Welcome to LSO 2005!
Welcome Note and Workshop Overview

Kaiserslautern, April 10, 2005

Welcome to LSO 2005

- LSO 2005 has come back to where it started in 1999!
- First of all: What are your expectations for the LSO 2005 workshop?

What are the assumptions underlying LSO?
- Software development is essentially an intellectual task.
- The success of software organizations is partly shaped by their abilities to make learning happen on individual, team, and organizational levels.
- Therefore software organizations must foster and balance people issues, organization, method, and appropriate tool support.
LSO 2005 Paper Submissions and Review Process

- 27 submissions
  - Europe
  - Americas
  - Asia
- Three to four reviews per paper
- 15 selected for presentation

LSO Workshop Program

- Learning Software Organizations in Action
  (Session I, Monday, 10:30 - 12:30)
- Product Families and Software Development Environments
  (Session II, Monday, 15:00 - 17:00)
- Panel Discussion: Spreading Software Engineering Experience: Through Communities of Practice and Experience Packaging
  (Session III, Monday, 17:30 - 19:00)
- Technology for the Learning Software Organization
  (Session IV, Tuesday, 10:30 - 12:30)
- Future Studies and Novel Approaches
  (Session V, Tuesday, 15:00 - 17:00)
- Panel Discussion: The Future of Learning Software Organizations
  (Session VI, Tuesday 17:30 - 18:30)
Organizational Issues of LSO

- LSO 2005 will focus on discussions. Paper presentations are clustered into topic areas. There will be one overall discussion for all paper presentations of each topic area.
- LSO 2005 will have two sessions with panel discussions.
- On Monday night, LSO participants can meet for dinner to continue discussions and to socialize in an informal setting.
  
  **20:15 at Restaurant „Flammkuche“, Downtown Kaiserslautern**

  Please register in the participants list during Session I.

  (The dinner will not be covered by your conference fees.)

Session I: Learning Software Organizations in Action

- Kick-off Workshops and Project Retrospectives: A good learning software organization practice
  
  *F.F. Fajtak (Siemens)*

- Substance, People, and Tools - Knowledge Management at sd&m
  
  *A. Buch, B. Humm (sd&m)*

- Harvesting Knowledge through a Method Framework in an Electronic Process Guide
  
  *F.O. Bjørnson, T. Stålhane (NTNU)*

- Rightsizing Feedback for Software Team Improvement
  
  *W. Zuser, T. Grechenig (Vienna University of Technology)*

- Discussion
Kickoff Workshops & Project Retrospectives
A good learning software organization practice

Fred Frowin Fajtak
PSE is a software and electronics house of Siemens AG Austria.

PSE contracts for most of the various Siemens divisions worldwide and for a few selected external customers.

PSE offers services in the field of product and system development, system integration and consulting.

PSE sets itself the highest possible standards and requirements regarding product and process know-how.

PSE sales up to € 500 million per fiscal year.

Innovation and continuous improvement are integral parts of PSE culture.
What is a Kick-off Workshop at PSE?

A Kick-off Workshop is a (facilitated) gathering of all persons (in one unit), who are starting to work on defined goals. Typical 1-2 days workshops.

Typical goals of a Kick-off Workshop

**Get the train on the right track!**

- Introduce the *mission statement* by the management
- Explain *goals, requirements and tasks* to the whole team
- Inform the team about *planned dates and milestones*
- Allocate the *responsibilities*
- Clarify *communication structures and interfaces* within the team
- Lay down the mechanisms for *risk management* and the *claim management strategy*
- Give information about *tools*
- Carry out *team building* activities
Different roles in a Kick-off Workshop

For the **project leader**
- it is the moment where he gives the main instructions to the whole team.

For the **management**
- it is the time to delegate the mission to the team.

For the **participants**
- it is the event that makes them feel as a member of a team, in which everyone knows where the journey goes.

For the **facilitator**
- it is the chance to spread project experiences and the art to facilitate without steeling the project leader the show ...

---

Kick-off Workshop impressions

- **Clarifications**
- Interactive work instead of frontal presentations
- Make it a social event
- Commitment
- "Let's go!"

---

Content
- Introduction
- Kickoff Workshops
- Project Retrospectives
- PS E Facilitator
- Network
- Conclusion: Learning Loops

---

Frederik Fajtak
2005-04-11
What is a Project Retrospective?

A Project Retrospective is the facilitated gathering of a project team to systematically learn from experience.


At PSE usually as a 1-2 days workshop

Typical main goals of PSE Project Retrospectives:
- Analyze problems and critical situations
- Analyze success factors and best practice
- Lay down mandatory measures and recorded insights

A typical agenda of a PSE Project Retrospective

Typical Agenda

- **Welcome** and warm-up phase:
  - Bring up the memories of the whole project period
- **Identify core topics**:
  - Problems, puzzles and success factors
- **Prioritize** the topics and build working groups
- **Group work** -
  - Lay down mandatory measures and insights in the plenary
  - Welcome the management for a final discussion
  - Agree on the next steps
Some success factors for Kickoff Workshops and Project Retrospectives

- Spend the **time** for the workshop that it really needs
- Engage a **professional facilitator**
  (contracting, workshop design, facilitation, protocol, …)
- Organize it as an **internal workshop**
- Prefer **interactive work** instead of slide presentations
- Choose a **place** away from the usual place of work
- Concentrate on **technical topics** as well as **social topics**
- Make sure that the outcomes are **mandatory measures** and recorded insights
- Establish Kickoffs and Retrospectives as a **ritual**
- Do workshops not only at the beginning and at the end of a project, but also at the beginning of **project phases**

---

Content
Introduction
Kickoff Workshops
Project Retrospectives
PSE Facilitator Network
Conclusion: Learning Loops
What is a PSE Support Center?

