How to Write a Good Design Pattern

Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice. - Christopher Alexander

For a shortened version of this document, visit the How to Write a Design Pattern FAQ at the Open Source Design Pattern Library site.

How to Write a Good Design Pattern

Who Should Read This Page

This page gives background information on design patterns followed by guidelines and advice on writing helpful and effective patterns. It should be of interest to pattern authors, designers, and solution implementers, as well as anyone else interested in how patterns can contribute to the creation of well-crafted user interfaces.

Background

When the central elements a design are abstracted from a particular solution and found to apply to a number of different solutions, we have a *design pattern*. While patterns in designs can be observed in many domains - from building architecture to object-oriented programming - the Open Source Design Pattern Library will initially focus on user experience patterns. Jennifer Tidwell describes these as "... structural and behavioral features that improve the 'habitability' of ... a user interface" (Tidwell, 2006). Following Tidwell's lead we will concentrate on how patterns relate to the user before considering other actors, such as the designer and the developer.

Where do design patterns come from?

A design pattern usually arises through a combination of creativity and discovery. They are sometimes seen to appear in the wild: a design is observed to have a wider application than its existing implementation, and someone records a description of it in general terms. In this case the pattern author may be considered to be the "first describer" rather than the original creator of the pattern. Christian Crumlish, the curator of Yahoo!'s design pattern library, considers himself to be more of a "pattern detective" than a pattern author.

Sometimes a designer will create a completely new solution to a problem, describe it the general terms of a pattern, and receive credit as its author. In most cases however, designs and implementations will appear before a pattern is codified. This is a natural part of the process, since patterns are expected to be widely proven before their general acceptance.

Design Pattern: A Formal Definition

Formally, the term "design pattern" has been defined as *a **proven** solution to a **common** problem within a **specified context***. Let's explore the details of this definition.

Here, the word *proven* means "tested or tried", with implications of trustworthiness and reliability. A design pattern is proven through assessment by design experts, and demonstrations that effective designs can be derived from it.

The word *common* in our definition simply means "recurring". Design patterns address problems that occur over and over.

Specified context also deserves some explanation. The context description tells us where the pattern works and where it doesn't. When a designer seeks guidance from a design pattern, she must match the context of the problem she is trying to solve with the range of contexts in which the pattern has been shown to be useful.

Elements of a Design Pattern

The core elements of a design pattern are:

- Statements of the **problem** it addresses, and the offered solution.
- The **scope** of a pattern's usefulness is outlined the context description - the situations in which the pattern leads to workable results.
- User interface patterns also include visual **examples** of how implementations based on the pattern will appear to the user.

Context Outline

The context outline describes the set of contexts in which the pattern is effective in responding to what the user needs. It may also indicate situations where the pattern is *not* a useful guide for the creation of designs.

Problem Statement

The problem statement is an expression of what the user needs to do, perceive, or understand.

Solution

The solution statement explains how the user's problem is solved through a clear and understandable set of steps and perceptions. Solutions are often expressed through strong visual metaphors which help the user understand the process taking place.

Illustrations and Examples

A well-expressed design pattern will include examples of derived designs and implementations.

From Patterns to Implementations

How specific are design patterns in the guidance they provide, and how do they relate to other expressions of design? If we view design as a set of refinements from the general to the particular, the information contained in a design pattern is at the most general and most abstract level. The design pattern author seeks to provide guidance to the solution of a general problem: one that may be encountered by users in a variety of situations. Let's consider the different levels at which a design can be expressed:

Design Pattern

The design pattern centres on the user's problem, and how it may occur in a variety of circumstances. The pattern provides a solution outline that employs well-established design principles. It is a model on which a designer can base solutions to specific cases of the problem.

Design

The designer makes design choices within the guidelines set by the design pattern. These are influenced by information that brings specificity to the problem. There will be personas that describe the user's likely intentions and desires as well as use cases that describe the user's objectives. All these serve to limit the scope of the problem to the application context, which should always be compatible with the design pattern context. The design description is user-focused, centring on the user's behaviour in interaction with the service.

Specification

A specification is a precise description of the behaviour of the service in its interactions with the user. Once a specification has been set, there are no user-perceptible design choices left to be made. To put this another way, a specification carries sufficient design information that different implementations – by different implementers – are behaviourally indistinguishable to the user.

Implementation

The implementations of a pattern are the working instantiations of designs based on the pattern.

Writing Up A Design Pattern for the OSDPL

In writing up a design pattern for the Open Source Design Pattern Library, you will want to capture all the essential elements of the pattern as well providing your readers with enough background information for them to use it effectively. The pattern submission process of the OSDPL makes this fairly simple. It presents an input form with tabbed sections that cover both the pattern description and guidelines for its use under the following headings:

1. Properties

In the Properties section, enter the title of your pattern. Try to choose a title that is descriptive, but short enough to be used in conversation. This makes it easy for people to use the pattern name to refer to its functionality.

