

Our mapping from scenarios to animated social interactions results in an animated character for each actor, agent, and entity desired to be shown in the visualization. Each character is accompanied by an identifying label, and actors and agents speak their actions as a means of expressing the accomplishment of those actions. Interacting characters move to face each other during their interaction. The social interactions thus modeled brings out the patterns of interaction between actors, agents, and entities, and the temporal pattern in which actions occur. Inconsistencies in those patterns within and between scenarios are perceived as inconsistent social interactions between the corresponding characters. Presenting these patterns in a social form provides an additional means of identifying missing information, unstated assumptions, and ill-defined terms by mapping them into a social space in which they may be more visible than in their original narrative form.

A video illustrating this work may be found at <http://orchid.calit2.uci.edu/~wmt/movies/softvis.mov>.

The remainder of this paper is organized as follows. Section 2 discusses the related work that provides a context for this research. Section 3 describes our implementation of the scenario-driven, 3D animated social interaction visualization. We present an evaluation of its effectiveness in Section 4. Section 5 discusses results, insights, and lessons learned. We outline some future work in Section 6, and summarize our conclusions in Section 7.

2 Related Work

Over the past decade, several researchers have investigated the use of images and multimedia to enhance scenarios.

Wood *et al.*'s AMORE tool enhanced requirements with multimedia captured during elicitation, primarily for better recordkeeping and traceability [Wood *et al.* 1994]. The recorded interviews provide a richer record than the derived requirements, and were also envisioned as a means of teaching and improving requirements elicitation skills.

Haumer *et al.*'s CREWS-EVE used rich media representations of scenarios showing uses of an existing system, for the purpose of recording more complete requirements information and providing a better basis for requirements evolution [Haumer *et al.* 1998]. The rich media were used to achieve better stakeholder involvement, complementing the strengths of scenarios and enhancing the requirements baseline. The rich media showed concrete system uses, and supported better explanation of the requirements, more effective understanding of variant ways of using a system, better presentation of multiple viewpoints, and structured review of current requirements.

The ART-SCENE tool of Maiden *et al.* uses rich media to complement scenarios [Zachos and Maiden 2004; Zachos *et al.* 2005]. Maiden points out that stakeholders are better at identifying errors

of commission than errors of omission. The enhanced context that rich media can provide gives more cues than text scenarios, and is more likely to elicit information that would remain tacit with text-only scenarios. A recent study showed that rich-media scenarios in ART-SCENE resulted in discovery of a larger number of alternative courses, though not a larger number of resulting requirements [Pang *et al.* 2005]. The study also showed that construction of rich media for enhancing scenarios was an expensive process, and was not necessarily cost-effective.

The visualization of software systems has been shown to enable computer scientists to work with greater comprehension, improved accuracy and increased speed [Tudoreanu 2003]. In particular, visualizations are helpful to enable novice users in fault localization tasks [Ruthruff *et al.* 2003].

The implementation of this project builds on previous research in the design of interactive virtual worlds and autonomous animated agents. These topics have been explored extensively by the commercial sector (e.g., the computer game industry), by academic researchers (e.g., the Autonomous Agents and Multi-Agent Systems conference), and by governmental organizations (e.g., the US military's Simulation, Training and Instrumentation Command (STRICOM)). While a full review of this literature is not possible in the confines of this paper, there are several projects that are most relevant to this work, and exemplary of other work in these fields.

This project has been programmed using a graphics system based on JOGL, the Java implementation of OpenGL. This graphics system uses an adaptation of a non-photorealistic renderer created for the AlphaWolf project [Tomlinson *et al.* 2002] and employs graphical effects developed for the Virtual Raft project [Tomlinson *et al.* 2005]. The animations of the autonomous agents are enabled by a motor system that blends example animations in real time [Downie 2001]. The autonomous behavior of the agents is built on a framework inspired by the work of Blumberg [Blumberg 1996] and Perlin [Perlin and Goldberg 1996].