**PSE Support Centers**
- are internal consulting centers
- are specialized on technical topics with high impact on PSE business
- can be used by each PSE project

**The 10 PSE Support Centers**
- Effort Estimation and Metrics
- Components & Internet Technology
- Configuration Management
- Databases
- Object Technologies
- Project Experience
- Project Management
- Test
- Usability
- Windows

---

PSE Support Center Project Experience

**Main goals**
- We support groups to learn from their own experiences
- We spread experiences and solutions

**Staff**
- 2 full time members
- ~30 network members (~20% of their time) of different business units and different locations

- We are experienced Siemens IT guys (developers, project managers, quality managers, consultants, unit managers, ...)
- We are professional workshop facilitators
- Our SC-PE team has facilitated Kickoff Workshops and Project Retrospectives for several hundreds of projects
Conclusion

Learning on several organizational levels

- **Participants** of a Project Retrospective learn from their experiences.
  ⇒ Project Retrospectives ensure that **empirically gained knowledge becomes reusable**.

- **Facilitators** of Project Retrospectives learn a lot about PSE projects and daily practices in other business units.
- **Facilitators in the SC-PE network** share the project experiences of the workshops.
- **Facilitators spread the experiences in their own business units and projects.**
  ⇒ The SC-PE network guarantees a **permanent knowledge exchange** between the business units.

Conclusion: Learning Loops

- The consulting of project leaders and managers during the preparation of a kickoff Workshop by experienced Project Retrospective facilitators is one of several ways to close the learning loop.
- Facilitators tell stories and pass recommendations during the workshops they facilitate. This multiplies the reuse of knowledge, gained through Project Retrospectives.
  ⇒ The SC-PE network has an authentic and efficient way to hand over the collected knowledge to many projects.

- The SC-PE network identifies company-wide patterns of common problems and solutions.
- The SC-PE network launches **PSE best practice reports**.
- The SC-PE insights are input for **PSE quality goals and improvement campaigns**.
  ⇒ The company wide use of SC-PE workshop insights is systematically managed!
**Conclusion**

**Project Retrospectives**
ensure that the empirically gained knowledge gets identified and becomes reusable!

**Kickoff Workshops**
are the most direct way to make sure that empirically gained knowledge is reused!

The company wide network of professional facilitators, working at PSE since 1996, shows that this is good learning software organization practice!

---

**Thank you for your attention!**

**Fred Frowin Fajtak**
Siemens AG Österreich
Program and System Engineering PSE

Werner von Siemens-Platz 1
A-5021 Salzburg

Phone: +43-51707-66242
Mobile: +43-676-4912569
Kickoff Workshops & Project Retrospectives
A good learning software organization practice

Fred Frowin Fajtak
Substance, People and Tools
Knowledge Management at sd&m

Arnim Buch, Bernhard Humm
LSO 2005
Kaiserslautern, 11.04.2005

sd&m AG – software design & management

Area of business
- Development and integration of tailored information systems for business critical processes
- IT consulting with engineering and implementation competence

Customers
- Major companies and organizations achieving a competitive edge by implementing custom solutions

Why Knowledge Management at sd&m?
- Deliver projects on time and in budget by using mature software technology and methods
- Make the team “fit for projects”
- Be prepared for tomorrow’s projects

Company Facts
sd&m AG – software design & management

Key facts 2004
- Staff: 950
- Revenue: € 125 million

Research (R&D)
Shareholder

www.sdm.de
A short history of knowledge management at sd&m

1982 sd&m was founded
- Coffee corners were sufficient knowledge management tools

1997 central technology- & knowledge management
- Knowledge brokers
- Intranet knowledge pages
- Schools
- Skill database

2002 sd&m Research
- Knowledge manager
- Communities
- Community platform,
- Knowledge database

2004 the journey continues
- Innovation management
- Consolidation of grown landscape

In 2004 we painted a „big picture“ of knowledge management at sd&m. It allows to „separate concerns“: substance, people, tools.
Substance is the starting point. Processes and tools can be derived from it.

Three ways to look at substance:
- Change over Time
- What is essential to know?
- Degree of Consolidation

For sd&m, software architecture is essential. The challenge is to deploy it.

**Quasar**
- "quality software architecture"
- basically deals with components, interfaces, software categories
- mature experience, distilled from hundreds of projects since 1995

**How we make it applicable**
- architecture community further develop Quasar
- Quasar principles and architecture tought in sd&m schools
- Reuseable components
- Result: People at sd&m use the Quasar principles, wording and architecture in every day work.
### Software Reuse as „plug and play“ knowledge management?

