Patterns of Productive Software Organisations

John Pagonis

17 April 2004, Oslo

Patterns of Productive Software Organisations

- Intro
- What are patterns and pattern languages
- Patterns of Productive Software Organisations
- Discussion

Intro - Why I am doing this

- An excuse to introduce patterns and organisational patterns (at least to some;-)
- Because we’re young and we can learn from others – make our lives easier
- Stimulate interest in making our development organisation rather than just accepting it; and also thinking about improvement.
- Bring to your attention potential solutions to problems that you may be experiencing
- To point out some of the good things you have in common with productive software organisations !!!

What is a pattern ? – ”a solution to a problem in a context”

”**A pattern** is a piece of literature that describes a design *problem* and a *general solution* for the problem in a particular *context*”. as J.O Coplien puts it.

Patterns can be thought of more like *recipes*, rather than plans which can be reverse engineered; our genome is a recipe, as Ward Cunningham puts it.

Patterns capture important empirical design information, thus making up for lapses in our memory.

Also patterns unearth and capture non immediately apparent structure (i.e important constructs which cut across objects in a system).

Some clarifications on patterns

None can become a better *XYZ* by following a method blindly.

- Patterns represent our memory of solutions and our collective experience.
- Human communication is a great problem in software development, patterns provide us with a common vocabulary.
- Patterns are discovered as opposed to designed !
- Patterns are GENERAL solutions, they do NOT solve any problem by themselves. As in any solution there are trade-offs to consider.

An architect, still, has to manipulate constraints and affordances, navigate through forces and identify the context for which to use different patterns and their consequences; in order to come to a satisfactory outcome.

Pattern Form(s)

Patterns are a *literary form* and there is a large variety of pattern forms; like the Alexandrian, GoF, Coplien and Portland forms.

e.g Coplien form;

The pattern name

The problem

The context

The forces

The solution

A rationale

Resulting context

What is a pattern language ?

A pattern language is a collection of patterns that build on each other to generate a system. –Coplien

Pattern languages place individual patterns in context (in a domain) where they are distinguished from their variability.

Patterns in a language form a network , where their links are as important as them and whose distinct number of paths through the language is (usually) very large.

When one follows a path through the language, then he/she can have a complete system build using that language.

....enough

Patterns of Productive Software Organisations

The plethora of these patterns are taken from studies conducted from Lucent Bell Labs over a period of 3 years on 40 highly productive software organisations.

By organisation I mean a software development organisation.

Common Characteristics

Keep Organisation Simple: Fewer roles in the organisation, usually around 16 vs 21 others had

Work flows Inward: The role of the developer is in the centre where information flows from producers of information to consumers of information (developers). The role of the developer is thus supported by the other roles.

Distribute Work Evenly: Don't allow the focus on any developers to become extreme, hence distribute communication around. Even the most central role would not have to deal with disproportionate amounts of communication/work.

Iterate, Iterate! : For these organisations, the traditional waterfall model of software development exists only on paper. Also design and coding were inseparably inter-wined.

Compensate success: With celebrations and other reward structures ;-)

Some of these patterns ...

1. Conway’s Law (organisation follows the architecture)
2. Organisation Follows Location
3. Size the Organisation (e.g 10 people ==60KSLOC in 8 months or 200KSLOC in 15)
4. Few Roles (e.g less than 16 – reduces communication overhead)
5. Patron
6. Engage Customers
7. Self Selecting Team
8. Developers Control Process
9. Lock’em Up Together

10. Upside Down Matrix Management
11. Face to Face Before Working Remotely (i.e PAN split between Cam/London)
12. Architect Also Implements
13. Review the Architecture
14. Group Validation (e.g CRC sessions, team debugging, team review etc)
15. Stand-up Meetings (e.g stand-up room, SCRUM meetings)
16. Engage QA
17. Scenarios Define Problem (e.g XP stories, not design documents ;-)
18. Work Queue (e.g SCRUM backlog, prioritise, planning output is smaller than input)
19. Mercenary Analyst (Documentation is a distraction, and someone is hired to take care of it)

20. Named Stable Bases (a.k.a release and mainline builds)
21. Incremental Integration
22. Early and Regular Delivery (a.k.a release early and release often !)
23. Programming Episodes (e.g XP weekly iterations)
24. Private World (independent team development – i.e dev-branch)
25. Prototype
26. Skunk Works
27. Solo Virtuoso
28. Fire-Walls
29. Gate Keeper
30. Apprentice
31. Day Care (e.g training Boot Camp)

32. Developing in Pairs !!!
33. Sacrifice One Person (to sort out many small distractions)
34. Interrupts Unjam Blocking
35. Don’t Interrupt an Interrupt

Bibliography and References

- *A Generative Development Process Pattern Language*, J. O. Coplien, in Pattern Languages of Program Design, Addison Wesley 1995
- *Patterns of Productive Software Organizations*, `http://www.lucent.com/minds/techjournal/summer_96/paper11/index.html`
- *Software Patterns*, J. O Coplien, SIGS 1996 (out of print) `http://www1.bell-labs.com/user/cope/Patterns/WhitePaper/SoftwarePatterns.pdf`
- *Perfecting Patterns*, A. O Callaghan, Application Development Advisor, vol 5, no.4, May 2001
- *Organization Book* `http://www.bell-labs.com/cgi-user/OrgPatterns/`

`OrgPatterns?OrganizationBook`

- *Organizational Patterns of Agile Software Development - draft* `http://www.easycomp.org/cgi-bin/OrgPatterns?BookOutline`
- *Organizational Patterns of Agile Software Development*, Coplien & Harrison, Prentice-Hall, July 2004
- *Managing the Flow of Technology*,Allen, Thomas. The MIT Press, Massachusetts, 1977.
- *SCRUM: An extension pattern language for hyperproductive software development*, M. Beedle et al. `http://www.controlchaos.com/scrum.pdf`
- *Process Patterns Home Page* `http://www.ambysoft.com/processPatternsPage.html`