Conceptually, the project draws on the idea of the Memory Palace, a mnemonic device that has been employed for millennia by orators such as Cicero. This method for remembering complex information involves making a connection between pieces of the information and different physical locations. These pieces of information may then be retrieved in order by mentally "walking" through the space, and remembering each piece of information by its connection to that place. This method for memorization draws on the superior ability of humans to remember the physical configurations of spaces, which exceeds our ability for rote memorization. In his keynote address at the 2006 Game Developers Conference, Philip Rosedale, the CEO of Linden Labs, proposed that the popular "Second Life" virtual world that his company created could be used as a memory tool [Rosedale 2006]. The project described in this paper draws on this same concept — the using of a real-world metaphor to connect potentially abstract concepts to mental structures that are easier for people to manipulate. The research described here extends the idea of the Memory Palace to include both physical and social elements. Social interactions are one of the few topics that people can remember as well as they remember physical configurations [Byrne and Whiten 1988]. In addition, the anthropomorphization of abstract concepts has been common across human history (e.g., the Everyman morality play of the 15th century, in which Good Deeds and Knowledge are personified characters [Anonymous 1998]). By using both a physical and a social metaphor, this "Memory World" can be a useful tool for remembering, comparing and mentally operating on a complex body of information such as a collection of scenarios.

```

<scenarioCollection name="ATM_SOFTVIS">
  <title>ATM Scenarios</title>
  <avp attr="author">Thomas A. Alspaugh</avp>
  <scenario>
    <title>Fast Cash</title>
    <eventChain>
      <simpleEvent>
        <text>
          The
          <ref to="ATM">
            <avp attr="EntityName">ATM</avp>
            <avp attr="ality">real</avp>
            <avp attr="stationary">true</avp>
            <avp attr="persistent">true</avp>
            <text>ATM</text>
          </ref>
          displays a "welcome" screen: "Insert card to begin".
        </text>
      </simpleEvent>
      <simpleEvent>
        <text>
          A
          <instance name="customer">
            <avp attr="EntityName">customer</avp>
            <avp attr="ality">real</avp>
            <avp attr="stationary">false</avp>
            <avp attr="persistent">true</avp>
            <text>customer</text>
          </instance>
          inserts an
        </text>
      </simpleEvent>
    </eventChain>
  </scenario>
</scenarioCollection>

```

Figure 2: ScenarioML scenario (excerpt)

3 Implementation

In the pursuit of visualizing software systems using the metaphor of social interaction, this paper presents a system that visualizes a collection of scenarios as interactions between autonomous animated agents. The system consists of several main components: the ScenarioML encoding of a scenario collection; an interface menu that lets the user select which scenario should be enacted by the animated agents; a real-time 3D graphical world inhabited by autonomous animated agents who enact the scenarios; a scenario manager that keeps track of what event is currently being enacted in which scenario.

3.1 ScenarioML

ScenarioML [Alspaugh 2006] is an XML language designed for expressing scenarios and making use of them. ScenarioML provides flexibility in determining the specificity with which a system is described, the individual properties used to describe the system, and structure of the events occurring in the system. It allows a designer to specify scenarios with widely varying degrees of complexity; a scenario could contain a single simple event, a chain of events to occur in sequence, a partial temporal order of events whose occurrence is constrained by the partial order, or any combination of the above. The events are further extended by lists of possible alternatives one of which may occur, iteration of an event occurring repeatedly under constraints, episodes reusing the events of another scenario, and other constructs. ScenarioML is designed to map onto the common human-readable scenario forms, and at the same time provide an undergirding of formal structure for automation and tool support. For the purposes of this initial investigation, scenario collections were used that contained chains of events to be carried out in the specified order.

3.2 Graphical 3D World

The virtual world used to visualize the scenarios in this project builds on previous research in the areas of interactive multimedia installations, synthetic characters, and autonomous social entities [Tomlinson et al. 2001; Tomlinson et al. 2005]. The graphics system was written entirely in Java and renders its graphics using JOGL, the Java implementation of OpenGL. This graphical world implements a background and ground plane against which characters interact, a virtual camera system, and a lighting model. Here, the world is populated by a set of autonomous animated agents, each of which represents an entity in the scenario collection being visualized. These animated agents cast shadows on the ground plane, which add to the three dimensionality of the visualization.

3.3 Animated Autonomous Agents

The animated autonomous agents in the virtual world are modeled and rigged (i.e., they had their geometric meshes attached to articulated skeletons) in a 3D animation program called 3D Studio Max. These virtual characters are designed to play animations in real time through Java. Each character has a motor system that seamlessly connects each animation and is also able to blend between several example animations. These example animations were created in 3D Studio Max and then exported in a raw data format so that they could be manipulated by the Java code.