<table>
<thead>
<tr>
<th></th>
<th>First Approach: Intranet Software Archive</th>
<th>Second Approach: ready-to-go java-components</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What</strong></td>
<td>Open for everybody’s software snippets</td>
<td>Technical components according to quasar architecture (e.g. persistence, authorization)</td>
</tr>
<tr>
<td></td>
<td>Rudimentary check of quality and</td>
<td>Planned, developed and supported by a</td>
</tr>
<tr>
<td></td>
<td>documentation</td>
<td>community</td>
</tr>
<tr>
<td></td>
<td>No support („drop and forget“)</td>
<td>Training via „component school“</td>
</tr>
<tr>
<td><strong>Result</strong></td>
<td>Few contributions</td>
<td>Many projects use the components</td>
</tr>
<tr>
<td></td>
<td>Only one successful reuse: Data</td>
<td>successfully</td>
</tr>
<tr>
<td></td>
<td>warehouse loading tool</td>
<td>community and projects contribute new</td>
</tr>
<tr>
<td></td>
<td></td>
<td>features or even new components</td>
</tr>
<tr>
<td>**Lessons</td>
<td>Critical size: components instead of</td>
<td></td>
</tr>
<tr>
<td>Learned**</td>
<td>snippets</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reusability is not for free and should</td>
<td></td>
</tr>
<tr>
<td></td>
<td>be planned before implementation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Quality must fit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Processes must fit (development, support</td>
<td></td>
</tr>
<tr>
<td></td>
<td>and qualification)</td>
<td></td>
</tr>
</tbody>
</table>

### Example 2

Everybody wants a complete and up to date project database, but nobody likes to fill it.

<table>
<thead>
<tr>
<th></th>
<th>First Approach</th>
<th>Second Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What</strong></td>
<td>Free text questionnaire</td>
<td>Project database (part of knowledge datab.)</td>
</tr>
<tr>
<td></td>
<td>available via intranet search engine</td>
<td>„Project“ holds data, documents, people</td>
</tr>
<tr>
<td></td>
<td>Several approaches to get the database complete: Monthly eMails, project interviews, colleagues collecting project information</td>
<td>taxonomy allows to find projects via what they did</td>
</tr>
<tr>
<td><strong>Result</strong></td>
<td>Poor quantity and quality</td>
<td>1.100 Projects, about 80% well-kept</td>
</tr>
<tr>
<td></td>
<td>Bad search results (via search engine)</td>
<td></td>
</tr>
<tr>
<td>**Lessons</td>
<td>If the goal is completeness you need a controlling process</td>
<td></td>
</tr>
<tr>
<td>Learned**</td>
<td>Make the „knowing“ put the information in the system</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Make the information easily available for the users</td>
<td></td>
</tr>
</tbody>
</table>
From these examples sd&m has learned a lot about knowledge management – and still does.

**Conclusion**

- Focus on substance
- Find appropriate processes for production and distribution of knowledge
- Concentrate on enabling the people to apply it
- Use tools to ease the use of knowledge

**Thank you for your attention**

**Arnim Buch**

0049 – 2102 – 9957 - 937
Harvesting Knowledge through a Method Framework in an Electronic Process Guide

Learning Software Organizations Workshop 2005, Kaiserslautern, Germany

Finn Olav Bjørnson, NTNU
Tor Stålhane, NTNU

Overview

• Research Context
• Method
• Initial Work and Results
• Future Work
Context

- SPI research project in Norwegian industry: SPIKE
- 3 Research Partners
  - NTNU
  - UiO / SIMULA
  - SINTEF
- 11 Industrial Partners

The Company

- Small Software Consultancy Company (17 employees)
- No more than 4-5 employees on any given project
- Most of the employees are working at the site of their customers
- Wish to improve their sharing of ideas and experience
Research Method: Action Research

Diagnosing: Approach

- Interviews with 12 employees
- Main Questions:
  - What was their current approach to knowledge sharing?
  - What information should the new tool contain?
  - What functionality should it provide?
Diagnosing: Results

• Current approach to knowledge sharing:
  – Large degree of informal sharing of experience
  – Easy to get help with technical questions
  – Need for more help with problems related to process
  – Gathering of experience from projects was ad hoc
  – Need for better availability of experience data

Diagnosing: Results

• New framework should contain:
  – Document templates
  – Patterns
  – Practical experience
  – A good development process
  – Hints on how to improve customer relations
Diagnosing: Results

• Functionality:
  – Contain good experiences
  – Easy to find what you are looking for
  – Structured after the development process

Action Planning

• Result of diagnosing was presented
• Plan for improvement:
  – Develop a method framework
  – Develop a dynamic Electronic Process Guide (EPG) coupled to an Experience Repository (ER)
  – Integrate the method framework in the EPG/ER
  – Let the employees fill the framework with their own experiences
Actions Taken

• Developing a method framework
  – One consultant put on the job
  – New process based on Rational Unified Process (RUP) tailored to the company
  – Initial focus is on artefacts
  – Input sought from both employees and scientists
  – Process defined in workshops with key employees
Actions Taken

• Developing a dynamic EPG/ER
  – Strategic partnership with a tool provider
  – Provides tool for process management
  – Company willing to adapt their product based on input from the consultancy company

Evaluating

• Challenges ahead: To keep the ER alive
  – The ER must contain a minimum of experience. With little experience available the developers will neither use it nor contribute to it.
  – The experience found, must be considered relevant. It must help them do a better job and be up to date.
  – It must be possible to establish a community of practice based on the ER. The ER must work as a forum where people can exchange ideas.
Evaluating

