A Community Learns Design: Towards a Pattern Language for Novice Visual Programmers

Tracy Lewis, Mary Beth Rosson, John Carroll, Cheryl Seals
Virginia Tech Center for Human-Computer Interaction, Blacksburg, VA 24060 USA
{tracyl, rosson, carroll, cseals}@ vt.edu

Abstract

We conducted a one-day design workshop in which residents of a community collaborated in learning about and designing community-related visual simulations (to be implemented in Stagecast Creator). An analysis of their design ideas and concerns revealed several visual design patterns that were apparent even in these very early stages of simulation design. This analysis helps us to characterize the design constructs people may be able to specify or learn when first encountering visual simulation design tasks and projects. In this paper, we introduce the concept of patterns and their relationship to visual programming and present several visual programming language patterns mined from the projects developed at the community simulation design workshop. Finally, we discuss how these patterns might be incorporated into minimalist tutorials that we are developing to introduce community residents to visual simulation design.

1. Introduction

Interest in patterns has fostered one of the fastest growing communities in contemporary software design [3, 5]. In recent years, the software engineering and human-computer interaction communities have enthusiastically embraced the patterns concept, due in part to the 1995 book *Design Patterns* [3, 5, 9]. A *pattern* is a recurring design problem-solution pair. The most important patterns capture important structures, practices, and techniques that are key competencies in a given field, but which may not be widely known [3]. Patterns of object-oriented design software provide design solutions that are concrete enough to immediately put into practice, and yet sufficiently abstract to apply to countless situations, limited only by the imagination and skill of the pattern user [14].

Erickson [4] has taken patterns out of the hands of solely experienced engineers and designers and disseminated them to all stakeholders in the design process (designers, engineers, managers, marketers, and users). Erickson proposes a *lingua franca*—a common language—for the design process, to address a need for shared concepts, experiences and perspectives.

In this research we are exploring the role of patterns in the learning and use of a visual simulation environment designed for non-programmers—Stagecast Creator [13]. Our approach is twofold: to identify patterns that emerge spontaneously in the initial design activities of novices, and to discover patterns in our own visual simulation designs that we believe might facilitate such activities. Our hope is that this will increase our understanding of how novices think about and approach these design projects, such that we can provide better instruction and tools in support of this work.

This effort is a part of a larger research project studying community-related visual simulations; a key element of the project is to establish a novel collaboration between members of differing ages and roles [11]. We are engaging diverse members from our local community (Blacksburg, Virginia USA) to work together to build community-oriented simulations in Stagecast Creator (e.g., noise pollution in a college town), and to share and discuss their projects with each other and other residents.

As part of this project, we conducted a one-day design workshop, where we asked a small group of residents (teachers and senior citizens) to conceive and design a small set of simulation projects. One goal of the workshop was to find out whether and how these residents could create informal design specifications. We found that similar design issues arose in different simulations. Instead of referencing an earlier design, they simply re-developed the idea for the new simulation. This led us to document several visual language design patterns, forming a beginning of the Visual Sims Patterns Language. This paper reports the results of this pattern analysis work.

There is no other work exploring design patterns for visual languages like Stagecast Creator, so our proposals are exploratory and tentative. It seems likely that our patterns can guide development of instructions and tools for novices, but the use of patterns raises other issues:

- Will residents understand and be motivated to use patterns in the development of their own simulations?
- Will use of patterns inhibit novice users' feelings of creativity in design?
- Can a Visual Sims Patterns Language provide a common language for adults, students and researchers to discuss visual simulation projects?

In the following sections, we first introduce the concept of patterns and their relationship to visual programming. We then describe a community design workshop and the design patterns it produced. We conclude with a discussion on how we plan to use these patterns as a basis for minimalist instruction on visual simulation design.

2. Patterns and pattern languages

We observe patterns in everyday structures: in buildings, in vehicle traffic, in organizations of people, and in our software [3]. Patterns first emerged in the field of architecture when noted architect Christopher Alexander wrote his landmark book *The Timeless Way of Building*. He believed that by documenting patterns, he could help people shape the buildings of their community to support life to its fullest. A *pattern* is a piece of literature that describes a design problem and a general solution for the problem in a particular context [3].