You can also enter tags or keywords that help classify your pattern for searches.

2. Problem

Here you provide a brief description of the problem from the user's perspective. For example:

> *The user needs to modify a simple text string on the current page, without losing track of the navigational position, or obscuring what is being displayed.*

Note how this statement reflects the user's situation, and frames the context for solution.

3. Solution

The solution section captures details of how the solution can be effected, as well as further clarifying scope and context.

Solution Description

Make a brief statement of the solution in language that could be understood by the user. If a particular visual metaphor is to be used, describe it here.

Solution Image Sources

User interface patterns can often be best expressed with an image or set of images that show what the user will understand as a solution to the problem. Ideally, there will be a primary solution image that can be entered here.

Use When

Here is where the applicable contexts for the solution are described. As a pattern author, you need to explain to the reader the circumstances under which the solution will be most effective.

How

Here you must explain how the interaction design should be structured to best represent the pattern. This section may include a discussion of design choices and trade-offs as well.

Rationale

The rationale section is where you explain why this pattern works. List any applicable design principles, and reference published research or experimental results that justify the use of the pattern. In writing this section you should ask yourself, "what are the advantages of using this pattern?".

4. Examples

In this section you provide references to any examples that will help the reader in understanding the pattern, how it is used elsewhere, and how it may be used in future designs. Here you can insert URLs, upload images, and enter text descriptions.

5. Accessibility

Accessibility is a critical dimension of usability. Address any accessibility concerns in this section, and explain to the reader how to overcome any restrictions or limitations.

6. Relationships

Published patterns rarely exist in isolation. There are often related patterns in other collections, as well as significant implementations of patterns. Enter the URLs of sites to which your pattern bears a relationship. If there is a Fluid Project component based on your pattern, enter a reference to the component description.

Guidelines for Pattern Authors

Writing a pattern requires careful balancing between generality and specificity. Your pattern should be general enough to cover the widest possible area of application, and yet sufficiently specific to ensure that all its inherent design principles are carried through to implementations. Here are some things to keep in mind as you write:

1. ***Draw upon principles and best practices.*** All design decisions implicit and explicit in the pattern should be based on principles derived from an understanding of best practices. The pattern author should quote the underlying principle where it is important for the reader to understand why a particular decision was made.

2. ***Focus on the user.*** A design pattern should be user-centric in its description. Everything in the pattern should be about what the user knows, perceives through the senses, understands, and may intend; as well as the actions the user takes in response to system behaviour.
3. ***Context is critical***. The context of application is critical to the success of any solution derived from a pattern. Martijn van Welie expressed this very clearly when he wrote: "Every

'solution' described in these patterns may succeed in one context but may also fail in another. The challenge is to understand why and how it depends on elements of the context of use." (http://www.welie.com/)

4. ***Seek strong examples***. A pattern description should be supported by solid examples of successful application. To demonstrate generality, a variety of examples exhibiting good design principles should be provided. While usually presented in the language of designs, examples may also be expressed as specifications or actual implementations.
5. ***A pattern is not a design.*** It is important to distinguish between a design pattern and a design. A design pattern reflects proven best practice, serving as a model or starting place for a designer to create a solution. The role of a pattern is to provide design guidance, assisting the designer in making the best possible design choices to satisfy the user's needs.
6. ***Patterns arise from designs.*** Often the first step in creating a design pattern is recognizing commonality across a set of successful designs. Codifying the pattern then consists of identifying and systematically recording the common flow and good design principles expressed in the designs.
7. ***Patterns are models.*** The primary objects in the library are design patterns. They are models from which actual designs (and implementations) can be derived. The author makes certain design choices, based on design principles, and leaves others to the designer and the implementer.
8. ***Patterns can be built from other patterns.*** Patterns can be combined to form other patterns. The proper granularity of a pattern must be considered carefully by the author. Can it be broken down into constituent smaller patterns? Is the pattern you are creating molecular or a composite?
9. ***Remain neutral about intent***. The pattern author should make minimal assumptions about the user's intentions. The specific requirements and intentions of the user are more the domain of the designer, who is likely to be working with a more finely specified problem and restricted scope.
10. ***Consider multiple audiences.*** Patterns should be useful to a variety of audiences. All the viewers of a design pattern should be able to appreciate its cohesiveness and form, and how it employs good design principles. Examples and supplementary material may be directed at specific audiences. For example, images or audio clips drawn from actual designs can be used as illustrations of good practice for designers. Code and markup fragments from existing exemplary implementations may be included for developers. How a pattern manages user workflow, user choices, and user understanding will be of interest to a business analyst.
11. ***Speak to the end user.*** The pattern author should consider the end user as an important part of the audience. One of the objectives of a well-written pattern should be to assist the designer in communicating with the end user
12. ***Patterns can grow to address wider audiences.*** A pattern can be augmented with material addressing a variety of disciplines. The original author may only write for a subset of the potential audiences. Another author may add supplementary material addressing the professional needs of others. Each author and editor must be clear about the audiences they are writing for.
13. ***Terminology must be consistent.*** A pattern description must always be written with attention to consistent and unambiguous terminology. In the OSDPL, a term used in one pattern should have the same meaning when it is used in others. Authors should draw upon and contribute to the OSDPL glossary to help maintain consistency.
14. ***Patterns can be memes.*** A well designed pattern describes a piece of functionality that can be referred to by name without explanation. This provides a grouping of concepts that enriches vocabulary and eases discussions between developers and designers. E.g. "At this point a *progress indicator* will appear." A pattern should aspire to become a meme.
15. ***Naming is important.*** The name of a pattern should reflect its purpose, requiring little explanation for people to understand what it's for.
16. ***Choose metaphors with care.*** If possible, choose clear, familiar metaphors from real-world experiences without relying on terminology. The metaphor you choose may already be a well-established meme in the user's understanding of the world. This can provide tremendous leverage in achieving instant understanding of what is taking place. For example, the image of a file moving between directories on the screen may be a good metaphor for the process of changing a file's point of attachment to the directory tree, even though we know that no data gets moved. The question is whether this is a good way for the user to understand the process in a familiar and comfortable way.
17. ***Patterns Contribute to A Shared Language.*** As the names of patterns are codified, they gain currency as a form of communication between designers, developers, business analysts, and even subject matter experts and end users. In this way, patterns become part of a shared language for discussing user interface design amongst all those who practice it and those affected by it.

