Notes on Computer Game Design Innovation with Patterns

Kevin McGee
Communications and New Media Programme
National University of Singapore
Singapore
mckevin@nus.edu.sg

ABSTRACT

How can we help people design well-formed and innovative games? The design Patterns of Christopher Alexander is one methodology that has been proposed to assist in the design of well-formed artifacts. However, most work on game-design Patterns to date has opted either for "best practice" style Patterns – or for an alternative model of Patterns to support game innovation. This paper describes initial work to develop materials to help developers identify and formulate "best practice" game design Patterns – and to use the resulting Patterns as part of creating innovative games.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques; H.1 [**Information Systems Applications**]: Miscellaneous

General Terms

Design, Human Factors, Theory

Keywords

Design Methodologies, Game Design, Design Patterns, Game Innovation, Computer Games, Design Education

1. INTRODUCTION

How can we help people design well-formed and innovative games?

Of the myriad methodologies for game design, when it comes to the actual process of designing something well-formed, the majority provide fairly general guidelines applicable to the design of any computer application (e.g., prototype, test, iterate) – or provide more or less crisp *evaluation criteria* for game prototypes. Similarly, work on methodologies for game innovation is mostly in the form of generic "how to" techniques for either the brainstorming of new concepts or for "translating innovative concepts into design."

IE2007 December 2007, Melbourne, Australia

The design Patterns of architect Christopher Alexander and his colleagues is one methodology that has been proposed to assist the design of well-formed architecture [2, 1]. Such Patterns are precise but flexible design rules that express a relationship between particular design contexts, forces (psychological, social, or structural constraints), and *desired* ("positive" or good) features. Design Patterns have been extended to such computer-related domains as software development [5, 6], HCI [4, 9], and computer game design. [3, 7].

Alexander's architectural Patterns are clearly *rule-like* and emphasize a feature of the built environment that is *experienced as good by the inhabitants.* For various reasons, most work on computer-related design Patterns has dropped one or both of these characteristics. Much of the work on software-development and HCI Patterns, for example, has retained the focus on rule-like (or "best practice") formulations, but has concentrated on Patterns that articulate best-practice knowledge about what is good *for developers* rather than for end-users.[1]

Another major limitation of work done on Patterns by the "best practice" community is that, although there is some work on formalizing the necessary elements and stylistics of Pattern creation [8], such work is often done in terms of a desired result rather than particular techniques for achieving those results. More importantly, there is very little in the literature – beyond Alexander's initial sketch [1]) – about the *process* of identifying and articulating the information necessary for a well-formulated Pattern. There seems to be an unspoken assumption that this is a social process of trial and error. Thus, there is very little in the way of proposals for particular tools or guidelines for creating and refining the content of a Pattern. For example, once one has identified a potential Feature, how does one identify relevant Forces? How does one articulate those Forces so that the express *user* (rather than, say, *developer*) concerns?

Furthermore, the main focus in Pattern development has always been on the articulation of design knowledge about what is already known to be good. One aspect of this is that, unlike the emphasis on "novelty" in much design work, the development of architectural Patterns has always emphasized identifying features where occupants respond with *recognition*: "of course." The Pattern community at large has taken this very much to heart, emphasizing the every-

[1]Of course, architectural Patterns are also intended for *makers*, but for Alexander, the *validation* of a Pattern is always in terms of whether *occupants* experience a built structure with the Pattern as better than one without it.

day, commonsense characteristic of Patterns. Indeed, many go so far as to emphasize the importance of anonymous authorship, since the focus should be on the articulation of existing "common knowledge" rather than individual creativity. Said another way, within the Patterns community there is a pervasive commitment to the belief that Patterns are not intended to support innovation, but rather to clearly document "what works."

This raises a challenge if we are considering Patterns as the basis for supporting *innovation* in game design. To be sure, architectural Patterns support some degree of novelty: each house, although generated from a specific Pattern language (collection of Patterns), is distinct and original. But the very design of Patterns seems to discourage their use for the creation of innovative alternatives.