Less than ten years ago the object-oriented software engineering community adapted Alexander's concept of patterns [3]. Members of the software engineering community felt there was a need to develop designs that were specific to the problem at hand, yet general enough to address future problems and requirements [4]. Shortly after, the human-computer interaction (HCI) community began to explore patterns that might help to bridge the gap between software engineering and HCI [4, 6, 8, 12]. User interface designers searched for effective design tools based on proven knowledge. Patterns were thought to replace the use of guidelines to capture design decisions. User interface designers began to see patterns as a potentially better tool than guidelines because they are explicitly related to a context and are problem-centered [12].

A *pattern language* is a structured collection of patterns that build on each other [3]. A good pattern language should give designers freedom to express themselves and to tailor their solutions to the particular needs of the context where the patterns are applied.

Patterns and pattern languages have found use in object-oriented design, HCI, database management, and recursive spreadsheet languages [1]. Visual languages tend to be component-oriented, promoting a culture of sharing and reuse [10]. A visual design pattern language may contribute directly to such sharing and reuse.

3. Community design of simulations

The Blacksburg Electronic Village (BEV) is an advanced community network with a high level of active local participation [1]. The resulting high quality and pervasive access to network services has prompted the education and involvement of many segments of the population in the networked infrastructure [2]. The town of Blacksburg has an active populace of retired residents who have use the BEV for civic and cultural activities (e.g. voter registration, zoning regulations, mentor outreach). This makes them a particularly attractive and appropriate population from which to enlist participants in community-oriented simulation projects [2].

Our vision for the Community Simulations project is that community members at diverse stages of life (age and profession) will collaborate in the design and construction of visual programming simulations about community concerns. This will serve both to enhance participants' programming literacy, and to promote collaborative problem-solving about community issues [11]. Our initial work focuses on middle school students (who we expect to be attracted to the game-like nature of the simulation environment) collaborating with senior citizens (who have a history of community-oriented interests and activities).

4. Community design workshop

As a step towards identifying and motivating participants, we conducted a one-day design workshop in which nine community residents were introduced to simulations built in Stagecast Creator. They critiqued these examples with respect to realism and potential for engaging youth in community-oriented discussions. We then asked the participants to identify and design their own community-related simulation projects.

Workshop participants represented two populations: middle school teachers and senior citizens. We expected that the teachers would provide insight into simulation projects that would appeal to middle school students and provide a forum for dialogue. In contrast, the older adults were recruited as a first step in their involvement as mentors for the school children, and as a source of simulation topics relevant to the community at large [11].

The workshop was divided into several phases: Stagecast Creator introduction, idea generation, and two refinement sessions. Each session concluded with a 45-minute discussion period. We also demonstrated a prototype of the community Web site that would be used to access, post, and discuss simulations.

4.1 Idea generation and refinement

We began the simulation design activities by pairing each middle school teacher with a senior citizen. We hoped that this pairing would encourage cooperative brainstorming and design. Once in pairs the participants worked for 45 minutes generating ideas for simulations. We gathered a list of over 25 ideas, after eliminating overlaps; each pair generated from 8 to 12 ideas. During the discussion period following this session, the pairs presented their concepts, while others offered feedback.

Design ideas were refined in two 45 minute sessions. Each pair selected and elaborated 1-2 ideas, leading to a total of nine community simulation proposals (Table 1). We encouraged pairs to choose ideas that had a visual character, and that embodied social or moral lessons.

Table 1: Nine project simulations, including the proposed design challenges.