Example

A Pattern for a Progress Indicator

We're not going to actually write a pattern for a progress indicator, we're just going to describe some of the things that should be considered in creating a good one.

Problem: The user needs to be made aware that a process is taking place, and progress is being made.

Context: A process has begun. It will take sufficient time that the user would like to know that progress is being made, at what rate, and when it will complete.

Solution In crafting a solution model, the author should think of questions that give insight into the user's perceptions, and the scope and context of the user's situation. Here are some sample questions for consideration:

1. What are the appropriate metaphors for progress: something moving, counting, going through a sequence (alphabetic or numeric), filling up, emptying out, growing or contracting, rising in intensity or volume, climbing the scale, etc?
2. Does the metaphor need to be visual? Would an audible indicator work as well or better?
3. How does each metaphor signal the user of progress being made?
4. What is the appropriate granularity (polling rate, change rate, refresh rate etc)?
5. Should the refresh rate change through time?
6. Can/should we indicate increasing precision as the process progresses?
7. Should there be indication of work accomplished and work remaining?
8. Is it important to indicate the rate of progress as well as the amount accomplished? That is, in addition to distance covered, do we report speed? How about changes in speed (acceleration/deceleration)?
9. Should multiple metaphors be employed for one process? What metaphors work well together?
10. Is it valuable to use depictions of the objects being acted upon by the process (e.g. files uploaded, files scanned, pages printed, spam delivered)?
11. Should the display of progress information be passive and continuous, or should there be alerts and alarms?
12. How can the designer create an association in the user's mind between the appearance of the indicator to the actual process?
13. How can the indicator inspire confidence that the process is in fact progressing and will terminate? This is especially important if accurate estimates cannot be made.
14. How can the indicator inspire confidence that the estimate of remaining time is accurate?
15. How much information is needed? Is a simple accurate prediction of completion time sufficient?
16. How far can a particular design scale? Can we use the same design for a process that is predicted to take seconds as one that takes many minutes? How about processes that may take hours or days?
17. Is the user likely to be engaged in something other than watching the progress indicator - reading mail for example? If so, how does this affect the presentation?
18. Are there things about the state of the process that should be reported - whether it has stalled, what resources it is consuming for example? Are these considerations within the scope of the design pattern?

The pattern author must decide which of these questions is relevant to the pattern, and provide provide meaningful responses. These may simply offer choices to the designer, given the actual design objectives the designer faces. In the full exposition of the pattern the author may include any material that may assist the reader, such as:

- Examples of design solutions based on the pattern. E.g. a well-constructed progress bar - of interest to the designer.
- Code fragments or markup examples showing how parts of the design solutions can be implemented - of interest to the implementer.

A Final Note

It all comes down to best practices. If a pattern conveys best practice in a way that is useful to the designer, and does it in a way that contributes to many solutions, then it is a success.

On this Page

References

- Fluid Open Source Design Pattern Library
- Fluid Summit Presentation - *Allison Bloodworth*
- Designing Interfaces - *Jennifer Tidwell*
- Patterns in Interaction Design - *Martijn van Welie*
- UED-Pattern Library - *Christian Crumlish* (video)
- A Pattern Language - *Christopher Alexander* (Wikipedia)