One approach to Patterns that might be a candidate for supporting game design innovation is the work of Bjork and Holopainen [3]. They explicitly claim that one potential function for their Patterns is "idea generation." Given what has been said so far, it should not be surprising that their *model* of design Patterns differs from the typical "best practice" model advocated by Alexander and most other Pattern developers. Rather, for Bjork and Holopainen, "game design patterns are semiformal interdependent descriptions of commonly recurring parts of the design of a game that concern gameplay" [3].

Specifically, their game design Patterns are largely *descriptive* rather than proscriptive (or rule-like). For example, their proposed game Patterns include such things as Lives, High Score List, Time Limit, and Power-Ups. Readers familiar with computer games should recognize these as *possible*, rather than necessary or good, attributes. And although the choice/use of Alexander's can also be optional for particular architectural structures, Alexander's Pattern formulation also specifies *when/where* a Pattern *should* be used ("context"). The way Patterns are formulated by Bjork and Holopainen does not include such proscriptive information. Their assumption seems to be, then, that their Patterns can function as a source of *inspiration* for the choice of potential game features; and, because of the way their Patterns are described and structured, they can indicate the possibility of related Pattern dependencies and conflicts. Unlike best-practice Patterns however, they are not *rule-like* and offer less guidance about their use in the process of design. Bjork and Holopainen might argue that their approach gives designers more creative freedom than the "best practice" approach. Nonetheless, the question remains whether it might be possible to combine the "best practice" approach to design Patterns with the notion of supporting game design innovation.

This then is the two-fold purpose of this paper: to sketch an example of how written materials can support the Pattern-creation *process* for game design – and to illustrate the possibility of Pattern-based *innovation*. Specifically, are there tools or techniques that can be used to guide Pattern creation and refinement – and once a viable, rule-like game design Pattern has been created, can it then be used in the process of design innovation? This paper describes some initial results derived from trying to address these two issues.

The structure of the remainder of this paper is as follows. First, Alexander's original methodology for creating architectural Patterns is summarized. This is followed by a description of an analogous process we have used to help students to create design Patterns for computer games: the tools we have used and the way we have used the resulting patterns as a source of game-design innovation.

2. CREATING DESIGN PATTERNS

Although design Patterns are widely known, less familiar is Alexander's proposal for the *process* of identifying and articulating them [1]. Since we have largely followed this approach in our work on game design Patterns, we will briefly sketch Alexander's model to provide context for our work.

A canonic example from the work of Christopher Alexander presents the problem this way: imagine there some particular room that you really like – some quality that makes you feel, *this is a good room*. Now, imagine you are going to ask someone to *build* a building – and you would like to ensure that the new building has a room with the "quality of goodness" experienced in the room that you like. How can we *describe* the architectural feature of the room you like in such a way that the builder *can make another one* that has this quality? Further, assuming that the new room cannot be an exact duplicate of the room you like, how to articulate the quality in a way that takes variety of architectural spaces into account? To be clear, this is not a question of personal taste ("the room is blue because blue is my favorite color") or experience ("the room is special because it is where I experienced my first kiss"). It is also not nebulous or vague: we cannot ask the builder to imbue the new room with the "aura" of the room we like.

Alexander's belief is that traditional builders knew the answer to this question: they *copied* existing buildings that were good. But they did not copy particular elements; rather, they copied the *patterns* of existing buildings. This, according to Alexander, is why buildings in traditional villages all feel as if they "belong together" even though each one differs from the other. Alexander's work on architectural design Patterns was initially motivated by the desire to make explicit the Patterns he felt were intuitively learned and understood by traditional cultures where people built their own buildings. Thus, Alexander proposes a two-part process for modern users of architectural Patterns. First, to the extent that there exists an explicit Pattern language, people should select from the language the Patterns that are relevant to their specific project. Then, to the extent that people feel there are Patterns "missing" for their particular project, he briefly outlines a process for inventing new, appropriate Patterns.