• Strategies to meet the challenges:
  – Keep the ER open
    • Everybody can add his or her own experiences
    • One restriction: All information must be traceable to the person who contributed it
  – Build discussion threads in the ER
    • To keep the experience up to date
    • To keep the community of practice alive

Future Work

• Document the process used to create and populate the EPG/ER
• Document the actual use of the EPG/ER
• Compare our findings to published results on the subject of EPG/ER
• Determine how successful the initiative has been for the company and how it might apply to other companies in similar contexts.
Not enough feedback

- Managers do not like to discuss weaknesses of workers directly.
- Workers do not like to hear about their weaknesses.
- Many people overrate their skills enormously.

- Feedback has direct correlations to job satisfaction, performance and motivation.
- Few improvement without feedback.
Reasonable Ways of Feedback

- Self assessment
- Internal team feedback
- External feedback for a team
- Surveys
- Tests
- Checklists
- Reports
- Project data collection

Team Phase Patterns Requiring Feedback

- People driven
  - Team establishment phase pattern
  - Project team crisis phase pattern
- Project driven
  - Project and process status phase pattern
  - Project cut phase pattern
Team establishment phase pattern

- Situation
  - Unawareness of strengths and weaknesses of team members.
  - Missing feedback about team colleagues’ perceptions.
  - Conflicting attraction to roles.
- Applicable Feedback Forms
  - Self assessments
  - Internal team feedback

Project team crisis phase pattern

- Situation
  - Conflicts arise regularly.
  - External mediation by be expensive and time consuming.
  - Mediation by the management may not be objective.
- Applicable Feedback Forms
  - Self assessments
  - Internal team feedback
Project and process status phase pattern

- Situation
  - Management needs feedback to estimate future performance.
  - Most managers concentrate on the hard facts.
  - However, soft facts are important and valuable as well.

- Applicable Feedback Forms
  - Checklists (hard facts)
  - Project data collection (hard facts)
  - Reports (hard facts and soft facts)
  - Surveys (soft facts)

Project cut phase pattern

- Situation
  - Team members may leave team.
  - Customer wants to give final feedback.
  - Many sources: customer, management, team members.

- Applicable Feedback Forms
  - Self assessments
  - Internal team feedback
  - Surveys
  - Project data collections
Summary and Outlook

- Feedback vital for improvement.
- Many feedback forms available.
- Team phase pattern require feedback.

- Tool support desirable.
- Teaching of feedback techniques necessary.
Welcome to LSO 2005!
Session 2

Session II: Product Families and Software Development Environments

- A Community Based Approach for Organizing Software Product Line Evolution
  B. Decker, D. Muthig (Fraunhofer IESE)

- Exploring Communities of Practice for Product Family Engineering
  T.E. Fægri, T. Dingsøyr (SINTEF), L. Jaccheri (NTNU), P. Lago, H. van Vliet
  (Free University of Amsterdam)

- Discussion: Learning product-family organizations

- Systematical Validation of Learning in Agile Software Development Environment
  O. Salo (VTT)

- On the Importance Attributed to Different Knowledge in Software Development Environments
  K. Villela, A.R. Rocha, G. Travassos (Federal University of Rio de Janeiro)

- Discussion: Software development environments
A Community Based Approach For Organizing Software Product Line Evolution

Dipl. Inform. Björn Decker

Introduction: Some words about me

- Organization: Fraunhofer Institute for Experimental Software Engineering
  - www.iese.fraunhofer.de
- Department: Experience Management
  - www.experiencemanagement.de
- My research interests:
  - Knowledge / Experience Management in software product lines
  - Software Engineering Communities / OSS
  - Social Software (i.p. Wikis)
A Community Based Approach For Organizing Software Product Line Evolution

Outline

• Product lines in a nutshell
  - Asset classification
• Motivation: Why communities
• Product line communities
  - Role model
  - Communication patterns
• Summary and future work

Product Lines in a Nutshell

• Domain vs application engineering
• Scoping by products
• Explicit variability & decision
• Defined approach on how to derive products
A Community Based Approach For Organizing Software Product Line Evolution

Asset Classification [Becker]

- Solution assets
  - Work-products of application engineering
    Example: Component used in project
  - Low risk of infrastructure degradation
- Domain assets
  - Reuseable assets,
    Example: Generic component
  - Meta-Assets: Information on how to reuse assets
    Example: Technology description
  - High risk of infrastructure degradation

Motivation: Why Communities

- Product lines affect whole organization
  - Coordinating people across departments (without changing the structure of the organization)
- Iterative introduction is less risky
  - Need for organizational principle that scales
- Maintain infrastructure consistency
  - Combination of technological and social approaches needed
  - Responsibilities to evolve assets have to be assigned to knowledgeable persons

→ Communities offer a lightweight solution to those challenges
What is a PRODUCT LINE Community …

- Definition Community [Wenger]
  - Group of people sharing the same (professional) interest and who interact regularly to learn how to do their jobs better

- Mission of PLC:
  - Support the exchange of experience about product line related issues among members
  - Facilitate the adaptation and evolution of product line practices
  - Introduce new members to product line engineering.