Pair	Simulation Name and Description	Design Challenges
1	*Flirting or hurting?* Girls standing in hall are approached by cool guys who taps one on the bottom; they smile and giggle. *To vote or not to vote?* Students on downtown avenue stop at voting booth, talk, get info, pass by then come back to help	• How to include dialogue, show emotions, design lockers, show boy touching girl, show girls chatting and guys walking by • Identify literature on the table, identify adults vs. middle school
2	*Bullies at BMS*: Girls and boys from different neighborhoods on the bus, one group gangs up on another, prevents seating *Traffic, to stop or not, that is the question*: A new school leads to increase parent/teacher/bus traffic *Noise pollution*: Group of students is having a large party in a residential neighborhood, loud music, lots of people, the police are called and issue a warning	• Waiting time for buses, taking another person' s seat • Yield right of way, speed limits, weather conditions, increased traffic • Conversation among neighbors, interaction with the police, increasing crowds
3	*Cliques*: Three stereotypical groups on a schoolyard, new nondescript students arrives, is left alone but eventually joined by others to form a new diverse group *Smart road*: User is “driver” moving along a smart highway, though a local valley, experiencing rain, sun, snow, etc.	• Responding based on clothing and appearance alone. • Effects of sunshine, rain, snow and ice on the cars (all are characters of simulation)
4	*Bullying for acceptance*: In cafeteria, students throws food, visual simulation appearance changes to show acceptance, escalates into pushing, fight, name-calling *Redevelopment of downtown Blacksburg*: downtown changes to show different stores, parking, walkways, etc.	• Reflection on the behaviors, label people who are different • Pedestrians vs. cars , how will the project influence the actors

Participants were given a template to record design proposals. This form asked them to name their project and its authors, describe the basic idea, where it will take place, what are the likely characters, what community issues are addressed and what skills might be taught through the development of this simulation. They were also asked to sketch what the simulation might look like. For instance, the “Flirting or hurting” project (see Table 1) is designed to teach boys not to touch girls inappropriately, and to teach girls that they do not have to stand around and allow the boys to get away with it.

4.2 Commonly recurring problems

After the workshop, we began a “problem mining” process to uncover possible visual design patterns. First we examined the design specifications produced. We concentrated our analysis on the summary descriptions, the skills that each simulation might teach, and the pictorial representation of project. After an initial review we found that participants had *not* listed skills that their projects would teach. Rather each pair listed the *design challenges* raised by each simulation (Table 1).

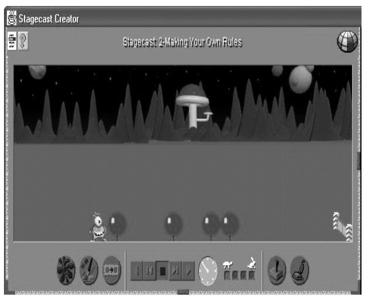
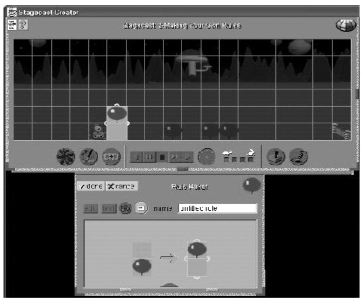
We then began step two, which involved developing meta-data to represent recurring design problems. For example, we identified situations in which contact was required between characters. A character, in the Creator environment, represents any object that is created in the simulation. In the proposed simulations, characters might be used to represent people, cars, rain, snow, stores, walkways and so forth. In “Flirting or hurting,” a boy touches a girl; this is an instance of contact between two characters within the simulation. We also find character contact in the Bullies at BMS, Bullying for acceptance, Smart road, and Traffic simulation. This design problem is addressed below in the “YOU' RE IN MY SPACE” pattern.

Working from these common design problems, we began writing patterns, using our own knowledge of visual simulation program to propose solutions. We did this in an iterative fashion, trying to make the context of the patterns ªfriendlyº to our participants. We chose terminology that our users might already know or could learn from our training materials. In the long term, we hope to provide a common language that we can use to understand and communicate design problems. We also hope that our participants will use this vocabulary in their online discussions about the simulations, sharing and reacting to each others' design ideas.

5. Patterns in Visual Simulations

It is generally agreed that a pattern contains several essential elements [2, 4]: Name, Problem, Solution and Consequences. We also include a Stagecast Creator scenario in our patterns, to illustrate the design problem. We hope that the scenario will help users understand the more abstract description of the pattern. The patterns are presented in Table 2-5.

