Community Meeting Notes (Feb 7, 2018) - An introduction to Pattern Languages

Description

Presenters: Simon Bates

An Infusion Pattern Language

Creative User Rights

Notes

Video Recording

Slides (pdf, pptx)

- Pattern language might be good to express design thinking
- Pattern language for Infusion:
    - What could be pattern languages for Infusion?
    - Technical patterns?
    - More scope to do more there.
- Today:
    - What pattern languages are in the broad sense.
        - Pattern languages are an approach to documenting design strategies/solutions in the 1960s.
        - Christopher Alexander
            - book series:
                - The Timeless Way of Building (1979)
                - A Pattern Language (1977)
                - The Oregon Experiment (1975)
            - looks for patterns in time and space
                - cannot separate these - e.g. watching the world go by, sitting on the porch
            - Writing generative patterns – recording them in a way that they can be reused, not just documenting observations.
            - Only interested in patterns with "the quality without a name"
            - Language or structure for making something (e.g. making a porch)
                - Loop through the language and choose patterns useful to what you want to make
                - Construct a mini-language for that situation
        - Richard Gabriel, Patterns of Software
            - "Habitable"; a good place to be
                - Understandable, maintainable, pleasant to work in.
            - PDF available from his website: https://www.dreamsongs.com/index.html ("Patterns of Software")
    - How they are applied to computer software.
        - Kent Beck (1987, "Using Pattern Languages for Object-Oriented Programs")
            - "Keep all of the operations in a method at the same level of abstraction"
                - what does this mean?
                    - Depends on the context of what you are writing the code for. It's matching how you might break the problem down in your head, and having the code structured in a way that matches that.
                    - Consistency of metaphor? How you understand how to break a task into smaller tasks.
        - 1994, "Design Patterns" by the Gang-of-4
            - There is a terminal, closed conversation in the Gang-of-4 patterns where as Alexandrian patterns are "ideas"
            - Paul Graham wrote:

> *When I see patterns in my programs, I consider it a sign of trouble. The shape of a program should reflect only the problem it needs to solve. Any other regularity in the code is a sign, to me at least, that I'm using abstractions that aren't powerful enough-- often that I'm generating by hand the expansions of some macro that I need to write.*

        - anti-pattern, a bad thing.
            - something could either be a pattern or an anti-pattern based on the context. (e.g. Data Transfer Object)
            - Requires too much ongoing work and maintenance over time
        - Computer users should write their own programs. Computer users as "inhabitants" of the space.
            - A way to look at this is that programmers are the "users" here, and the patterns are for the programmers. Another type of "user" is the "end user" and they won't be using these programming patterns. It is these "end users" that more closely resemble the "users" – inhabitants – in Alexander's sense of patterns for cities, towns, neighbourhoods, buildings, etc.
            - Role of "consumer" versus role of "inhabitant" – things are designed for you and you pick and choose whereas as inhabitant you are a contributor to a community; you can't really contribute to something without knowing the shape of it; inhabitants – people as being in the same space as the object they're interacting with.
    - What a pattern language for Infusion might be.
        - Not just patterns of software development code
            - User experience

        - System integration
        - Software design/architecture
        - Construction
- Do we have an articulation of the quality that we want to have, not necessarily what we currently have; what are the kinds of abilities we want people to be able to do with the software which shapes the qualities that we want to have
    - Creative User Rights – create stuff not just use stuff