An Alexandrian Pattern is presented in the form of a description that highlights the Pattern's *name*, the *Forces* (human concerns) it addresses, the *Feature* that resolves the tension of those Forces, and the Pattern's *Context* (the *when* or *where* that this Pattern is appropriate).

- Feature: *what* is the "something" we want to build?
- Forces: *why* is this "something" helping to make the built structure Good?
- Context: *when* (or, *where*) will this pattern work?

As an example of a Pattern that Alexander and his colleagues articulated, consider their perhaps best-known example: when designing a building, make sure that there is natural Light on Two Sides of Every Room. Below is an abbreviated version of the Pattern.

Name: LIGHT ON TWO SIDES OF EVERY ROOM

Forces: In rooms lit on one side, the glare which surrounds people's faces prevents people from understanding one another.

Therefore:

Feature: locate each room so that it has outdoor space outside it on at least two sides, and then place windows in these outdoor walls so that natural light falls into every room from more than one direction.

As we can see, such a Pattern is in the form of a "positive" *rule*: it specifies something specific to achieve – rather than being a rule of the form, "do not do X."

There have been many debates about the validity of specific Patterns (and the Pattern approach in general). For our purposes here, we wish merely to emphasize certain characteristics of the Patterns. They are *operational and precise* (we "know what to do" to realize a pattern); *flexible* (there is more than one solution that satisfies a pattern – e.g., there are a multitude of different ways to attain "natural light on two sides of a room"); *testable* (we can confirm empirically whether people feel better in structures that contain the Pattern – versus those that do not); *debatable* (it is clear enough to criticize); something good about *end-user experience* (that is, not just something good from the perspective of builders/implementers); often *obvious* (in retrospect).

How do we create new Patterns? The main method sketched by Alexander is roughly as follows. Start by noticing an architectural situation where one feels good. Now, try to identify something *architectural* that contributes to this good feeling: try to articulate it in the form of an architectural relationship that can clearly be present (or not) in a structure. This aspect of Pattern creation can take a long time, with many iterations of discussion and revision. Once one feels one has identified such a Feature, work to identify the conflicting Forces it resolves. Finally, identify the Context in which it is relevant (for example, Alexander claims that LIGHT ON TWO SIDES OF EACH ROOM is not necessary for rooms that are very shallow – or that have very tall).

3. CREATING GAME-DESIGN PATTERNS

Following Alexander, we would like good methods for identifying the features of games that make them *good* – and for articulating *design knowledge* about those features that makes it possible to communicate what they are *and how to build them*.

One incident will serve to illustrate *what* I have in mind as a target for such design Patterns – and some of the difficulties about *creating* such a Pattern. In 2001 I taught a university course on the design of media technologies to support end-users. The course consisted of different design teams and they were required to use the method of identifying, articulating, and applying a design Pattern relevant to their project-focus. One team had members with children who were actively engaged in making stories in the form of physical "multimedia books" by folding paper, cutting out pictures to glue on, drawing, creating texts, and so on. One of the issues was that sometimes the children "couldn't think of anything to make a story about." This team's goal was to design a computer-based tool/environment for making analogous multi-media stories. In particular, they wanted their tool to address the problem encountered by one of their test-users: "how can the system help when users 'run out of story ideas'?" (Before reading further, readers might consider for a moment how they would try to solve this design problem.)

Since we were working specifically with the use of design Patterns, the team first tried to identify an existing example of "success." In other words, they tried to identify some example where someone had "writer's block" and "something helped." One of the team members started trying to learn "what her son did" in such cases.[2] Eventually, after many attempts on the part of his mother to formulate partial insights in terms of a design Pattern, it was the son who almost casually articulated the design Pattern himself: PLACE A WELL-KNOWN FIGURE IN AN UNUSUAL SITUATION.

It turned out that the boy liked to draw a particular character and make stories about the character. Whenever he couldn't think of a story, he would put the character in odd situations to see if it inspired him. In this case, the particular character was "well known" to the boy because it was the boy's own creation, but the same heuristic can obviously be used with characters that are more widely "well known."