The Three Pillars of Product Line Communities

1. **Research**
   - Define and Improve Product Line Infrastructure
   - Direct feedback (real-world data and artifacts; expert involvement)
   - Up-to-date information;
   - Direct feedback;
   - Top experts from practice

2. **Education**
   - Preparation and Education of Engineers in Product Line Infrastructure
   - Up-to-date information;
   - Direct feedback;
   - Top experts from practice

3. **Production**
   - Development of Applications according to Product Line Infrastructure
   - Highly skilled people move into practice;
   - Direct feedback

   Realistic and approved methods, techniques, tools, and models
A Community Based Approach For Organizing Software Product Line Evolution

Basic Ideas

- Provide career model within community based on contributions
- Distribute assets responsibility based on risk / impact
- Communication patterns support collaboration across organizational boundaries

Inside Product Lines Communities

- Product Line User Community
  - Production & education
  - Members using the same asset build a "sub-community"
  - Identification of members: Artifact usage / variability
- Product Line Support Community
  - Research and education
  - Support and moderate user community when needed
  - Overview of infrastructure
Adapted Role Model [Kim]

<table>
<thead>
<tr>
<th>Sub-Community</th>
<th>Role</th>
<th>Joining Event</th>
<th>Responsibilities</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Visitor</td>
<td>Interest in product line</td>
<td>Determine whether joining makes sense</td>
<td>Read domain asset</td>
</tr>
<tr>
<td>Newbie</td>
<td></td>
<td>Registering: Infrastructure usage</td>
<td>Take part in educational activities</td>
<td>Support by community, educational activities Apply asset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support</td>
<td>Member</td>
<td>Give feedback about usage</td>
<td>Participate in evolution</td>
<td>Synergy effects</td>
</tr>
<tr>
<td></td>
<td>Leader</td>
<td>Responsibility for one (domain) asset</td>
<td>Evolve domain asset Education</td>
<td>Effort for coordination Merits</td>
</tr>
<tr>
<td></td>
<td>Senior</td>
<td>Responsibility for meta asset</td>
<td>Evolve meta asset Research</td>
<td>(More) effort for coordination Merits</td>
</tr>
</tbody>
</table>

Communication Patterns

- Structure communication between product line infrastructure and application project environment
  - Project environment contains instantiated product line infrastructure
- General Idea:
  - Balancing based on impact of asset: control of infrastructure vs. freedom of project
  - Focus communication based on asset usage and variability
- Assign communication pattern to asset
  - Relate and comment
  - Consensus-based
  - Transfer and re-integrate
A Community Based Approach For Organizing Software Product Line Evolution

Communication Pattern: Relate and Comment

- Read-only use of (meta) asset
- User Community gives feedback
- Support Community integrates feedback
- Example: Technology description

Communication Pattern: Consensus-Based

- (Domain) asset changes are performed by User Community based on consensus
- Support Community moderates if needed
- Example: Component used by projects
A Community Based Approach For Organizing Software Product Line Evolution

Communication Pattern: Transfer and Re-Integrate

- Transfer asset to project
- Re-integrate changes after project
- Example: New variation of component

Summary

- Product line communities support iterative introduction of product lines
- Community substructures
  - User and support community
  - Sub-communities based on assets / variabilities used
- Communication procedures structure feedback
Future Work

- Adapt best practices from Open Source development
  - In particular “meritocracy” approach
- Support adaptation
  - Mapping between community roles to standardized product line roles
- Tools / method support
  - Integration of community platform with development environment
  - Define and use community metrics for infrastructure evolution
- Continuing application in industrial projects
  - Issue management in product lines
A Community Based Approach For Organizing Software Product Line Evolution

Sub-Communities : Identification and Communication

Product Line Support Community

- PL Infrastructure
- Issue Mgt
- Adaptation Support / Issue Management

Support Community

- Project Team
- Another Project Team

Product Line User Community

- Variability Doc.
- Identify

Project Team

- Variability Doc.
- Another

Another Project Team

- Variability Doc.
- Application

Adaptation Support / Issue Management

1. Report issue
2. Contact projects working with artifact(s)
3. Moderate and coordinate solution
4. Update

Copyright © Fraunhofer IESE 2005
LSO 2005

Seite 19/17
Exploring Communities of Practice for Product Family Engineering Organizations

Tor Erlend Fægri, Torgeir Dingsøyr: SINTEF, Trondheim
Patricia Lago, Hans van Vliet: Vrije Universiteit, Amsterdam
Letizia Jacheri: NTNU, Trondheim

LSO-2005, April 11. 2005, Kaiserslautern

Overview

- Problem area, background
  - Product Family Engineering
  - PFE and knowledge management
- Enabling techniques, practices
  - Communities of Practice
- How we will do it
  - Study proposal
  - Research plan
- What we are doing
  - Current work
Project context

- SPIKE – Software Process Improvement through Knowledge and Experience

Product Family Engineering

- PFE is a natural step towards industrialised software engineering
  - Promise improved resource economy in the delivery of a set of similar products
- PFE is systematic reuse
  - *Domain engineering* (platforms, reference architectures, common assets, variability management)
  - *Product engineering* (resolves variability to give products, feedback to domain assets)
- PFE lends itself well to an “open market” philosophy in asset development and procurement
  - Also influence the nature of organisations (virtual organisations, flexible collaboration patterns)
  - Adds complexity to managing development processes
Critical aspects of PFE

- Increased flexibility
  - Later binding of requirements to final product
  - Depends upon effective exploitation of existing knowledge in the organisation (speed, accuracy)

- Strategic reuse
  - Depends upon clear communication of vision
  - Demands more complex technical alignment (new organisational level)
  - Increasing emphasis on software maintenance, reduced creative expression

- Specialisation among individuals
  - May cause materialisation of boundaries
  - Lower employee satisfaction

PFE and knowledge management

- Explicit knowledge
  - Product Family Architectures
  - Reference architectures
  - Architectural patterns
  - ....