The characters represent entities in the scenario collection being visualized. Each character is also accompanied by a nametag, which floats above its head and displays the name of the entity the character represents. Also, they are able to produce audible spoken utterances using the FreeTTS text-to-speech system (<http://freetts.sourceforge.net/>). Since the “expressive resolution” of a character’s animation (i.e. how much viewers can tell about the character’s role in the scenario) from watching its animation may not be good enough to express all of the elements in a scenario, the name tags and speech capabilities can augment their animated behavior with specifics such as numbers, names and other more precise information.

Entities in this system can have a number of properties. First, entities can either be physical, such as an ATM or a bank, or virtual, such as a query the ATM might send to the bank. This distinction is made visible by the color of the crown on the entity’s head; a blue crown is used for physical entities and a green for virtual (see Figure 3).

The visualization also includes a blue sign that reads “Physical Entities,” and a green sign that reads “Virtual Entities,” to remind the user which color is which. Green was chosen for virtual entities because of the popularity of The Matrix movies over the past several years, in which green text on a black background represents a data world. Blue was chosen to represent real entities because it was one of the two other primary colors of light, and was deemed to have fewer strong connotations than red, which is inextricably tied to ideas such as “stop,” “blood,” etc. Entities can also have an owner, another entity the first entity will follow around. For example, an ATM card is owned by a customer, and thus when the customer approaches the ATM, its ATM card automatically goes with it. In early prototypes of this system, it became clear that many scenarios relied on the reader having pre-existing knowledge of relationships such as the ownership of an ATM card by a customer. While it may be viable to rely on an audience having such knowledge in a common interaction such as an ATM withdrawal, especially when dealing with a US audience, there are many other collections of scenarios where the reader will not have this kind of pre-existing

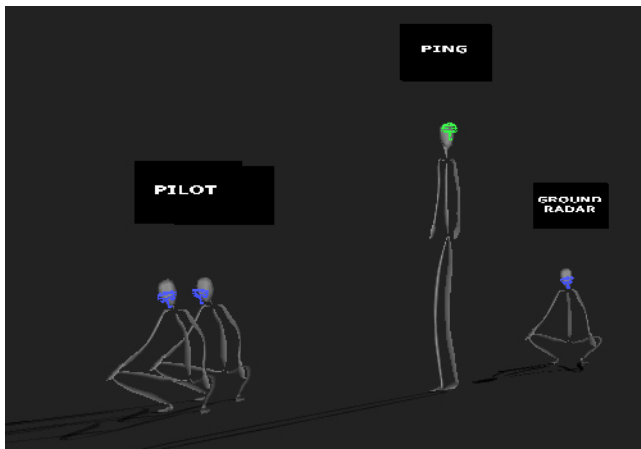


Figure 3: Visualization of a TIS event. Actual entities such as the pilot, TIS (partially obscured), and ground radar have blue crowns, and virtual entities such as a ping carrying data are crowned in green.

expertise. In these contexts, it is critical that the scenarios are explicit in expressing all of the relationships among the entities.

An entity can also be stationary or mobile; a customer is mobile, while an ATM is stationary. Stationary entities are prohibited from walking around in the graphical world. This information may also be obvious to many observers in certain collections of scenarios — for example, many people will understand that banks should not walk around. However, as with the ownership issue mentioned above, it is important for scenarios to represent explicitly information such as the mobility of the entity so that observers may have a solid understanding of the interactions that are occurring. This explicitness is particularly important in situations where many observers do not have pre-existing expertise with the content domain.

Lastly, some entities are persistent, while others are transient. For example, the ATM, customer, ATM card, and bank are all persistent, and thus present during all events in the scenario collection. However, entities such as queries from the ATM and responses to those queries from the bank are transient. When the visualization of a scenario reaches an event that involves one of these entities, the entity appears. It remains present until it is no longer part of the scenario, then disappears. For example, a scenario collection about a traffic information system (TIS) in airplane cockpits, the pilot, the cockpit's TIS, and the ground radar are all persistent. Entities such as other planes that may enter the same airspace or pings sent between the ground radar and TIS are transient. When the ground radar sends out a ping, a virtual entity representing the ping appears next to the ground radar. When the TIS receives the ping, the ping moves from the radar to the TIS. Once the TIS has received and processed the ping, the character representing the ping disappears. This use of transient entities helps to reduce cognitive load on the user. Rather than having to keep track of as many as a dozen entities that may all take part in a scenario collection, the user can focus on just the entities participating in the current event of the scenario being enacted.