There are many insights to be drawn from this example, but for now it is enough to highlight how it meets the main requirements of a design Pattern: it is centered on the end-user experience, generative, flexible, testable, and it is clear enough to be debatable. Finally, it is almost *obvious* (in retrospect). Indeed, it is a well-known heuristic for generating new ideas formulated as a design Pattern. Although each team in the course arrived at more or less successful design Patterns for their projects, it became clear that it would be nice if there was better support for students in the *process* of discovering/creating such patterns. One obvious target would be "a pattern-language for creating patterns." The next section highlights some initial attempts to start creating this.

3.1 Game-design Pattern Course

Subsequent to the course described above, I co-taught a course on the design of computer games. My main role in the course was to facilitate the creation and application of design Patterns to the computer games each team was required to implement.

The Pattern-relevant structure of the course was as follows. In order for students to appreciate the value of the Pattern approach, in the first session students were introduced to some architectural design Patterns from Alexander. We then had them work in five small teams and use a limited set of the patterns to design a workspace for twenty-five (25) people. When they were done, we discussed the *design process* they had just experienced. A number of students commented with surprise that it was more *design*-oriented than most of their experiences "designing something with others." Those other design experiences were often dominated by arguments and debates of "personal opinion" and "being lost." Although neither I nor the students were expecting design Patterns to be a "magic bullet" for the game design process, they were all nonetheless fascinated that their brief exposure to the use of Patterns led to a *process* of design that

[2]Readers familiar with such "ethnographic investigations of end-users" can imagine the richness of issues that arose. Exploring them here would take us too far afield, but are the subject of a separate paper recently submitted for publication.

was fairly concrete, focused, and productive.

Then the five design results were displayed for everyone. Students were surprised at both the variety of solutions that nonetheless met the Pattern constraints – and at the fact that the solutions together formed a *collection*. That is, although each particular design was individual and distinctive – although this result was not one of the goals of the exercise – looking at them together, we could see how they were *generated* by the same design language.

Finally, we also discussed the fact that "going the other way" would be difficult – that is, it would be a hard task to start by studying the resulting examples and then *deriving* a common pattern-language. Nonetheless, students were told that this "hard task" would be something we would do as part of the course.

During the remainder of the course, students worked in teams and each team was expected to implement three different types of computer games: a variation of *Breakout*, *Bul*, and *Hammurabi*. Teams were provided with an initial reference specification for the original version of the game – which they then had to implement. As they were working on this, each team was simultaneously working to identify a Pattern of something about the reference game that made it good/fun to play. When teams completed the implementation of the reference version of the game, they were then required to modify the implementation to make their own original variation. In particular, they were expected to modify the original version so that the particular design Pattern identified by the team had a "stronger" presence in the new game.

The Pattern-creation process was one in which each team proposed an initial draft Pattern, posted it to the course members, and then we would devote a design session to discussing each Pattern, evaluating them and proposing improvements. Simultaneously, during lab (implementation) sessions each team was given feedback about their efforts to incorporate the Pattern into the new game. Final submission of each project consisted of the game implementation, a single-page design Pattern, and a short (half-page) document describing how the team's design Pattern was realized in its game.

Thus, the course involved a lot of the social process of discussion and iteration. However, the remainder of this section describes some of the specific written instructions and templates we used to support the process of Pattern identification and articulation. In the interests of brevity, the example documents only relate to the identification of Features and Forces. Also, although the course ultimately resulted in fifteen design Patterns (five student-teams, each producing a different Pattern for each type of game), the description will highlight the identification, formulation and application of one particular Pattern by one of the teams.

3.1.1 Stage 1: Pattern Creation

In preparation for developing Patterns, students were given a sheet that summarized certain "canonic wisdom" about Pattern development (e.g., "identify something you *actually find fun* – not something you are 'supposed' to find fun"). They were also given a game-design Pattern document that included guidelines and a Pattern template.

