Making Agile Processes Scalable

Harald Störrle
University of Munich & FJA GmbH

3.5.2003
International Workshop on
Process Simulation and Modelling
Portland, Oregon

http://www.pst.informatik.uni-muenchen.de/personen/stoerrle/Veroeffentlichungen

Two competing paradigms *comprehensive* and *adaptive*

- Currently, we see two competing paradigms in modeling and managing software:
  - traditional approaches like VM, CMM, SPICE;
  - so called „agile“ approaches like XP, SCRUM, FDD.
- These paradigms have different focus, benefits and problems.
- I call these two *comprehensive* and *adaptive*, since these are their main goals, respectively.

The comprehensive paradigm Characteristics

- Comprehensive („traditional“) models provided a conceptual framework to understand/model processeses in the first place, and manage them at all, for the first time.

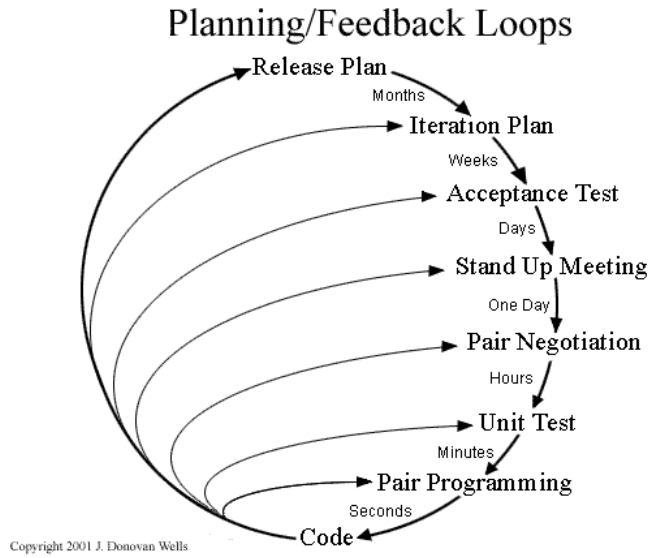
- Comprehensive processes are
  - + efficient under stable conditions
  - + applicable for very large projects
  - + good understanding of process improvement
  - - very large models (steep learning curve, top-down req.‘d)
  - - require tailoring
  - - too unflexible at times
  - - large overhead for small projects and small organizations

The adaptive paradigm Characteristics

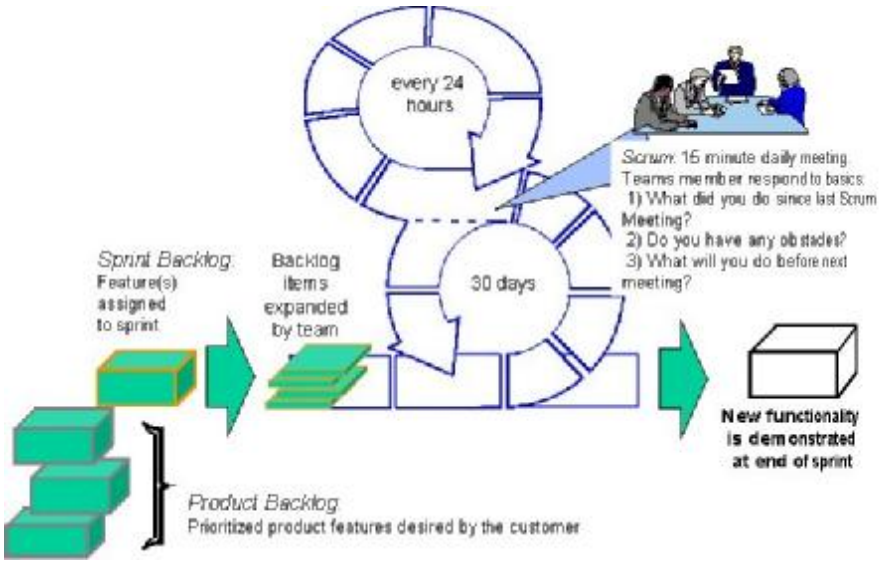
- Adaptive („agile“) models improved management under changing environment conditions (e.g. requirements, technology), but gave away some benefits.

- Adaptive processes are
  - + very responsive to changing environment
  - + easy to spread („sexy“, simple)
  - - unclear notion of improvement
  - - very limited scalability
  - - often just an excuse for hacking

The adaptive paradigm
Feedback in XP



The adaptive paradigm
Feedback in Scrum



Blending the two paradigms

- Can we have the best of both paradigms?
- In particular, can we have an adaptive process
  - that works for non-miniscule projects,
  - that has a useful notion of process improvement,
  - that allows selective usage of different approaches,
  - and thus offers a better cost/benefit-ratio?
- First of all, we need a common framework that is capable of encompassing both paradigms.

Process Patterns
Rationale & Characteristics

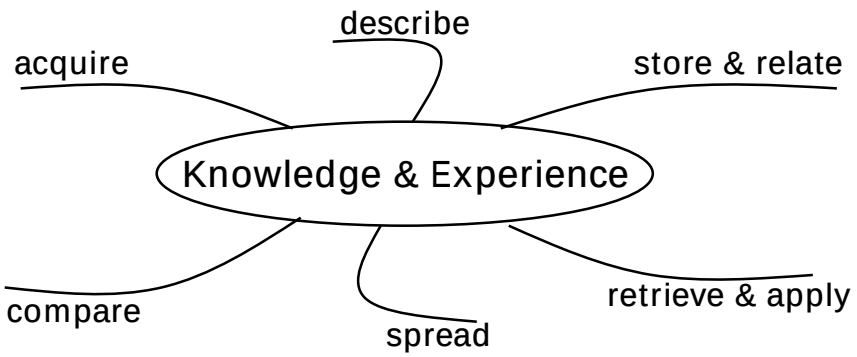
Definition

„A pattern is a practically proven solution to a recurring problem."

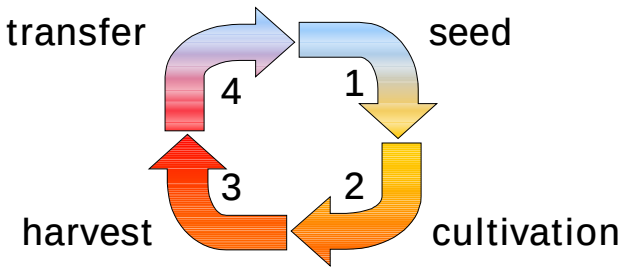
Characteristics

- good cost/benefit-ration (Pareto-principle)
- easy to integrate existing knowledge
- supports internal marketing
- patterns are (as a concept) scale invariant
- conceptual framework to blend both approaches
- patterns always come in families

Dealing with Software Processes is Knowledge Management



Introduction, Improvement & Spreading of Process Patterns



What else, what next?

- Modularity
  - Binding within (process) patterns
  - Coupling between (process) patterns
  - Compositionality - treatable in isolation
- Simulation
  - Could it be done?
  - Would it make sense?

Wrap up

- Process Patterns are
  - structured just like design patterns;
  - good to capture existing knowledge;
  - a unifying framework for both paradigms;
  - help spread knowledge at grassroots level.

www.pst.informatik.uni-muenchen.de/~stoerrle