To describe the above characteristics in pseudocode:

```
Entity {
boolean physical;
    //true is physical, false is virtual
Entity owner;
    //who, if anyone, owns this entity and
```

```
    //determines where its "home position" is
boolean mobile;
    //true is mobile, false is stationary
boolean persistent;
    //true is persistent, false is transient
}
```

It is important to reiterate that when a human reads a scenario, many of these properties that an entity may have are included in the reader's common sense knowledge about the systems in question. Many people know that an ATM is stationary. However, the presence of this background experience becomes less reliable when describing complex systems to novices and non-experts. It might not be common knowledge to every reader that a ground radar system is actually on the ground, rather than on the plane, and is thus stationary. In this way, the visualization described here can help readers unfamiliar with the topical matter of a scenario to understand basic attributes about the various entities involved.

Also, the matter of entities and properties points to a strength of ScenarioML. There is nothing about a simple, basic encoding of a scenario collection into ScenarioML that would make entities be explicitly noted as entities. Rather, as an XML language, ScenarioML provides the tools to extend the language and create whatever tags are necessary. Using ScenarioML's attribute-value pair (AVP), a new attribute, EntityName, was created that signified which elements of the scenario should be considered entities. Furthermore, AVPs were also used to encode all the entity's properties. This has the negative effect of requiring a little more work in writing the scenarios themselves. However, the benefit in making the scenarios more readily comprehensible to all stakeholders should outweigh the extra time required. Furthermore, scenario collections with such enhancements are still perfectly valid in the ScenarioML schema, and thus these enhancements do not prevent the scenario collection from being used in a way any other scenario collection could.

The core behavior of the autonomous agents includes two core skills — navigation and expression. These two skills are triggered by the scenario manager (see below). When combined with the scenario manager's ability to generate spoken utterances, these skills provide a sufficient range of behavior for a wide variety of scenarios, while remaining simple enough to be clear to both the scenario designers and the scenario's end users.

The goTo skill enables a mobile agent to navigate to an arbitrary position and orientation, and enables a stationary agent to orient to an arbitrary target. This process involves finding the direction vector from the character's position to the target, and then dynamically blending animations with different degrees of turning so that the character exhibits a walk or turn behavior with an appropriate degree of turning to bring it into the correct physical relationship with the target. Apparent physical relationships are critical to the perception of interactions among people, devices or software systems; while two computers can network across the world nearly as easily as they can network to a device in the same room, people are much more closely tied to the physical configurations of entities for those entities to be seen as social.

When an agent navigates to a new position, any agents that are owned by that agent also follow along until they are within a suitable distance from their owner. Through this mechanism, groups of entities may be caused to travel in groups through the visualization space.

Once an agent has arrived at the correct position and orientation, it will perform the expressive behavior dictated by scenario manager. This behavior usually involves gesturing with the hands and upper body while standing, whereas the default pose for agents is

squatting down. This difference in primary pose helps to direct an observer's to the action currently taking place in the enactment of the scenario — characters that are standing, moving and/or gesturing are the key actors at any given moment of the scenario.

3.4 Scenario Manager

The scenario manager is similar to the director of a play, or the drama manager in an interactive narrative [Mateas 1999]. It must determine which entities are involved in the current event, keep track of the visualization's progress through a scenario, determine when the current event has been completed, select the next event, and determine when the entirety of the scenario has been enacted.

Currently, the scenario manager handles only scenarios consisting of single events or chains of successive events. In this way, the scenario manager is not required to determine which path to follow in a graph of scenarios, but rather goes through the scenarios in the order specified. When an event begins, the scenario manager finds all the entities that are participating in that scenario. If any of those entities are transient, they appear when the event begins. The average position of all the entities involved is calculated, and those entities go there. When the entities arrive at that location, they gesticulate as if conversing, representing that these entities are currently interacting while the other entities are not currently active. If the scenario involves a stationary entity, the other entities approach the stationary one. This visualization prohibits two stationary entities from participating in the same scenario. In an event like an ATM sending a query to the bank, the sending of the query is one event, and the bank receiving it is a separate event. Not only does this help to make the scenario clearer, but it provides a failure mode if there is a problem with the scenario in that it requires two entities that cannot move to interact, such as requiring the ATM to interact directly with the bank.

