Abstract

The act of designing is pervasive in software development. As software developers we, de facto, design everything from business processes to user interfaces to program architecture to algorithms. Unfortunately, as software professionals we receive very little explicit design education or training. This results in a lot of design being done without conscious attention – we simply make design decisions based on innate cultural bias, standards, or idiosyncratic preferences. We are becoming increasingly aware of the need for “good” design and the power of design. We are even creating new fields of software related design practice – like user experience design – but still do not incorporate design thinking into our professional education. This paper introduces a nascent pattern language in support of design thinking.

Design Thinking

Designers are different. They are right-brained artists with esoteric knowledge of line, form, and color who somehow magically distill iPod form-factors from clouds of possibility. Or so the stereotype assures us.

Software developers are different. They are left-brained scientists with esoteric knowledge of arcane mathematical symbols, op-codes, and logic circuits who some magically distill payroll programs from clouds of possibility. Or so the stereotype assures us.

Never accurate, the stereotypes have been shattered in recent years as designers have been challenged to apply their modes of thought to problems far beyond logo, poster, and industrial design into user interface, business process, user experience, and social network design. At the same time, software developers have been confronting challenges beyond making the computer perform tricks to confronting the use of software in complex adaptive systems.
There is a huge overlap as designers confront the technical and technicians confront the artistic. Software developers are, inescapably designers. Something as simple as selecting a font for a drop-down menu or a background color for a Web page is an act of design.

Unfortunately, most software developers have no clue about what design is, how design is done, or even – too often – that they are involved in designing at all. The consequence of this lack in preparation is software that is overly mechanical; that meets technical specifications but not usability specifications; that is obviously and painfully disconnected with the context, human/business system, in which it is deployed; hard to use; and basically ugly. The one notably exception is gaming software where huge amounts of conscious design effort is manifest.

Designers, like Tom Kelley of IDEO, have been asked to extend their understanding of design to help business enterprises create a culture of innovation and adaptation. New software disciplines, communities really, like the UX (User experience) community, have been attempting to adopt and adapt design concepts and language to better perform their roles.

We desire to make a small contribution to the essential dialog between designers and developers. Our initial effort is focused on identifying and articulating, in a general pattern format, some of the important concepts that will eventually be included in a Pattern Language of Design.

The Patterns

We are violating the canonical form of pattern presentation. Our primary reason is that we are dealing with patterns of thought or activity and not of form. This is not a new problem and the PLoP community has been lenient in the past. We do try to stay true to the canonical form so some degree, specifically by including a Provocative (and hence memorable) Title, Problem, Context, Forces, Pattern Description, and Discussion sections for each pattern.

The patterns as presented will seem to be an ad hoc collection with not strong relationships among them. This means they cannot be construed as a pattern language, but it is our hope that future work can fill in some of the connections and create a true Pattern Language of Design.

Briefs or Boxers?

**Problem:** Design has an object, an outcome. Most design is done on behalf of a customer and the customer needs to articulate what it is that they desire. Of course the customer does not know exactly and precisely what they want; if they did they would not need a designer, they could go directly to a programmer and tell her precisely what to do. We need a means for the customer to convey sufficient information to the designer that she can do the necessary work.

The solution is a design brief – a formatted means of expressing the needs of the customer in a way that the designer can proceed.

**Context:** This pattern is useful at the beginning of a project and sets the goals and identifies any constraints that must be observed by the designer.
Forces: The primary forces involve a balance between ambiguity and precision. Certain parts of the brief must be ambiguous so that the designer has latitude to explore possibilities but other parts, inviolate constraints like budget or technology givens, must be as precisely stated as possible. The large number of unknowns present at the beginning of any project is another force. The customer may not know what they really want, in part because they do not know what is possible.

Pattern Description: A design brief is a story. As such it has a cast of characters, a plot, a script (a sequence of interactions among the characters), a set, and props. A design brief is usually presented in narrative form. Constraints need to be clearly stated, and it is often useful to do so outside the narrative. The plot of the narrative is a description of the state of affairs that will exist once the desired change in today’s situation has been designed and implemented.

