Patterns of Productive Software Organisations
(a revisit of the 2001 presentation)

John Pagonis (a.k.a. employee 131)

Menu

- Intro
- What are patterns and pattern languages
- Patterns of productive software organisations
- Discussion

Disclaimer: A lot of the info captured in this presentation is ‘’borrowed’’ from works and personal communication with J. O. Coplien :-).

Intro
(part I)

Why I am doing this

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

Before we start… a word from the wise

A reminder from Fred. P. Brooks ‘’ No Silver Bullet - essence and accidents of software engineering’’,1986

There is NO silver bullet!

The Software Engineering Book Trilogy

‘’The Psychology of Computer Programming’’, Gerald M. Weinberg, 1971

‘’The Mythical Man-Month’’, Frederick P. Brookes Jr, 1974

‘’Organisational Patterns of Agile Software Development’’ , James O. Coplien, Neil B. Harrison, 2004

The Software Engineering Book Trilogy

‘’The Psychology of Computer Programming’’, Gerald M. Weinberg, 1971

Experiences.. about the professional programmer

‘’The Mythical Man-Month’’, Frederick P. Brookes Jr, 1974

Experiences… about the project team (more professional programmers)

‘’Organisational Patterns of Agile Software Development’’ , James O. Coplien, Neil B. Harrison, 2004

Experiences… about the software producing organisation (of teams of programmers)

The Software Engineering Book Trilogy

‘’The Psychology of Computer Programming’’, Gerald M. Weinberg, 1971

Experiences.. about the professional programmer

‘’The Mythical Man-Month’’, Frederick P. Brookes Jr, 1974

Experiences… about the project team (more professional programmers)

‘’Organisational Patterns of Agile Software Development’’ , James O. Coplien, Neil B. Harrison, 2004

Experiences… about the software producing organisation (of teams of programmers)

There is NO substitute for experience!

Patterns and Pattern Languages
(part II)

What is a pattern?

?

What is a pattern?

”a solution to a problem in a context”

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*”**.** - J.O Coplien.

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**

Where did patterns come from?

The concept comes from buildings architecture and in particular from an architect named Christopher Alexander.

The idea was adopted by the software community in the early 90s.

Patterns

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

Patterns

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.

Patterns unearth and capture non immediately apparent ***structure*** (i.e. important constructs which cut across components in a system).

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. The 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.

Remember: there is no silver bullet

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

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.

Organisational Patterns

(part III)

Organisational improvement

To improve an organisation, you must understand it.

e.g., what is software development?

Organisational improvement

To improve an organisation, you must understand it.

Software development is a creative, <u>social</u> activity

(for those that haven’t read those three books:-)

Organisational improvement

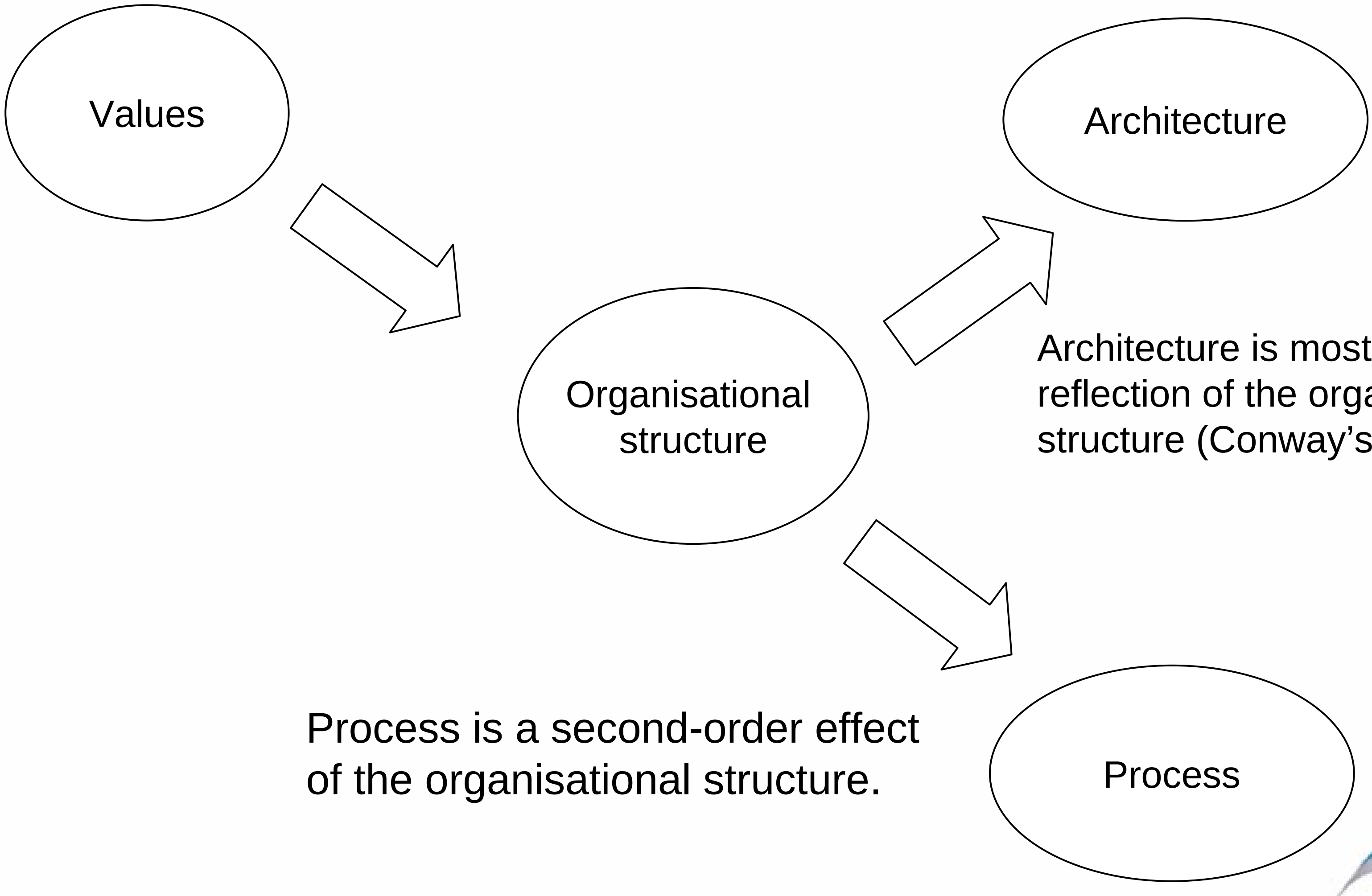
Far too many improvement efforts, focus solely on the development process.