In addition to coordinating the animation of the agents enacting the selected scenario, the scenario manager also vocalizes the textual content of the scenario using the Java Speech API. While the animated agents have nametags so as to denote which entities are interacting with which, the "expressive resolution" of the character's animation (i.e. how much viewers can tell about the character's role in the scenario) is intentionally low, so that the same animation can be used simply to indicate that an entity is participating in an event. The scenario manager's vocalization of the textual content of the scenario makes use of multiple modalities to provide the user with information about the scenario. According to the cognitive theory of multimedia learning [Mayer 2001], a multimedia system should not simultaneously present the user with text and animation, because trying to watch both becomes too much cognitive load for that single modality. By employing audio, the system can use an additional modality to convey additional information without cognitively overloading users.

The scenario manager determines that an event has ended when all the entities involved in that event have reached the destination specified at the beginning of the event and the speech engine has completed reading the text of the event. When this occurs, the scenario manager proceeds to the next event in the scenario. If there are no more events in the scenario, the speech engine is used to notify the user that the selected scenario has been completed. The user can choose to leave all the entities where they ended up at the end of the scenario, or he or she can reset all the entities to their initial positions.

ATM Scenarios

Contents

1. "Fast Cash"
2. "Balance and Withdrawal"
3. "Stand-in Fast Cash"

1. "Fast Cash"

EVENT CHAIN:

1. The [ATM](#) displays a "welcome" screen: "Insert card to begin".
2. A [customer](#) inserts an [ATM card](#) into the [ATM](#).
3. The [ATM](#) displays "Please select your language preference", with choices "English" and "Spanish".
4. The [customer](#) selects "English".
5. The [ATM](#) displays "Please enter PIN", with choices "Please press cancel if error" and "Press if correct".
6. The [customer](#) enters a [PIN](#) and chooses "Press if correct".
7. The [ATM](#) displays "Please select a transaction. Please press cancel if error. Transfer, Deposit, Payment, Cash Check, Fast Cash From Checking, Withdrawal, Balance Inquiry."
8. The [customer](#) selects "Fast Cash From Checking".
9. The [ATM](#) sends a query "[Checking - \\$160 OK?](#)" to the [credit union](#).
10. The [credit union](#) receives the query "[Checking - \\$160 OK?](#)".
11. The [credit union](#) sends the response "[Checking - \\$160 OK](#)" to the [ATM](#).
12. The [ATM](#) receives the response "[Checking - \\$100 OK](#)".
13. The [ATM](#) dispenses [\\$160](#).
14. The [customer](#) takes the [\\$160](#).
15. The [ATM](#) displays "Please take your card / Thank you", and ejects the [ATM card](#) halfway.

Figure 4: Scenario in printed form

3.5 Human-Computer Interface

The interface for this visualization consists of a Macromedia Flash animation running in a web browser, which displays a menu of the scenarios in the current collection. The flash animation communicates with the Java program running the visualization using a socket connection. When the flash animation starts up, it contacts the visualization to get the names of the scenarios, which it then uses to populate the menu it presents to the user. When the user clicks on one of the items in the menu, the animation sends the name of the item clicked to the visualization. The characters then begin to enact that scenario.

4 Evaluation

We hypothesized that our visualization would be of benefit to non-experts trying to understand a collection of scenarios. We evaluated the visualization by presenting two small scenario collections in two different ways to two groups of sophomore and junior computer science students at the University of California, Irvine. The 22 students were appropriate subjects because they themselves are non-experts in understanding and analyzing collections of scenarios, and the systems represented by them. One of the scenario collections represented a familiar system and context, an Automated Teller Machine (ATM); the other system represented a system and context with which nearly all the students were unfamiliar, the Traffic Information System (TIS) used in some private planes to help avoid mid-air collisions. Each collection consisted of three scenarios, described in approximately 800 words; the ATM collection had about 60 events and the TIS collection about 40. Both collections were derived from observations of actual uses of the systems in question, which were ATMs of the first author's credit union and the Bendix/King KMD250 Multi-Function Display/GPS.

The two scenario collections were presented to the first group of eleven subjects in printed form only, and to the second group of eleven subjects in printed form accompanied by an animated social interaction visualization. This portion of the experiment lasted 30 minutes, to give each group time to work through the scenarios and/or view all of the visualizations. The two groups then switched tasks, with the first group viewing both printed form and visualization, and the second group the printed form only. Both groups were then given the same fixed amount of time (30 minutes) to examine and analyze both scenario collections. Thus each scenario collection was evaluated twice by both groups but in different presentations.