Design briefs may be complicated and consist of multiple intertwined and nested stories. Patterns for Story Telling (Sugarloaf PLoP 2010 by Quillien and West) will be summarized here to describe the pattern description in more detail.

Discussion:

Software developers are clearly hung up on the idea of formal specifications. Project Managers expect a Project Plan document at the beginning of each project. These documents are formally defined and contain estimates and other values that are almost total guesswork. The Design Brief replaces these more formal documents, allows the design to emerge, and the design, in turn, allows for more accurate projections for subsequent planning purposes. The Design Brief pattern is also more consistent with Agile development practices that the more formal documentation currently in vogue in software development / project management.

Forever Jung

Problem: A design must be compelling to human beings for it to be considered “good” design. What makes something compelling is largely unexplored territory. However, there is significant evidence, known to experienced designers, of the power of certain colors, symbols, shapes, and other sensory signals to evoke certain reactions in humans. In many cases these ‘triggers’ are common across cultures and temporal eras.

Context: Design, particularly software design, projects may have multiple targets, some of which are hidden inside of a computer. Other aspects of design are directed to providing certain sensory input to human beings – e.g. visual, tactile, and auditory sensations. It is necessary for the designer to be aware of any correlations between triggers and known responses in order to make appropriate design decisions and selections.

Forces: We are very aware that small sensory triggers can evoke feelings of appreciation or repulsion in human beings. We also know that the same triggers may have differing effects in different circumstances, at different times, and in different cultures. We also know that some triggers seem to have common effects without regard to context. There are a lot of triggers and only some of them have been documented.
**Pattern Description:** This section will include a discussion of Christopher Alexanders fifteen principles as presented in his *Nature of Order*, as well as a discussion of Jung, Campbell, and other designers documentation of primal symbols. The pattern is basically one of “use these” and “avoid these,” a kind of structured reference.

**Discussion:**

The discussion will focus on variations and how to match the pattern’s structured reference to commonly encountered circumstances: i.e., when to use the pattern and when to eschew its use.

---

**No Man is an Island**

**Problem:** All humans are products of their culture and their circumstances. This can lead to an inability to see potential design solutions. Conversely it make make a designer “more compatible” with a given customer if they share the same cultural background.

**Context:** this pattern comes into play at the very beginning of a project as the client and designer seek a good ‘fit.’ It also should come into play when the designer is exploring possible solutions, both as a way to assure acceptability (cultural conformance) and to force the designer to make sure they are not simply falling into routine ways of thinking about potential solutions and thereby missing an optimal design.

**Forces:** The need to make the familiar strange and the strange familiar. One force is common to all systems, maintain the status quo and be sensitive to the strange, a form of xenophobia. Creativity and innovation, however, often compel something ‘different.’

**Pattern Description:** The pattern will show how to make the non-conscious conscious in order to expose both the designer’s and the client’s assumed cultural context. The pattern is largely a description of ethnographic techniques including the Thick Description of Clifford Geertz.

**Discussion:** The discussion will focus on how to use this pattern in the context of project inception and again during project retrospectives – the two times that it is most likely to be applied.

---

**Can’t We All Just Get Along?**

**Problem:** Design is largely a collaborative effort and therefore is subject to the same kind of challenges as any other teamwork. The problem can be exacerbated when the team is geographically dispersed. Better results come from diverse teams, but diversity presents its own set of additional challenges. A third set of challenges arise from Agile style development where design is ‘distributed’ across time and project phases.

**Context:** This pattern is useful when forming and ‘managing’ a team as they engage in a design project.
**Forces:** Innovation and creativity arise from diverse perspectives but humans with diverse perspectives have proven somewhat difficult when it comes to working together.

**Pattern Description:** This pattern is grounded in a redefinition of roles that lead to successful collaboration in design projects. The new roles are associated with existing roles, particularly those in an agile team. The roles themselves are derivative of, and summarize, Tom Kelley’s (IDEO Design), *Ten Faces of Innovation*.

**Discussion:** Will focus on how to apply the pattern in identifying individuals and associating them with roles and merging roles that are more distinctively design in orientation with those that are developmental.