Software development is a creative, social activity that can’t easily be subjected to repeatable process models.

Change for improvement

To improve the process or the architecture, you need to ***change***.

The chain of (organisational) change



Architecture is most often a reflection of the organisational structure (Conway's law)

Process is a second-order effect of the organisational structure.

Change for improvement

To improve the process or the architecture, you need to ***change***.

To change the process or the architecture you need to **change the organisational structure** first!

Structure

What is stable in software development, as in any culture, is the taxonomy of roles.

structure is embodied in roles

Roles and their **relationships,** form the structure of the organisation.

.

Values

It is the structure that makes the organisation what it is.

but

structure comes from deeper values

Since values are notoriously difficult to elicit and to ‘’install’’ in an organisation, it is better to work at the level of structure (which can also enable and attract values).

Organisational patterns

An organisational pattern captures a structure of a successful organisation.

Organisational patterns can be described as organisational structures (or <u>configurations</u> if you prefer) <u>that enable the people within</u> them to be highly productive.

(Structure == roles and their relationships)

Patterns of Productive Software Organisations

- Work started at Bell Labs, in the Pasteur research project circa 1991.
- Initial premise of the project, was that communication is key to effective software development (as a social activity that it is).
- Therefore a view of the social structure, using social network analysis tools is much more appropriate than ISO 9000 techniques and CMM.
- The plethora of these patterns are taken from masses of data collected from studies conducted from Lucent Bell Labs over a period of 3 years on 40 highly productive software organisations.
- For about 12 years, this empirical research, that started by J. O Coplien at Bell Labs, documented and analysed what works in software production.

Beware!

- The organisational patterns in the book are a toolkit
- Not a dogma
- Not a fad
- Not a methodology
- Not a prescription

Organisational patterns capture what has worked before. This knowledge cannot be ignored and is there to be reused.

…from little seeds

We are satisfied by doing real work, Software is like a plant that grows:

You can't predict its exact shape or how big it will grow; you can control its growth only to a limited degree.

There are no rules for this kind of thing – it's never done before.

-- Charlie Anderson, Architect, Borland Quattro Pro for Windows

Common Characteristics (of hyper productive organisations)

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 were 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.

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 and architecture are isomorphic)
2. Organisation follows location (i.e Ronneby/ Quartz/ UIQ)
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 (NASA JPL do that)
8. Developers control process
9. Lock'em up together

Some more…

10. Upside down matrix management
11. Face to face before working remotely (i.e., PAN split between Cambridge/London circa 2000)
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 (stories, not design documents ;-)
18. Work queue (e.g., SCRUM backlog, Lean, planning output is smaller than input)
19. Mercenary analyst (Documentation is a distraction, and someone is hired to take care of it)

Even more…

20. Named Stable Bases (a.k.a. release and mainline builds)
21. Incremental Integration
22. Early and regular delivery
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., Boot Camp)

Still more…

32. Developing in pairs
33. Sacrifice one person (to sort out many small distractions)
34. Interrupts un-jam blocking
35. Don’t interrupt an interrupt
36. Recommitment meeting
37. Completion headroom
38. Take no small slips
39. Team per task
40. Public character
41. Community of trust
42. Unity of purpose

…and even more

…and my favourite is:

<u>Wise Fool</u>

Culture discourages troublemakers

But we grow through painful introspection

Nurture individuals who raise uncomfortable truths

- This is the person who points out that the emperor is wearing no clothes!
- It is dangerous to be a Wise Fool (due to corporate Stalinism)
- In Shakespeare’s King Lear the Fool is the only one who has any sense!

To probe further

- “A Pattern Language: Towns, Buildings, Construction”, Christopher Alexander, 1978
- “The Timeless Way of Building”, Christopher Alexander, 1980
- “Organizational Patterns of Agile Software Development”, James O. Coplien, Neil B. Harrison, 2004
- *Patterns of Productive Software Organizations*, http://www.lucent.com/minds/techjournal/summer96/paper11/index.html
- www.pagonis.org

It’s true…

The End

Goodbye Symbian :-)