The visualizations were presented twice to each group, in the sequence ATM ATM TIS TIS. The presentation took almost the entire 30 minutes.

Each collection was seeded with approximately twenty faults. These faults covered a range of common types of scenario faults, including the types we expected our social visualization to be especially effective with.

Gap Some behavior or information is not given, but is needed to understand the system.

Local inconsistency The scenarios describe nearby events that are apparently contradictory.

Nonlocal inconsistency The scenarios describe events that are apparently contradictory, in two different scenarios or separated by many intervening events in the same scenario.

External inconsistency The scenarios describe unrealistic events that could not happen in the physical world, or that conflict with the constraints of the system's environment.

Undefined item A scenario uses a term, actor, entity, action, or other item that is not defined and unlikely to be familiar to the readers.

Ambiguously-defined item A term, actor, entity, action or other item is familiar to the readers in one sense but is used differently in the scenarios.

The subjects were also given questionnaires to fill out during the 30-minute period. The questionnaire consisted of these questions:

1. What problems did you find, and when (what minute) did you find each one? (*with sufficient space for 30 items.*)
2. Have you ever used a system like the one the scenarios describe (ATM or TIS)?
3. What other comments or suggestions do you have about the scenarios, the way they were presented, or this study?

At the end of the 30 minutes, the two groups were given fresh copies of the questionnaire and they swapped places, so that the first (print-only) group now saw the animated social interaction visualization, and the second (print plus visualization) group was given time to read and analyze the printed scenarios. Each group was again given 30 minutes.

The questionnaires were analyzed by classifying each noted problem, if possible, as an identification of a specific fault. These faults included the intentionally seeded faults and a small number of additional "authentic" faults that the subjects identified. The problems were then organized by the context in which they were noted:

	ATM	TIS
P1	43	23
PV1	37	21
P2	39	27
PV2	10	5
P1+PV2	53	28
PV1+P2	76	48

Table 1: Summary of study results

PV1 By a subject working with the printed scenarios and animated visualization, during the first 30 minutes.

P1 By a subject working with the printed scenarios only, during the first 30 minutes.

P2 By a subject working with the printed scenarios only, during the second 30 minutes; all these subjects had been in PV1 earlier.

PV2 By a subject working with the printed scenarios and animated visualization, during the second 30 minutes; all these subjects had been in P1 earlier.

We hypothesized that an initial analysis would show the effectiveness of the visualization with respect to the print presentation for problem identification and overall comprehension, and that this effectiveness would vary for different kinds of problems. We also hypothesized that a further analysis in more detail (not yet completed) would show the extent to which the two forms favored identification of each individual problem. Our expectation was that some problems would be more easily identified in one form or the other, and that this would be confirmed by cases in which a subject initially did not identify that problem using one form (print, or print plus visualization) but then in the second period identified it using the other form. We chose to give the subjects the printed scenarios to refer to as they viewed the visualization, because we thought it possible that many subjects would not effectively grasp the details of the visualization without a printed copy to refer to occasionally.

The overall results of this initial study are summarized in Table 1. The figures in the table sum the numbers of problems found by each student. The data are from 10 students in each group; the data for one student in each group was found to be unusable for this study.

Subjects working first with this prototype of the visualization identified somewhat fewer problems than those working first from the printed scenarios alone, both for the familiar system (ATM, 14% fewer) and the unfamiliar system (TIS, 9% fewer). It was interesting that the subjects who worked first with the visualization and printed scenarios (PV1), and second with the printed scenarios alone (P2), identified a substantial number of additional problems in the second half-hour (P2). This was in contrast to the subjects who worked first with the printed scenarios (P1) and then also with the visualization (PV2); these subjects found relatively few additional problems. This suggests that the visualization may significantly augment the effectiveness of the printed scenarios. A further study will be necessary comparing PV1+P2 to an equivalent time spent only on the printed scenarios.

5 Discussion

One demonstration of the value of this system lies in the fact that, upon viewing the visualization of the scenarios that they themselves had written, the designers of the system found numerous errors in those scenarios. In addition, the visualization made it apparent how to correct those errors. For example, when the ATM customer

walked to the ATM and the ATM card stayed back at its initial position, it pointed out very clearly the failure mode of the user forgetting his/her ATM card at home.