Game-design Pattern Guidelines/Template

Name: It is very important for a Pattern to have a good name. A good name is usually in a form that is *an answer to the question*, "What should a designer *make*?" Note that an architectural pattern name describes a *physical attribute*. For computer game, the pattern name describes a *procedural relationship* – that is, a mapping of user-interaction to game behavior.

Example pattern-name: " Light On Two Sides of Every Room"

Warning! With a good name, it should be possible to answer clearly: "is the pattern *present* and *significant* in the intended artifact?"

Example: "Does the room have natural Light On Two Sides?"

Forces: The Forces should be things that *people care about* (psychologically, emotionally, economically, etc.). The description of Forces should include a *maximum* of *two* forces – and they must be in *conflict*.

So, the pattern for a description of Forces:

- Force 1: if a game does *not* have/allow [A], then players will experience problem [X].
- the word "But,"
- Force 2: if a game *does* have/allow [A], then players will experience problem [Y].

Do *not* include "solutions" in the descriptions of Forces. Rather, one force usually expresses a problem that happens if we "go to far" with the *opposite* of the other force.

Feature: The feature should be something "positive" – that is, it should describe something we *should* make (as opposed to something we should not make). The description of Forces should present a kind of *conflict* – and the pattern description continues by saying, in effect, "therefore, designers should *make* (or *build* or *do*) the following *solution* (feature)."

So, the pattern for a description of Feature:

- the word "Therefore,"
- a word that means *making* or *building* or *doing*
- the specific *rule to follow* (feature) that will allow designers to resolve the conflicts (forces) described.

Observe that although a feature is a *rule to follow*, it is almost never expressed in terms of a single *value* ("make it *blue*" or "build it *20 meters long*"); rather, it is a *ratio* ("place windows so as to create natural light on two sides of a room") or a *relationship* ("put the doors as near the corners of the room as possible").

Example

Name: Difficulty-increase Driven by Player

Force 1: if a game does *not* have/allow *challenge*, then players will experience problem of *boredom*.

BUT,

Force 2: if a game *does* have/allow *too much challenge*, then players will experience problem of *"it is too hard" and just give up*.

Feature: **Therefore, implement games so they start "easy", but the increase in difficulty is directly driven by the rate and degree to which player succeeds at the challenges.**

Before engaging in the use of this template document to create their first drafts, students also read some additional material from Christopher Alexander and attended a combination lecture/discussion on the use of the template and ways to apply it toward the analysis of the first game (*Breakout*). Below is part of one team's first draft of a design Pattern developed for *Breakout*-like games:

Name: Reward increased Hand-eye Coordination

Forces: For games that require timing and hand-eye skill:

Force 1: If a game behaves randomly and the player cannot effect the game sufficiently in a positive direction, the player will lose interest in improving and will stop playing.

But,

Force 2: If the game provides rewards for increased hand-eye skill in the form of better results then the player will discover a reason to continue and try to improve.

Feature: Therefore, provide rewards in the form of better results for hand-eye coordination and skill.

The Pattern shows insight into something important, but it is still rough. For example, the Feature is a bit too general for a developer to really use (what is "better"?), etc.

3.1.2 Stage 2: Pattern Improvement

After each team submitted a written draft of a design Pattern, students were provided with a checklist to support an evaluation and discussion of the different Patterns proposed. Specifically, each student was asked to look at the patterns that all the other teams had submitted and to try to identify one thing concretely they could say about each pattern that needs improvement – and one concrete suggestion for *how* to improve it.

As context for this activity, students were advised that Patterns are initially weak *hypotheses*. We need to develop them to the point where they are *strong enough to test* – and then we need to start testing them. As part of probing an initial proposal for a Pattern, one should be broadly evaluating Force and Features as follows. Forces: *is it true that the stated "conflicting forces" actually do occur in the stated context*? Feature: *is it true that the stated feature actually does resolve the "conflicting forces" in all cases*?

In addition to such broad guidelines, students also received an evaluation checklist:

- Is the Pattern *really* present in the *original* game?
- Do you "believe" the Pattern – does it express something about the game that actually makes the game *fun to play*?
- Is it well-described:

  Forces

  - Is each Force a *real Force* (that people *really care about*)?
  - Is each Force *relevant to the game*?
  - Are the Forces in *conflict*?