Table 2 Pattern I - "You're in my space"

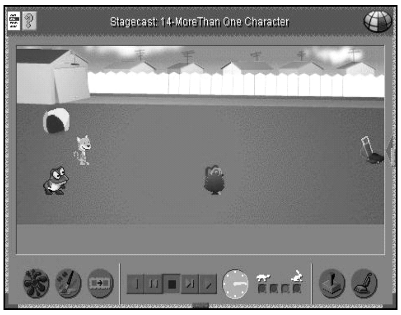
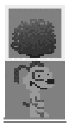
Name: YOU'RE IN MY SPACE		
Problem	**Applicability**	**Possible Consequences**
If an object is inhibiting another from performing certain behaviors because the two objects have encountered a territorial conflict, there is a need to decide how each object will be expected to behave.	Use this pattern whenever this there is a situation where a territorial conflict causes one character to not perform the desired behaviors.	If it is necessary for one character to continue to perform a desired behavior, we must alter the behavior of the other character. It is important to decide which characters' behavior should be temporarily altered to accommodate the territorial conflict. Often the least significant character has its behavior altered.
Scenario	**Stagecast Creator Solution**	
Imagine a gaming simulation where an alien is expected to walk around the stage and exit through the caution gates end of the stage to reach his space ship. The walking alien encounters balloons in its path. Because the alien does not have a rule that handles such situations, the alien can no longer walk to the desired location. 	The implementation varies by simulation. The implantation of this pattern in relation to the scenario is as follows: 1. Ensure that characters are positioned next to each other. 2. Alter the behavior of the character that is least significant at this point of the simulation (the balloons). 3. Create a rule in the balloon that is triggered when the alien is in position to the left of the balloon. 	

- ªYOU'RE IN MY SPACEº (Table 2) – this pattern takes into account the situation of spatial conflict resolution. For example, a character may be performing a certain behavior (e.g., walking) and is forced to stop this behavior because another character is ªin the wayº. The problem is to decide which character should continue its default behavior and which character should get ªout of the wayº. Potential applications of this pattern include the *Flirting of hurting* simulation to show the boy touching the girl; the *Bullies at BMS* simulation to show the bully taking another person's seat; the *Smart road* simulation to create the effects of the rain and snow on the cars; and the *Redevelopment* simulation to show the pedestrians vs. the cars.

- ªI'M IN FRONT OF YOUº (Table 3) – this pattern describes the design problem that occurs when two characters have to share the same space for some period of time. In a visual array, only one character can be ªon topº, so the problem is to decide which character should be in this position initially, and when the other character should appear. Identified uses of this pattern include the *Bullying for acceptanc*e food fight simulation, where students and food move in front of one another, and the *Traffic* simulation where show the police ªhide behindº the trees to catch law breakers.

- ªI DON'T CAREº (Table 4) – this pattern is triggered by the design problem that occurs when a character is expected to perform a set behavior regardless of characters that may be around it (e.g., walking next to other characters). We have identified that this pattern can be used in all the simulations described by the workshop participants. It is an essential pattern for designing school yards, hallways and downtown areas that are able to function with multiple people performing different tasks simultaneously (i.e. walking, stopping to chat, reading literature).

Table 3 Pattern II - "I'm in front of you"

Name: I'M IN FRONT OF YOU		
Problem	**Applicability**	**Possible Consequences**
You are creating rules on a stage and realize that you have reached a point where two characters need to occupy the same space on the stage, yet only one object can be on top at any given point in time. Questions to consider are: • How do you address the different layers of visual programs? • How do the objects isolate or cooperate within the same visual space?	Use this layering/organization pattern when • You want to layer your objects sharing the same visual space. • There is a temporal organizational need for objects to share the same visual space. There is a need for objects sharing a spatial location to perform certain behaviors that are triggered by the surrounding objects on the stage.	None Identified
Scenario	**Stagecast Creator Solution**	
You just created a dog that is running around the park playing in the grass. The dog spots a cat that he wants to scare so he decides to hide behind a bush. There comes a point where the dog is hidden behind the bush, and decides to jump out at the cat when she walks by. While the dog and the bush occupy the same space, you must decide when the bush is shown on top and when the dog is shown on top. 	1. Position the characters one on top of the other on the stage. It doesn't matter which is on top at this point. 2. On the PC, use one left mouse-click to show the objects in the same square. 3. Place the object that is designated to be on top, in first position.  To change the behavior of a character based upon other elements of the stage (i.e. the cat coming toward the bush), create a rule in the character that enacts this swapping behavior (e.g., a rule that causes the dog to move to the front when the cat is next to the bush).	