A similar small epiphany occurred when one of the designers was giving the entities their animations, so that any entities participating in a given event gesticulate while the others just squat. At the point in the scenario where the customer enters a PIN, he noticed that the PIN entity was gesturing, but not the customer entity. Thinking he must have coded something wrong, he opened up the XML file to double check. Sure enough, in that particular event, the PIN had been tagged as an entity, but not the customer. The visualization made this error explicit in a way that would have been easy to miss in a text-only format.

While multimodality may not be optimal for all technological systems, there are certain elements of scenarios that might be most effectively understood using one or another of the conventional modalities through which people interact with computational systems. Some scenario issues might be most readily observed in a visual format, others in an auditory format, still others in a text-based format. By enabling observers to examine scenarios in a medium with both audible text and visible movement, certain issues with a collection of scenarios may come to light that might otherwise remain largely hidden.

In the pilot study, it became clear that revisions would be needed to several elements of the implementation of the visualization. For example, the green and blue crowns of the agents appeared to be too subtle for most of the participants to perceive. In a future version, the entire character's body will be blue or green. Similarly, the text-to-speech capabilities were a bit too difficult to understand on a casual viewing than would be optimal. In addition the fonts on the nametags were too small to be viewed from a distance. For this tool to be most useful to novices and non-experts, and also for it to be useful to groups of users, the identities and characteristics of the animated agents need to be exceedingly clear, lest people miss critical information.

6 Future Work

This project has several clear directions for future work. Specifically, future versions of the project will be created to visualize collections of more complex scenarios, such as eventDAGs (directed acyclic graphs of events) and multiple scenarios being enacted simultaneously. In addition, these projects will involve the incorporation of temporal constraints on when events can occur.

With regard to the visualization, future versions will involve a more thorough treatment of transient entities. Transient entities should be seen coming and going from the scenario, so as not to break the social metaphor. Real social entities do not spontaneously appear and disappear in the middle of an interaction, but sometimes people do come and go during the course of an interaction.

There will also be an effort to enable the bottom up organization of entities, scenarios and collections of scenarios. This organizational process will allow the individual entities to determine what scenarios they know about, watch other entities interacting, try to determine what scenario is being followed, and thereby improve their performance in various scenarios.

There will also be a continuation of the process to evaluate the efficacy of this system. The study described in this paper was valuable in framing the core questions to be asked of a system such as this one; future efforts will involve an effort to answer these questions in a rigorous and statistically significant way.

7 Conclusions

Scenarios are a useful tool for specifying software systems, especially for novices and non-experts. Collections of scenarios often have many complex interdependencies, which may make the collections difficult to understand. However, people's basic competences with everyday interactions can be leveraged to aid in the understanding of these complex systems. This research uses the metaphor of social interaction to improve people's comprehension of sets of scenarios.

The system presented in this paper uses animated autonomous agents to visualize the interactions between entities in a collection of scenarios. The system presents the user with a collection of scenarios from which the user can choose one for visualization. The animated agents then visually enact the scenario and vocalize its textual components.

The scenarios for this system are encoded using ScenarioML, an XML language that provides formal structure to textual scenarios. This tool enables designers to make explicit the most relevant interactions, relationships, and interdependencies in the scenarios. Furthermore, this formal structure allows a computational system to read, process, and enact the scenarios.

The system described here was demonstrated to a group of undergraduate students as part of a pilot study to examine ways to test the system's efficacy in showing the interdependencies within collections of scenarios. This pilot study brought to light a number of elements that are critical to the success of a social visualization - legibility, audibility and clear agent interactions. Based on this pilot study, future studies are being planned to determine the degree to which social software visualization can harness people's innate ability to understand social systems, helping them comprehend other complex phenomena such as the operation of current software systems.

With the abundance of computational power now available, it may not be unreasonable to devote significant computational resources to the creation of expressive visualizations that help to engage observers with the usage scenarios of software systems. Social visualizations such as the one described here may be able to add value to text presentation of scenarios, granting novices and non-experts the ability to understand complex software systems more efficiently and more deeply.

Acknowledgements

The authors would like to thank the California Institute for Telecommunications and Information Technology; Suzanne Shaefer; and Zack (Gang) Ji, Man Lok Yau, Andrew Correa, Uel McMahon, and the other members of the Social Code Group at UC Irvine.

References

- ALEXANDER, I. 2004. Introduction: Scenarios in system development. In *Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle*, I. F. Alexander and N. Maiden, Eds. John Wiley & Sons, Ltd., 3–24.
- ALSPAUGH, T. A., 2006. Scenarios, business rules, and matching. Submitted to *International Requirements Engineering Conference (RE'06)*.