- Tacit knowledge
  - Domain knowledge
  - Product knowledge
  - Architectural assumptions
  - ....
Communities of Practice

CoP: “Groups of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise in this area by interacting on an ongoing basis”

- Communities as vehicle to disseminate knowledge
  - Tacit knowledge transfer by participation
  - Explicit knowledge transfer by reification
  - Knowledge transfer between communities through brokering or boundary objects
  - CoP is a lightweight, flexible approach with modest costs

Study proposal

- By considering PFs in the following way:

  ![Diagram showing feature models and structural models]

- We seek to investigate effects of CoP
  - Feature and structural models as means of brokering
  - Implementation of features (at PF- and product level)
    - Implementation of features from PF level in products
  - Knowledge transfer from product level back into PF level

- Generally
  - People’s satisfaction with working in reuse oriented software development
  - Effects of CoP activities upon their satisfaction
Research plan

Research questions
- What are the effects of brokering, as defined by Communities of Practice, on the dissemination of knowledge in PFE organizations?
- What reification activities are used and to what effect in PFE organizations?

Research method
- Semi-structured interviews
- Qualitative data analysis

Current work

We are conducting a study at Kongsberg Spacetec to explore CoP in terms of the maintenance of domain and architectural knowledge
- Moderate size, about 50 people in R & D
- Products & services include:
  - Meteorological, environmental and marine surveillance systems and applications
  - Engineering, installation, training, maintenance and support
  - Consultancy studies
- Lightweight PFE approach, extensive use of reusable components but no explicit strategy for platform
- Investigating the exploitation of domain knowledge in product architecture
  - Externalization practices for domain and architecture knowledge
  - Use of brokering activities to stimulate transfer of tacit knowledge
  - Effects of knowledge management practices on the development of new product architectures
Next

- Investigations of CoP in other, possibly larger, companies
Background on Agile SPI

"At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly." (Agile Principle)

"Welcome changes during the process" (Agile Principle)

"Build projects around motivated individuals" (Agile Principle)

"The best architectures, requirements, and designs emerge from self-organizing teams" (Agile Principle)

"Each situation calls for a different methodology" (A. Cockburn, Agile Software Development, 2002)

Just rules (Practice of XP)

Incremental change (Basic Principle of XP)
Research Topic:
SPI in Agile Software Development Context

Focus of this paper:
- What can the *iterative team reflection* provide organizational level with?
- What does the organizational level need to know in order to utilize the cumulated process learning/process knowledge of individual project teams?
- How can this knowledge be transferred?

Motivation for Learning from Teams to Organization

The Experiences of the People who Actually Execute a Process can be one of the Most Important Sources of Input to a Process Improvement (J. W. Vanseville)

New Ideas are Essential if Learning is to Take Place (David A. Garvin)

Raw unfiltered Information is of limited value (David A. Garvin)

Knowledge is only Maximized if the Organization can Learn (Michael J. Earli)

Knowledge must be shared collectively, rather than limited to a privileged few (David A. Garvin)

Purely local knowledge is valuable, but it does not mark the existence of an organization that has learned (David A. Garvin)
Research Context

- 1st - 5th of ongoing series of Agile software development case studies at VTT

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>eXpert</th>
<th>zOmbie</th>
<th>bAmbie</th>
<th>uniCorn</th>
<th>Bubble</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of project (in person months)</td>
<td>8.5</td>
<td>10</td>
<td>5.5</td>
<td>9.1</td>
<td>7.1</td>
</tr>
<tr>
<td>End product</td>
<td>Intranet app</td>
<td>Mobile service</td>
<td>Mobile extension to an existing information system</td>
<td>Mobile extension to an existing information system</td>
<td>Mobile extension to an existing information system</td>
</tr>
<tr>
<td>Duration</td>
<td>9 weeks</td>
<td>9 weeks</td>
<td>9 weeks</td>
<td>11 weeks</td>
<td>8 weeks</td>
</tr>
<tr>
<td>Iteration Length</td>
<td>3x two weeks, 3x one week</td>
<td>1x one week, 3x two weeks, 2x one week</td>
<td>1x one week, 3x two weeks, 2x one week</td>
<td>2x two weeks, 7x one week</td>
<td>1x one week, 3x two weeks, 1x one week</td>
</tr>
<tr>
<td>Team size</td>
<td>3-4</td>
<td>5-6</td>
<td>4</td>
<td>4-5</td>
<td>4-6</td>
</tr>
</tbody>
</table>

