ACM SIGPLAN Patterns

Language Trends

Editor: *Ron K. Cytron, Dept. of Computer Science, Washington University in St. Louis, MO 63130; cytron@cs.wustl.edu*

Patterns, Architecture and Software

Richard Helm

Introduction

Patterns are a hot topic in the object-oriented software community at present. At the recent OOPLSA'95 conference there were numerous talks, tutorials, workshops and experience reports all focusing on the application of patterns to building software. But what are they, where did they come from, why all the fuss?

Most of the recent interest in patterns sprang from a Birds of Feather session titled "Towards a Software Architecture and Design Handbook" organized by Bruce Anderson at OOPLSA'90 and followed by a workshop of the same name at OOPLSA'91. There Bruce offered the following thought experiment:[1] Imagine yourself seated by your computer at your desk. You have been sweating for many hours on a really tough software design problem. Nothing you can think of works, you reach dead-end after dead-end, and, worse, a critical product deadline looms. Eventually after much frustration, you reach in desperation to your bookshelf, pull down Volume III of *ACM's Handbook of Software Architecture and Design* and casually leaf through it, Suddenly, on a page towards the end of the handbook, you come across an entry that solves your very problem. You read the entry, rapidly implement the solution, and with some relief meet your ship date.

Unfortunately no such handbook exists. But what would it look like if it did? What would the entries be, how would they be organized? What topics would they discuss? Who would write it? All these questions were discussed at this workshops, but what was perhaps more remarkable was that many people's independent threads of investigation into software architecture and design met and started to weave together. Out of this meeting the idea of patterns for software emerged.[2]

The inspiration for the patterns community comes from many sources. If we look at disciplines such as mechanical, electrical and civil engineering or architecture we can find a rich body of experience and knowledge recorded in the form of engineering handbooks, or codes of practice. One of my favorite examples is a Russian mechanical engineering handbook *Mechanisms for Modern Engineering Design* by Ivan Artobolevsky. This five-volume handbook contains thousands of descriptions of mechanical devices. The first two volumes concern just lever mechanisms and contains 2288 entries describing everything from clutches, to governors to aircraft landing gear. The entries are all indexed and each is less than a page long. Note that such a handbook is not a product catalog, nor is it a collection of specifications—none of the entries say anything about materials, dimensions or tolerances. Yet each entry provides enough information to describe what the device is, its key features, and how it works so that a mechanical engineer can simply implement it.

Another source of inspiration is the work of the architect Christopher Alexander and his books *Timeless Way of Building* and *A Pattern Language*. In these books Alexander set out to describe what it is that gives buildings their life, harmony, freedom, comfort, or as Alexander calls it, their "quality without a name". Alexander's answers to this question is recorded as a set of 253 linked patterns forming a pattern language. Patterns such as 112:Entrance Transition, 127: Intimacy Gradient, 159: Light on two sides of every room each describe a problem encountered during the design of, what is the context for this problem and finally its solution. For example 112:Entrance Transition concerns the problem of how to create the entrance to a building. The context is that the entrance of the building does much to shape your impression of the interior, yet you do not want people walk straight off the streets into the main part of the building. The solution described by the pattern, is to create an entrance transition, a space, change of direction or light that clearly indicates the transition from inside to outside. Buildings with entrance transitions allow people to make the mental shift from an outside mode of thinking to an interior mode.

Alexander's work gives rise the current de-facto defi-

[1] This is roughly how I recall it, but I have added some embellishments.

[2] A more complete history may be found at the Portland Patterns Repository (`http://c2.com.ppr/index.html`)

nition of a pattern: A pattern is a solution to a problem in a context. Essentially a pattern tries to capture and distill recurring experience about a domain in a form which makes this knowledge easily acquired by someone who does not have it. Simply reflecting and recording our experiences may seem trivially simple and not a suitably worthy or lofty goal. Yet the lack of software engineering handbooks which document proven software design techniques suggests otherwise. As a software community we have no common vocabulary or language that is rich enough to describe the issues in creating, understanding a system's architecture or design - what are the software equivalents to terms such as trusses, vaults, cantilevers, columns and arches. And although we have notations, these are often intimately linked to the underlying implementation technology, not the problem—to the bricks and mortar, not to the problems to be solved such as spanning a space, or supporting a beam.

Part of the reason why no such equivalents exists for software is that they are hard to create. Software is ephemeral and intangible. In the software world we tend to spend much effort describing descriptions of software (notations), or descriptions of how to create software (methods), but far less effort describing the actual software artifact itself.

The pattern community is trying to do the latter. We try to simply write descriptions of recurring software design structures, techniques and architectures as patterns. Our approach is very pragmatic. We focus on describing real systems and the experiences that we face in our everyday work in the software industry. Our goal is to record and make explicit all the *stuff* that we all know inside our heads but think is either too trivial or too well known to bother writing down—invariably it is neither, and usually far deeper than we imagine.

Patterns have been used to describe different parts of the software development process, including reusable object-oriented designs, team structure and process organization, how to reuse frameworks, and to describe common themes during systems analysis. There are a number of books and papers about patterns, many of these can be accessed directly though the patterns home page.[3]

There is also an annual conference PLoP - Pattern Languages of Programs which provides a forum for people to present their patterns. A unique feature of this conference is that papers are not presented by authors to the audience as they quietly listen, quite the opposite: the audience discusses each paper in front of its author while he or she remains silent. This *Writers Workshop* is very much in keeping with the idea that patterns are not about presenting new ideas, but presenting experience in such a way that readers can absorb it as quickly as possible. There is no better way to see if readers are *getting it* than to hear, directly from them, what they think.

[3] http://st-www.cs.uiuc.edu//users/patterns/patterns.html

Richard Helm (Richard.Helm@msn.com) is a consultant with the object technology practice with IBM Consulting Group/ISSC Australia in Sydney Australia. There he is actively applying patterns to the design of commercial systems. Prior to IBM, Richard was with DMR Group based in Montreal, Quebec, and prior to that he was a research staff member with IBM at the T.J. Watson Research Center in New York. Richard has numerous international publications, is a frequent speaker at international conferences, and is one of the four co-authors of the award-winning book Design Patterns: Elements of Reusable Object-Oriented Software. Richard has a Ph.D. from the University of Melbourne Australia.