Software Process Improvement

Peter Dolog
dolog [at] cs [dot] aau [dot] dk
5.2.47
Information Systems
April 7, 2008

Blueprints versus recipes

2002-2003

The Blueprint approach

Externalization

- What is a software process? Where is it?

Separation

- Who designs the software process? How are they designed?

Structuration

- When are software processes designed? Why are they designed?

Improvement by Design threats

Separation of knowledge from use	Externalization of knowledge
Gold-plating process models at the expense of knowledge flow.	Confusing information with knowledge.
Lack of reflective dialogue between process users and designers.	Reifying process knowledge.
Process rollout by distributing process information.	Paying little heed to the role and importance of tacit knowledge.
Focusing on the past and the present and not the future.	Process ossification - downplaying thinking and reasoning.
Standard processes leaving little room for experimentation.	Read it on the intranet - substituting technological contact for human interface.

Eleven Deadly Sins

Not Developing a Working Definition of Knowledge

Emphasizing Knowledge Stock to the Detriment of Knowledge Flow

Viewing Knowledge as Existing Predominantly Outside the Heads of Individuals

Not Understanding that a Fundamental Intermediate Purpose of Managing Knowledge Is to Create Shared Context

Paying Little Heed to the Role and Importance of Tacit Knowledge

Disentangling Knowledge from Its Uses

Downplaying Thinking and Reasoning

Focusing on the Past and the Present and Not the Future

Failing to Recognize the Importance of Experimentation

Substituting Technological Contact for Human Interface

Seeking to Develop Direct Measures of Knowledge

Fahey, L., and Prusak, L. (1998) The eleven deadliest sins of knowledge management. California Management Review, 40, 3, 265-276.

Design: Verb or noun?

Software process design as architecture	Software process design as improvisation
A software process is a blueprint.	A software process is a recipe.
A software process is constructed at a single point in time.	A software process is continually reconstructed.
Software processes produce order through intention.	Software processes produce order through attention.
A software process creates planned change.	A software process codifies unplanned change after the fact.

Inspired by: Weick, K. E. (1993) "Organizational redesign as improvisation." In G.P. Huber and W.H. Glick, Eds., Organizational Change and Redesign, 346-379. New York: Oxford University Press.

Recipes - the concept

As an alternative to Blueprints, we can conceive of software development being supported by recipes—guidelines that we can tailor to specific and shifting conditions.

Using a recipe, process users collectively design the software processes through facilitation, reflection, and improvisation.

Externalization -> Facilitation

Process information

- Methods, procedures, experiences, patterns

Tools

- Enhance capabilities of individuals and teams

Organization & competence

- Facilitate sharing of knowledge (pair programming, collective code ownership, mentoring, ...)

Separation -> Reflection

Engagement

- See activity as a context for learning - use reviews, tracking and project post-mortems

Imagination

- Create shared visions - use project kickoff meetings

Alignment

- Coordinate energies and activities - use standards, policies, and values

Structuration -> Improvisation

Focus on interactions
See everyday projects as experiments
Ensure the ability and latitude to improvise under common standards
Empower teams to manage the unexpected
Encourage creativity through interaction