---

**Magical Liminality**

**Problem:** At some point the designer must confront the design space with all of its potential and ambiguity in order to coalesce a concrete design product. This is largely a solo task which means the designer needs some degree of isolation from distractions – it is the period of literal design thinking.

**Context:** The designer has the brief and has spent time disambiguating that brief to the point that he is ready to “do the design.” At this point he may go to his office, close the door, think and doodle until something coalesces in his mind. Whatever that may be, it is the initial preliminary solution that will then be tempered with discussion and interaction with the design team and the client.

**Forces:** Lots of potential solutions, lots of evaluation criteria, lots of unknowns. The mental actions that take place in this situation can be overwhelming for a novice and still challenging for an experienced designer. It is a solo act that can benefit from direction or assistance by others.

**Pattern Description:** The pattern will involve a description of the liminal period discussed by von Gennep, Turner, and others as the middle stage of a rite of passage. The designer is literally in a state of betwixt and between – separated from what was, not yet integrated into what will be.

**Discussion:** will focus on why this fundamentally magical step can be made more systematic with the application of the pattern.

---

**Russian Dolls**

**Problem:** Design was once (and too often still is) seen as a kind of decision tree exploration – with decisions leading to partial solutions at each node of the tree. This is the kind of thinking behind the classic waterfall software development lifecycle. We now know better, and need a design thinking model that is consistent with current understanding.

**Context:** Essentially the multi-cyclic exploratory-iterative-incremental development common to Agile projects. Design activities are integral to each step of that environment – distributed across that environment – the context for this pattern.
**Forces:** The development process is incremental as well as iterative. Agile demands “production ready” software at the end of each iterative cycle, even though that software may not be fully featured. Designs must be equally whole.

**Pattern Description:** Like the name implies this shows how larger and larger version of a whole product / whole design can be created at multiple levels of scale. It creates a structure for the well known design principle that every design – however incomplete is a whole.

**Discussion:**

*Connections*

**Problem:** Designs are made up of multiple parts – they are always a kind of collage. But how do you determine what parts go well with each other?

**Context:** This pattern is applied in conjunction with The Magic of Liminality – as a technique for thinking about all that is possible during the liminal phase of development.

**Forces:** Lots of parts, an exponentially increasing number of potential connections as the number of parts increases.

**Pattern Description:** juxtaposition and free association in a disciplined manner – prototyping and how to effectively use prototypes is included in the pattern.

**Discussion:**

*Prometheus Bound*

**Problem:** No designer has absolute freedom. Some designs cannot be implemented due to technological or scientific limitations. Others are bound by the amount of money or time available from the client. Still others may falter because of the available skills of the development team.

**Context:** This pattern is used in conjunction with the Briefs or Boxers pattern – at the beginning of a development project as input to the designer as she initiates the Magical Liminality pattern.

**Forces:** finding a balance between realism and pessimism / optimism as to what realization space a design must live within.

**Pattern Description:** How to identify real versus imagined constraints – how to use constraints to define a solution space without predetermining the outcome of a design – how to engage in a “dialog” between the design concept and constraints to see if the concept alters the constraints in a useful way, allowing designs that would otherwise be rejected by reflex rather than judgement.

**Discussion:**
**Glossolalia**

**Problem:** The language that one uses is a constraint on their thinking. The problem is a modest form of the Sapir-Whorf hypothesis (once rejected, but currently gaining respect in a less stringent form). This particular constraint is not one that is useful to design – it is, in fact, a highly non-desirable constraint.

**Context:** This pattern is applicable across the design process and influences all design activities.

**Forces:** We have to think about design using a natural language, we have to think about software design using a particular programming language. Languages impose constraints on our thinking or make it difficult if not impossible to elegantly express our design in a given language.

**Pattern Description:** The pattern addresses ways to overcome the limits of language: utilizing multiple languages (hard for U.S. designers, easier for European designers because the respective traditions of being sole English speakers, and the need to be polyglot speakers), finding or creating a common language – usually a graphical language of some sort (like UML is supposed to span any implementation language idiosyncracies), and developing an evocative language of the sort that is supposed to be supported by provocative pattern titles plus story telling.

**Discussion:**

**References**