- ªWE REGULATEº (Table 5) ± this pattern describes situations when there is a need to control the global climate of the simulation. Often called a global variable, this pattern should be used whenever there are situations that require the multiple characters to stop performing their current behavior and react to the global change in the simulation (i.e. tension control). We have identified the use of this pattern is such simulations as *Bullying for acceptance* food fight simulation, the *Noise Pollution* simulation as the neighbor calls the people and the *Cliques* simulation when new diverse groups are formed.

6. Discussion

The process of identifying and documenting patterns is usually referred to as ªpattern miningº. We label our analysis process as *problem* mining instead of pattern mining for several reasons. First of all, pattern mining is done by consulting design experts in a domain (e.g., object-oriented software engineering), and asking them to identify key problems and solution strategies. We did something very different: we asked *novices*, with minimal exposure to visual simulation programming or design, to identify design challenges associated with their ideas for community simulations. In pattern mining, the person interviewing the design expert simply documents the reported strategies in a pattern format. We instead examined the problems identified by the novices, and compared them with our experience, as experts in visual simulation design and programming. By doing this, we were led to focus on problems that simulation designers with very modest exposure could foresee. These are problems other novices may also encounter, and will need help in how to solve them.

Table 4 Pattern III - "I don't care"

Name: I DON'T CARE		
Problem	**Applicability**	**Possible Consequences**
In a visual programming environment, behavioral characteristics of a character are triggered by the timing, staging, and location of other characters. When characters are moving around the simulation it is likely that one character will enter into the spatial proximity of another. Normally, in order for a character to behave, it must specify *exactly* the objects present when the rule is created. Sometimes the character must begin or continue a specific behavior regardless of what else is in close proximity.	Use this pattern whenever there are situations where it in necessary for a character to continue performing a particular behavior, regardless of other characters around it.	There are occasions when the behavior of character should change as a result of other characters in close proximity. But once the ªdon't careº indication is attached to a space the activity in that space is ignored. A more complex rule structure may be necessary that sets an order to how the rules are checked.
Scenario	**Stagecast Creator Solution**	
If we have a simulation in which a man is walking his dog in the park where there are trees, benches and other walkers, it is not necessary for the man and his dog to stop their walking behavior just because they pass a tree to the left or to the right of them; you want to indicate that you don't care. If the man encounters an object directly in his walking path, it may be necessary to use the **YOU'RE IN MY SPACE** pattern to resolve the spatial conflict.	To use the don't care square: It is found with the And-If Tools • Click on the tools to pick it up • Move the tool to one of the squares in the before picture. Click again to drop it. A gray border appears are the square.	

We are excited about the discovery of these visual simulation design patterns and we are actively trying to identify more. This study provided us with data that suggests that even novice simulation designers are able to predict potential design problems (i.e., as documented in the challenges they described, see Table 1). In future work, we hope to show that, with proper training, these novices will be able to solve such design problems.

As we work to build a community of Stagecast Creator simulation designers, we aim to provide these users with a common vernacular to discuss the design decisions made when they design or build simulations. The patterns presented in this paper are one way of assigning a *name* to common concerns and concepts that arise when during visual simulation development.