    Warning! *One Force is not a "solution" to another Force*. Rather, one Force usually expresses a problem that happens if we "go too far" with the *opposite* of the other Force.

  Feature

  - Does the Feature actually *resolve the conflict in the Forces*?
  - Is the Feature expressed as something we can *do*? (example: "therefore, *do/make* X")

  Name

  - Does the Pattern *name* clearly express the Feature we should build?
- Do you find yourself *nodding in agreement* as you read the Pattern description?
- "If I had to *use* this Pattern to *build something*, would it help? Would it help *enough*?"
- Does the Pattern suggest *interesting* ways to *improve the game*?
- Is pattern clear enough that we can separate games that have the Pattern from those that do not?
- Would games of the same type that do not have the Pattern be *more fun to play* if they did?

As a result of discussions and feedback based on using this template, the earlier student Pattern was revised to be:

Name: Reward in Kind

Forces: For games that require quick hand-eye skill:

Force 1: if the game does *not* provide players with rewards, players will experience the feeling that "there is no point" in playing.

But,

Force 2: if the game *does* provide provide players with rewards, then players may lose interest in the *play* (and start to care only about the rewards)

Feature: Therefore, reward player's success at some demonstration of skill by giving the player increased challenge of the same kind. For example, rewards for increased hand skill should be in the *form* of something demanding even further increases in hand-skills.

3.1.3 Stage 3: Pattern-based Innovation

The team that developed the REWARD IN KIND Pattern wound up creating a number of interesting extensions to the base version of *Breakout*. In some cases these were "obvious" extensions that were well known from commercial variants of the game (e.g., increasing the speed of the ball, number of blocks, speed of descending blocks, and number of balls). On the other hand, they also arrived at some interesting novel variations by focusing on different aspects of "what a player finds challenging" in terms of hand-eye coordination. This led them to implement, for example, situations where the ball(s) and/or blocks were made harder to see in various ways – and where the notion of paddle/ball control was modified.

Aside from the particular innovations, it is worth noting another aspect of this team's design process: the team members were remarkably *focused* on a consistent set of extensions to their implementation. By contrast, the process and results of teams that had a less clear Pattern – or, for some reason did not stay with the innovations suggested by their Pattern – were quite different. The new games implemented by those teams gave the impression of a "grab bag" of different game-play features. Not only that, but the process of development for those teams often got mired in incidental debates or concerns.

4. DISCUSSION

To be clear, the claim is not that the design Patterns made the only (or even the major) difference in team process or results. Certainly, there was a great deal of team-support that came in many different forms and it was not the purpose of the course to try and localize the consequences of particular kinds of support to the particular results. However, through-out the course there did seem to be a strong correlation between the degree to which a team arrived at a clear Pattern, the degree to which they remained focused on it as a source of innovation, and the degree to which the resulting implementation was interesting.

Beyond that, the results of the course suggest that "best practice" design Patterns may indeed contribute to design innovation. Furthermore, it may be fruitful to try developing a more elaborate "Pattern language for developing design Patterns." To be sure, the Pattern design Template described here is extremely limited (and obviously does not follow the format of typical design Patterns). Indeed, it is a Pattern only to the extent that it (somewhat) highlights tensions (Forces) – and it expresses desired solutions (Features) in the form of ratios/relationships.

A better example of the potential of Patterns for pattern design comes from the paper by Meszaros and Doble [8] mentioned earlier. Although most of the Patterns proposed in that paper provide "stylistic" guidance, they do propose three Patterns for *how* to create a good Pattern *name*. This is a crucial aspect of Pattern-creation, one that can be quite difficult – and one that makes a difference to *end-users* of Patterns. Furthermore, the emphasis they place on *method* in these particular Patterns is clearly an approach to meta-Patterns that is similar to the effort described here.