SPI in Agile Software Development Case Studies

- **Post-Iteration Workshops:**
  - to ease up and improve the daily work of software developers
  - to provide organizational level with concrete & validated process knowledge
    "bottom-up" to improve organizational development of mobile software (Mobile-D1)
- **Post-Project Workshops:**
  - to find organizational improvement opportunities
- **Quality-Team Meetings (QM):**
  - to handle the process knowledge of project teams

---

Background

Lightweight postmortem review technique¹

Reflection workshop technique²

Post-Iteration Workshops


Process of Post-Iteration Workshops ¹

1) Preparation
2) Experience Collection
3) Ranking of Improvement Actions
4) Fostering
5) Follow-up and Validation
6) Packaging

¹ O. Salo and P. Abrahamsson: A Post-Iteration Workshop Approach for Agile Software Process Improvement: Implications from a Multiple Case Study (currently under review process)
Knowledge Flows from PIW’s to Organization

Structured AP List was needed for:

- **KJ method** used to group the experiences generated by the project team (suggested in postmortem review technique)
- Traceability between flap-sheet & AP list
- Open discussion lead by the facilitator used to
  - generate issues guided by the AP template

<table>
<thead>
<tr>
<th>Problem Area: TDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem</td>
</tr>
<tr>
<td>Why is the improvement needed?</td>
</tr>
</tbody>
</table>
Experiences of team members from 1st iteration (1st PIW) on "planning and estimation"

Task times: approximately only 50% of time used

Stories: tasks should be more accurate / comprehensive

Team overestimated work.

1st step of improvement: more knowledge needed

In the 2nd PIW

Interpret the metrics data in the next PIW to see if effort estimations improved.

Task will be split to max. 4 hours size.

More effort is allocated for defining tasks during planning game.

Analysis of effort estimations

Too big and indistinct tasks.

Task estimations too inaccurate in release 1.

Analyse task data to evaluate the cause of inaccuracy in each task.

Project Manager

Interpret the metrics data in the next PIW to see if effort estimations improved.

Project Manager

Project Team

Tracker

Team interprets the data in the next PIW to find ways to improve.

Data from 1st and 2nd iteration revealed too big and indistinct tasks.

Team interprets the data in the next PIW to find ways to improve.

Tracker

Analysed task data to evaluate the cause of inaccuracy in each task.

Analysis of effort estimations

Validation plan

Actor

Problem: Task estimations too inaccurate in release 1.

Action Point: Analyse task data to evaluate the cause of inaccuracy in each task.

Actor: Tracker

Validation plan

Actor: Tracker

Actor: Project Manager

Actor: Project Manager

Actor: Project Team

Actor: Tracker

Actor: Project Manager

Actor: Project Manager

Actor: Tracker

Actor: Project Manager

Actor: Project Team

Actor: Tracker

Validated process

Knowledge

Problem Area: Planning and Estimation

Experiences of team members from 1st iteration (1st PIW) on "planning and estimation"

Task times: approximately only 50% of time used

Stories: tasks should be more accurate / comprehensive

Team overestimated work.

1st step of improvement: more knowledge needed

In the 2nd PIW

Interpret the metrics data in the next PIW to see if effort estimations improved.

Task will be split to max. 4 hours size.

More effort is allocated for defining tasks during planning game.

Analysis of effort estimations

Too big and indistinct tasks.

Task estimations too inaccurate in release 1.

Analyse task data to evaluate the cause of inaccuracy in each task.

Actor: Tracker

Validation plan

Actor: Tracker

Actor: Project Manager

Actor: Project Manager

Actor: Project Team

Actor: Tracker

Actor: Project Manager

Actor: Project Manager

Actor: Tracker

Actor: Project Manager

Actor: Project Team

Actor: Tracker

Validated process

Knowledge

Problem: Task estimations too inaccurate in release 1.

Action Point: Analyse task data to evaluate the cause of inaccuracy in each task.

Actor: Tracker

Validation plan

Actor: Tracker

Actor: Project Manager

Actor: Project Manager

Actor: Project Team

Actor: Tracker

Actor: Project Manager

Actor: Project Manager

Actor: Tracker

Actor: Project Manager

Actor: Project Team

Actor: Tracker

Implications from the Case Projects

Average Estimation Accuracy

Average Estimated Task Size

Minutes

Iteration 1 2 3 4 5

-100 -50 0 50 100 150 200 250 300 350 400

Implications from the Case Projects

Knowledge
### Implications from the Case Projects: SPI in Project Level

- Organizational support and decision making was needed in majority of project level SPI actions during the projects.
- Organizational level was able to participate in SPI action taking due to AP lists.

<table>
<thead>
<tr>
<th>Project</th>
<th>SPI Actions with Org. support</th>
<th>SPI Actions without Org. support</th>
</tr>
</thead>
<tbody>
<tr>
<td>cXpert</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>zOmbie</td>
<td>16</td>
<td>9</td>
</tr>
<tr>
<td>bAmbie</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>uniCorn</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Bubble</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>47</strong></td>
<td><strong>23</strong></td>
</tr>
</tbody>
</table>

### Implications from the Case Projects: SPI in Organizational Level

- Project level process knowledge played an important role in organizational SPI.
- Project level process knowledge (validated SPI actions of project teams / improvement opportunities) was used in evolving MOBILE-D software development process.
- A selection of validated improvements was included in Mobile-D as such, potential improvement opportunities were piloted in later projects.

