Summary of the open session on pattern selection and pattern repositories

Edited by Aliaksandr Birukou, using comments by Alexander Ernst, Alexandros Karagkasidis, Till Schümmer

During this open session we summarized existing repositories and solutions for pattern selection and then brainstormed on issues in pattern selection and how one can build a new pattern repository to overcome those issues. This summary document first lists the participants of the session, then summarizes existing solutions and issues, and, finally, lists architecture components of the proposed repository and the benefits such repository will provide to pattern authors and users.

Participants

1. Paris Avgeriou
2. Aliaksandr Birukou
3. Alexander Ernst
4. Ed Fernandez
5. Uwe van Heesch
6. Jim Hensman
7. Alexandros Karagkasidis
8. Stefan Sobernig
9. Uwe Zdun
10. Till Schümmer (post-session discussion)

Existing solutions

Pattern repositiories and catalogs.

Some recent examples:

- PatternForge http://www.patternforge.net/wiki/index.php?title=Main_Page
- CMI patterns http://www.cmi-patterns.org/
- J2EE patterns http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html
- Portland Pattern Repository http://c2.com/ppr/
- EAM Patterns (http://eampc-wiki.systemcartography.info)
- PatternSeer (not available anymore, hosted by Microsoft till 2007) - http://www.patternseer.org/

Formal approaches for pattern selection

See works by Paris Avgeriou, Uwe van Heesch, Uwe Zdun and others

Pattern search engines and similar tools

- work by Aliaksandr Birukou and Michael Weiss
- Google Design Pattern custom search http://www.google.com/coop/cse?cx=000531763273211731096%3Ab-lv61obcte.

Raised issues

- Community is important

- How existing pattern repositories are used?
  - Jim told that people contribute and use patterns to their repository called Pattern Language Network (PLANET)
- In some cases, catalogs are not useful when you design an application
  - however, if you have a network of patterns, it can be much easier to explore electronically than in a book
  - more general issue can be raised - “patterns are not useful when you design an application”. May be catalogs/patterns are not useful because people do not use/believe in/understand/have unsuccessful experience with them?
- There is a need for context-aware guidelines
  - the system should know which part of the project you are working on (e.g., security) and helps you to browse only related patterns (e.g., security patterns)
  - another example – the system knows you have applied A, so A is not recommended anymore, but B is, because it usually follows A
  - also links between patterns are guidelines
- Why would people provide feedback?
  - Reasons for providing feedback include but not limited to these:
    - Because they are proud of their way applying the pattern
    - Because they want to thank the author for the advice
    - Because they want to correct the authors
    - Because they want to get rewarded somehow
    - Organizing a focus group at EuroPLoP was mentioned as one of the options. Participants could tag pattern and specify links between them
  - How many people who are really interested in patterns have enough time to provide feedback? Is it a realistic assumption that a_common_developer_under_constant_pressure has enough time to contribute, to provide valuable feedback? This also relates to the issue what is the community.
- What is the target community of the repository?
  - Pattern contributors: Search for the related patterns to avoid reinventing the wheel.
  - Pattern freaks or hobbyists, and students: people just interested in patterns who want to know what’s up in the area. The repository could substitute the search on Amazon or Google or running through all the *PLoPs proceedings as a newcomer.
  - Constant pattern practitioners – the developers who systematically apply the patterns and want to learn more about new patterns.
- Is format a problem?
  - There was a common agreement that you should not provide a uniform format, several formats might and should be supported as long as we know how to search and display patterns using each of the formats.
  - Uploading a single PDF per pattern + author + title would not take much time
  - We should be careful when dealing with patterns that appeared in books
- Possible problems with naming – same name for different pattern, similar (the same?) patterns with different names

    - this is a minor issue
- Different levels of abstractions of patterns
- If possible, the repository should trace pattern sequences
- How to convince practitioners to apply patterns?
    - If we knew, the existing pattern repositories would have been more successful?
- If we include links between forces, developers (or more general: users) can explore the problem space and thereby reach an understanding of available patterns that inform their design of a solution
- We need to reach the understanding of how the process of pattern application looks (or could look) like, i.e. concrete Use Cases. Then we can think about how the repository could support this process.
- Think not only about the design phase
- It is also about the "software engineering culture" and about developers – repository should take into account their skills and practices
- The repository is a tool, and the developers must be protected against wrong application of this tool.

Architectural components

- Repository
- Indexing (of pattern description)
- Search (text-based)
- Metadata + semantics

To the repository you contribute patterns in .pdf or .wiki formats. It would be nice to have there all patterns from Wiley, Addisson-Wesley, but just indexing all *PLoP patterns would be a good start.

Contributions to the repository should have: name, authors, .pdf for each pattern

Optionally, authors can provide the categorization of their patterns as well. The categorization must include many independent criteria (or axes) like layer of abstraction, development phase, type of application (distributed systems, data bases,…), relation to the non-functional requirements, etc. Probably, the POSA I is a good starting point.

Wiki is there to gather metadata from the community:

- Links to related patterns
    - links are forged by users, e.g. "these two patterns are related"
    - some links can be mined from the information about usage of patterns, if we have it
- Sequences
- Associations between patterns (in a graphical way), a pattern map (like in POSA 1 book or J2EE pattern map)
    - we should be aware that it can become extremely unhandy – too large to make any sense. The art is to provide meaningful subgraphs
    - pattern maps should be optional
- Topics (what pattern is about), etc.
- Resolving ambiguities with names

To build the repository we should use Hillside material

Benefits for researchers and writers

- Comments about (using) patterns
- Patterns applicable to my problems
- Patterns related to my research (e.g., CHI patterns)
- ...

Building new repository

- Who is going to use the repository (target audience)?
- We should not do automatic implementation of patterns, instead support he user of patterns in making the “right” design decisions.
- Have concrete use cases about how the process of pattern application looks like.
- The repository should “fit” existing practices/culture of developers.