Even granting the limitations of the template used in our course, it is worth commenting on one particular aspect of it, namely the degree to which it restricted the way students were required to identify and specify Forces. There were a number of reasons for this. One straight-forward reason is that initially, when the template did not restrict students to two Forces, there was a tendency for teams to propose lots of Forces that were weak or unconvincing, rather than identifying one or two and improving them. But there were several other categories of difficulties in the formulation of Forces that suggested the need to restrict them. For example, Forces sometimes seemed unrelated – to each other or to the proposed Feature. Another common problem was the description of a Force as the "opposite" of the proposed Feature. For example, a proposed Force might be: "people don't like it when they don't have enough resources" – and the proposed Feature would be: "give them enough resources."

Beyond issues related to Forces, unfortunately the limits of space prevent elaboration of the many issues relevant to creating appropriate Pattern names, Features, and Context descriptions. To varying extents, these were raised during the course and attempts were made to address them with revisions to the course template.

5. CONCLUSIONS

One of the major issues we did not explore in the work reported here is the degree to which the approach outlined in this paper can lead to qualitatively new games, rather than variations on existing ones. This is a complex topic worthy of sustained examination. Nonetheless, the results described here suggest that it is worth pursuing.

In our experience, once designers have patterns, it is very clear "what to do" – whether one is trying to evaluate the extent to which a Pattern is present in an existing game or whether one is trying to improve a game by strengthening the Pattern. That is not to say it is *easy*. Design is still challenging – but it is hard in the sense that we are "engaged with a difficult and challenging problem" as opposed to "it is hard because we have no idea what to do." In this sense, design becomes difficult the way a good game is difficult – it is the kind of difficulty we *enjoy*.

There is certainly much work to be done on the development of materials to support the identification, creation, and revision of Patterns – whatever form those materials take.

5.1 Future Work

There are several aspects of Alexandrian patterns that become problematic in the context of game design. Here we highlight three that we are pursuing as part of ongoing research.

How can Patterns account for temporality or *process* in games – something that is not addressed by Alexander, with his claim that architecture structures *space*? Intuitively this seems like an essential distinction between software systems and (physical) architectural structures. Unfortunately, as with so many common sense intuitions, the viability of this distinction becomes problematic on close inspection.

A related issue is Alexander's claim, "if you can't draw a *diagram* of it, it isn't a pattern" [1]. A number of the proposals for HCI design patterns include illustrations or

pictures of target features. Nonetheless, intuitively it feels as if we would like something other than "can you *illustrate* it?" as a criterion for confirming whether we have truly identified computational design patterns.

Finally, in Alexander's original presentation of Pattern languages, there is a claim that it is not only possible to articulate the Patterns that make *existing* built environments good – it is possible to study the Forces at work in people's lives to develop *new* Patterns. There seems to have been virtually no work on this kind of Pattern development.

6. ACKNOWLEDGMENTS

Thanks to all my former students in the design courses at Uppsala University and Linkoping University for their energy, creativity, and discipline. Also, thanks to my colleagues in those courses, Oskar Jonsson and Mikael Kindborg, for supporting my "crazy ideas" about the design of media technology.

7. REFERENCES

[1] C. Alexander. *The Timeless Way of Building*. Oxford University Press, 1987.

[2] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1977.

[3] S. Bjork and J. Holopainen. *Patterns in Game Design*. Charles River Media, 2004.

[4] J. Borchers. *A Pattern Approach to Interaction Design*. Wiley, 2001.

[5] W. Cunningham and K. Beck. Using pattern languages for object-oriented programs. In *Proceedings of OOPSLA'87*, Orlando, Florida, 1987.

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

[7] B. Kreimeier. The case for game design patterns. *Gamasutra*, March 2002.

[8] G. Meszaros and J. Doble. A pattern language for pattern writing. In R. C. Martin, D. Riehle, and F. Buschmann, editors, *Pattern Languages of Program Design 3*, pages 529–574. Addison-Wesley Professional, 1997.

[9] J. Tidwell. *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly Media, Inc., 2005.