- ANONYMOUS. 1998. Everyman. In *Internet Medieval Source Book*, P. Halsall, Ed. <http://www.fordham.edu/HALSALL/basis/everyman.html>.
- BENNER, K., FEATHER, M. S., JOHNSON, W. L., AND ZORMAN, L. 1992. Utilizing scenarios in the software development process. In *IFIP Working Group 8.1 Working Conference on Information Systems Development Processes*.
- BLUMBERG, B. 1996. *Old Tricks, New Dogs: Ethology and Interactive Creatures*. Media Laboratory, MIT, Cambridge, MA.
- BYRNE, R., AND WHITEN, A., Eds. 1988. *Machiavellian Intelligence*. Clarendon Press, Oxford.
- DOWNIE, M. 2001. *behavior, animation, music: the music and movement of synthetic characters*. Media Arts & Sciences, MIT, Cambridge, MA.
- HAUMER, P., POHL, K., AND WEIDENHAUPT, K. 1998. Requirements elicitation and validation with real world scenes. *IEEE Transactions on Software Engineering* 24, 12 (Dec.), 1036–1054.
- MARCUS, A., FENG, L., AND MALETIC, J. I. 2003. 3d representations for software visualization. In *SoftVis '03: Proceedings of the 2003 ACM symposium on Software visualization*, ACM Press, New York, NY, USA, 27–ff.
- MATEAS, M. 1999. An oz-centric review of interactive drama and believable agents. In *Artificial Intelligence Today: Recent Trends and Developments*, M. Wooldridge and M. Veloso, Eds., vol. 1600 of *Lecture Notes in AI*. 297–328.
- MAYER, R. E. 2001. *Multimedia Learning*. Cambridge University Press.
- PANG, T., MAIDEN, N., ZACHOS, K., AND NCUBE, C. 2005. Do rich media scenarios support requirements discovery? In *11th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'05)*, 152–166.
- PERLIN, K., AND GOLDBERG, A. 1996. Improv: A system for scripting interactive actors in virtual worlds. In *SIGGRAPH*, 205–216.
- ROSEDALE, P., 2006. Serious Games Summit keynote: You can (not) be serious. http://www.gamasutra.com/features/20060320/carless_01.shtml.
- RUTHRUFF, J., CRESWICK, E., BURNETT, M., COOK, C., PRABHAKARARAO, S., M. FISHER, I., AND MAIN, M. 2003. End-user software visualizations for fault localization. In *SoftVis '03: Proceedings of the 2003 ACM symposium on Software visualization*, 123–132.
- TOMLINSON, B., ET AL. 2001. AlphaWolf. In *Proceedings of SIGGRAPH 2001: conference abstracts and applications*.
- TOMLINSON, B., DOWNIE, M., BERLIN, M., GRAY, J., LYONS, D., COCHRAN, J., AND BLUMBERG, B. 2002. Leashing the AlphaWolves: Mixing user direction with autonomous emotion in a pack of semi-autonomous virtual characters. In *Proceedings of the 2002 ACM SIGGRAPH Symposium on Computer Animation (SCA-02)*, ACM Press, New York, S. N. Spencer, Ed., 7–14.
- TOMLINSON, B., YAU, M. L., O'CONNELL, J., WILLIAMS, K., AND YAMAOKA, S. 2005. The virtual raft project: a mobile interface for interacting with communities of autonomous characters. In *CHI Extended Abstracts*, 1150–1151.
- TUDOREANU, M. E. 2003. Designing effective program visualization tools for reducing user's cognitive effort. In *SoftVis '03: Proceedings of the 2003 ACM symposium on Software visualization*, 105–ff.
- WOOD, D. P., CHRISTEL, M. G., AND STEVENS, S. M. 1994. A multimedia approach to requirements capture and modeling. In *First International Conference on Requirements Engineering (ICRE'94)*, 53–56.
- ZACHOS, K., AND MAIDEN, N. 2004. ART-SCENE: Enhancing scenario walkthroughs with multi-media scenarios. In *12th IEEE International Requirements Engineering Conference (RE'04)*, 360–361.
- ZACHOS, K., MAIDEN, N., AND TOSAR, A. 2005. Rich-media scenarios for discovering requirements. *IEEE Software* 22, 5 (Sept./Oct.), 89–97.