Position Paper for Focus Group
"What makes pattern languages work well", EuroPloP 2002

Arno Haase
Arno.Haase@Haase-Consulting.com
Arno.Haase@acm.org

Introduction

This paper is a collection of ideas about pattern languages. They are based on my personal experience and should to date be viewed as neither more nor less than ideas.

Pattern languages as a necessary basis of communication

The key idea of this position paper is that (implicit or explicit) pattern languages are a necessary prerequisite for communication in a software development context - or at least for reasonably effective communication.

In a team programming effort, it is not sufficient for all programmers to know the programming language, but they must rather share a common understanding of how it is and should be used to solve specific problems. It is only this shared knowledge of recurring solutions together with an understanding of why they are applied and what trade-offs and issues are involved that allows team members to discuss their code at the high level that is often necessary. (Programming is the context I am most familiar with, but I assume that in other contexts the situation is similar).

Pattern languages as part of a programming language

In this sense, it is helpful to view pattern languages as part of the programming language that is used for a particular project. That is fairly obvious for common and well-established patterns that become part of next-generation languages (interface in Java, Observer in C#) or are supported by code generators (Proxies for Corba/RMI) or part of core libraries (I/O streams in C++ or Java).

The same is true if patterns are applied in the code, especially if their dynamic behavior is complex. If for example code uses Composite or even Visitor, it is fairly necessary for everyone to know and understand the patterns in order to discuss the code in a meaningful way.

But the design of a system is based on the relationships between patterns as well as separate patterns, and one or more pattern languages are used as part of the common language used to implement the system. While some of these pattern languages are usually of a general nature, they are typically both extended and supplemented in a system specific way. Discussion of such a design requires that all participants share a common understanding of these pattern languages.

Consequences for pattern languages

One of the consequences of this is that a "well-designed" (whatever that is) system is a good source for mining pattern languages, especially if the development team participates in the mining. The resulting pattern languages make the evolving design and domain expertise explicit and reusable for future projects.

Another consequence is that a development effort can profit from making the pattern languages that are used explicit early in the project and refactoring them on an ongoing basis. The following list contains some ideas that I think are helpful in this process.

- *Clear scope*. Instead of having a single pattern language which contains every conceivable aspect of the system and which quickly grows to an unwieldy size, it is better to break it down into several languages with clear scopes and explicit relationships between them.
- *Hierarchical structure*. A pattern languages should address issues at all relevant granularities, from coarse to fine.
- *Good patterns*. Good patterns are obviously at the core of a good pattern language.
- *Overview diagram.* An overview diagram including the relationships between the patterns can serve as a starting point and a reference.
- *Relationships.* A rich set of explicit relationships between the patterns are the blood that lets a pattern language become alive. Possible kinds of relationships include
  - Solution is context: the solution of one pattern is part of the context of another. This includes the common case that one pattern provides support for another.
  - Specialization: one pattern solves a more specific problem than another.
  - Pattern pair: two patterns solve the same general problem but with complementary forces
  - Inverted patterns: Solutions for similar problems that have almost inverse structure [Henney99]

References

[Henney99] K. Henney, *Patterns Inside Out*, Talk presented at Application Development 1999, London