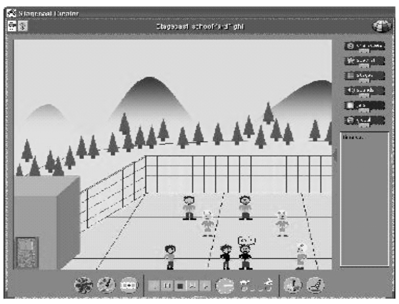
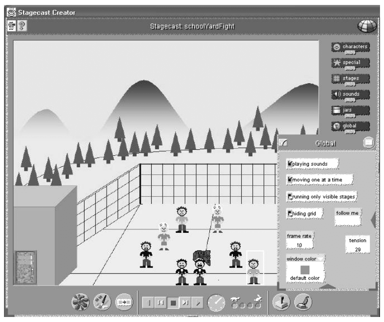
Researchers working with pattern languages have distinguished between language *idioms* and design patterns [3]. Idioms are low-level patterns that depend on a specific implementation technology such as a programming language. A software design pattern is broader in scope than idioms; it is intended to be language independent, and to represent general design practices [2].

We refer to our analysis as patterns, even though the scenarios and solutions are tied to Stagecast Creator (and thus are more appropriately classified as idioms). Our hope is that we can broaden the problem descriptions and solution strategies, demonstrating their applicability to other visual programming domains. For instance, we expect that ªYOU'RE IN MY SPACE,º ªI'M ON TOP OF YOU,º and ªI DON'T CAREº will apply to any visual design domain in which the spatial position of objects is used to control their behavior, but in which other objects are allowed to move freely in the space. At this point, we have no further analysis or empirical data to support such a generalization. However, our future work will investigate this possibility.

Users of the patterns described here need only have a basic understanding of Stagecast Creator. This can come from completing a tutorial or by past experience in creating simulations. We want to provide users with simple reflection devices that will help them to address the design problems that commonly arise when creating visual simulations. We want to alert the users to some of the design decisions that might be made when creating simulations. At the same time, we hope to guide them toward possible implementations of these decisions. Patterns such as those presented here do not dictate what a simulation should do or how it should operate, but do offer ways of recognizing and addressing design issues, as well as discussing these issues with other users.

Table 5 Patten IV - "We regulate"

Name: WE REGULATE		
Problem	**Applicability**	**Possible Consequences**
Often we want characters to exhibit certain behaviors based on the ªglobal climateº. There should be some way of allowing multiple characters to adjust their behavior in response to a global change to the simulation world.	Use this pattern whenever there are situations that require characters to stop performing the current behavior and react to global changes in the simulation.	None Identified

Scenario	**Stagecast Creator Solution**
We have a school yard in which students just hang around after school before the bus comes to pick them up. Two students are having a heated conversation and start yelling; other students egg them on. This increases the level of tension in the school yard. When the tension gets too high, a fight breaks out. 	1. Click on the global drawer to open it. Global variables will appear. 2. Use the new variable tool to create a blank variable. 3. Select the new variable's name and type a new one. 4. Type a starting numerical value in the white area below the name.  Stagecast Creator Stagecast: schoolYardFight Global playing sounds moving one at a time running only visible stages hiding grid follow me frame rate 10 tension 29 window color default color 5. Create rules that test this variable by dragging it from the global variable drawer onto a rule test, and testing to see if it has reached a threshold value. 6. Create rules that change this variable by dragging it from the global variable drawer onto a rule action that adds or subtracts to its value.

6.1 Related work

We have found no previous work on the use of patterns in visual simulation design. Research on visual simulations has primarily focused on the development of new environments or pedagogical techniques to motivate children to use visual environments in the creation games or scientific models [7].

In the larger visual programming community, there is an existing tutorial on the use of visual programming patterns for intermediate users of Visual Age Java [6]. These patterns were designed to provide a toolkit of readily reusable design patterns, so as to avoid some of the pitfalls of learning and using a visual programming environment. The documentation within the tutorial primarily consists of sample Visual Age Java visual programs, giving both the anti-pattern (illustrating the problem) and the visual design pattern (suggesting the design solution) [6].

6.2 Future Plans

In parallel with our analysis of visual simulation design patterns, we have been building and evaluating an introductory tutorial designed to teach middle school students about Stagecast Creator. The tutorial is *minimalist* in design, emphasizing rapid start up on meaningful tasks, minimal verbiage, and support for error recognition and recovery [Carroll]. As part of building the tutorial, we have carried out a formative evaluation, recruiting students aged 11-13 who expressed an interest in learning more about the Creator tool [11]. The students were generally successful in working through the tutorial materials and reacted favorably to the concept of using Stagecast to create or refine simulations.

In the near future we will incorporate our work on patterns into more advanced tutorial materials. Note however, that we do not expect that a ªdesign patterns catalogº similar to those provided for software engineers [5] is appropriate for this novice designer community. Rather, we intend to follow the minimalist approach of our successful introductory tutorial. Like this initial tutorial, the advanced tutorial will be example-based: learners will first be guided through a realistic design problem and solution that illustrates one of our patterns in use. We will then use carefully constructed ªreflectionº questions to introduce the pattern names and encourage the learners to adopt these names as labels for these common problem-solution pairings.

We also plan to examine the usefulness of the patterns described here for other visual programming environments. We will begin with a visual language that is quite similar to Stagecast CreatorÐ Visual AgenTalk for Agentsheets [9], which also relies on visual specification of before and after states to program characters' behaviors. We will then examine environments that are more distinct, for instance Visual Basic or Macromedia Director. Our goal in this is to identify and validate visual programming patterns that are as general as possible while still being useful for specific problem contexts.

Acknowledgements

This work is supported by NSF ITR EIA-0081102. We thank the other members of the Community Simulations research project ± Matt Capanile, Jason Snook, Jennifer Wissman, Trent Tolley, and Eric Dooley. We also thank the nine community residents who participated in our first design workshop.

References

[1] M. Burnett, B. Ren, A. Ko, C. Cook, and G. Rothermel, ªVisually Testing Recursive programs in Spreadsheet Languagesº, *Human Centric Computing* '01, IEEE.

[2] J.M. Carroll, and M.B. Rosson, 'Developing the Blacksburg Electronic Villageº *Communications of the ACM*, December 1996, 39(12), pp 69-74.

[3] J. Coplien, *Software Patterns: SIGS Management Briefings*. SIGS Books and Multimedia, New York, 1996.

[4] T. Erickson, ªSupporting Interdisciplinary Design: towards Pattern Languages for Workplacesº, Workplace Studies: *Recovering Work Practice and Informing systems Design*. (ed. P. Luff, J. hindmarsh, C. Heath). Cambridge: Cambridge University Press 2000.

[5] E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software* ,Addison Wesley Longman, Reading, MA, 1995.

[6] M. J. Mahemoff and L. J. Johnston. ªPrinciples for a Usability-Oriented Pattern Languageº, *OZCHI* '98 Proceedings, Adelaide, Australia, Dec 1998, pp 132-139.

[7] D. Parsons and M. Cranshaw, ªPatterns of Visual Programmingº, *Object Technology* '99 Tutorial, British Computer Society, Oxford, England.

[8] L. Pemberton. ªThe Promise of Pattern Languages for Interaction Designº, HF 2000, Loughborough University, UK.

[9] Repenning, A., and J. Ambach, ªTactile Programming: A Unified Manipulation Paradigm Supporting Program Comprehension, Composition and

Sharing°, *Proceedings of VL 1996,* Computer Society, 1996, pp. 102-109

[10] Repenning, A., J. Ambach, ªThe Agentsheets Behavior Exchange: Supporting Social Behavior Processing°, *CHI 97 Extended Abstracts,* ACM, 1997, pp. 26-27.

[11] M.B. Rosson, J.M. Carroll, C.D. Seals and T.L. Lewis, ªCommunity Design of Community Simulations° *DIS* 2002, Arlington, VA.

[12] J. Tidwell. ªCommon Ground: A Pattern Language for Human-Computer Interaction Interface Design°, http://www.mit.edu/~jtidwell/ui_patterns_essay.html, 1999.

[13] Smith, D.C. & Cypher. A. 1999. Making Programming Easier for Children. In Druin A., ed. *The Design of Children's Technology,* Morgan Kaufmann, San Francisco, 1999, pp. 201-222.

[14] M.V. Welie, G.C. van der Veer and A. Eliens, ªPatterns as Tools for User Interface Design°, *International workshop on Tools for Working with Working with Guidelines,* Oct. 7-8, 2000, Biarritz, France, pp. 318 ± 324.