<table>
<thead>
<tr>
<th>Quality Team</th>
<th>cXpert</th>
<th>zOmbie</th>
<th>bAmbie</th>
<th>uniCorn</th>
<th>Bubble</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SPI actions</strong></td>
<td>-</td>
<td>-</td>
<td>17</td>
<td>36</td>
<td>26</td>
</tr>
<tr>
<td><strong>Other actions</strong></td>
<td>-</td>
<td>-</td>
<td>8</td>
<td>21</td>
<td>4</td>
</tr>
</tbody>
</table>
Conclusions & Shortcomings

• Team level agile learning/SPI provides organizational level with a large amount of valuable process knowledge

• The process knowledge of project teams can be iteratively validated during the project in order to increase the value of "bottom-up" learning in organization

• Tool support is needed for storing transferring and filtering the data, information, and knowledge from the team reflections of multiple projects effectively to organizational level, including e.g.:
  • Common classification for the "problem areas"
  • Storage and links to the validation data (e.g. illustrations of data analysis)
  • Context information on the project environment
  • Different needs and viewpoints on the process knowledge

• Organizational structures for organizational learning should be established to enable effective analysis & dissemination of the process knowledge in a "bottom-up" manner (e.g., in organizational processes)

Thank You!

Questions?
On the Importance Attributed to Different Knowledge in Software Development Environments

Karina Villela,
Guilherme Travassos,
Ana Regina Rocha

Overview

- Context
- Problem
- Survey Description
  - Definition
  - Sample Selection
  - Instrumentation
  - Data Analysis
  - Results
  - Interpretation of Results
- Conclusion
Context

Knowledge Management (KM)
Software Development Environment (SDE)
Enterprise Oriented Software Development Environment (EOSDE)

Problem

- Many different types of knowledge are produced and used during software development and maintenance
- It is important to administrate this knowledge creatively, productively and effectively

A fundamental step towards the successful introduction of Knowledge Management in software development environments is the identification and prioritization of the required knowledge
**Survey Definition**

*Analyse* the organizational environments for software development and maintenance

*For the purpose of* characterizing the knowledge mentioned in technical literature on Software Engineering

*With respect to* perceived importance in terms of the need to have the knowledge available

*From the point of view of* programmers, analysts, project managers and company directors

*In the context of* software projects carried out in the year 2002 by companies located in Salvador

---

**Survey Sample**

- **Target Population**
  - 472 people - according to figures from the Brazilian Ministry of Work

- **Sample Selection**
  - 99 programmers
  - 203 analysts
  - 61 managers
  - 30 directors
  - 393 people

- **Final Sample**
  - 41 programmers
  - 70 analysts
  - 32 managers
  - 10 directors
  - 153 people
Survey Instrumentation

Set of Questionnaires – Internet Site

Screen for Participant Identification

Survey Instrumentation

Set of Questionnaires – Internet Site

Screen with Filling Instructions
Survey Instrumentation

- Set of Questionnaires – Internet Site

Screen for Project Description

Screen for Collecting Importance of SE Experience
Survey Instrumentation

Terms Explained in Footnotes

- System Engineering
- Product and/or Service Supplier
- Technical Partner
- Mission of the Organization
- Company Software Processes
- Software Item
- Best Practices
- Software Reverse Engineering
- Software Reengineering
- Problem Solving Process
- Lessons Learned

Survey Results

<table>
<thead>
<tr>
<th>Type of Knowledge</th>
<th>Best Practices</th>
<th>Application Domain</th>
<th>Organization</th>
<th>Technical Literature</th>
<th>Historical Data</th>
<th>Clients</th>
<th>Technical Partners</th>
<th>Suppliers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Company best practices for coding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Company best practices for tests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Company best practices for software documentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Company best practices for quality assurance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Knowledge about the company software processes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Company best practices for the problem solving process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Company best practices for specification and analysis of software requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Company best practices for software design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Knowledge about the application domain</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Company best practices for project management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Company best practices for system reengineering</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Templates of documents, including real examples of their use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Company best practices for the evaluation and improvement of software processes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Software items such as specifications, system architecture, source code and test cases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Company best practices for process modeling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Company best practices for software maintenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Knowledge about the evaluation of the company software processes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Knowledge about the distribution of competences among the company professionals</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Reports on the best practices in the software industry</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Knowledge about organizational objectives and goals</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Company best practices for training</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Company historical data related to project management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Knowledge about the types of software developed in the company</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Company best practices for reuse of software items</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Best Practices
- Software Items and others
- Application Domain
- Organization
- Technical Literature
- Historical Data
- Clients
- Technical Partners
- Suppliers
Survey Results

Type of Knowledge

1. Company best practices for coding
2. Company best practices for tests
3. Company best practices for software documentation
4. Company best practices for quality assurance
5. Knowledge about the company software processes
6. Company best practices for the problem solving process
7. Company best practices for specification and analysis of software requirements
8. Company best practices for software design
9. Knowledge about the application domain
10. Company best practices for project management
11. Company best practices for system engineering
12. Templates of documents, including real examples of their use
13. Company best practices for the evaluation and improvement of software processes
14. Software items such as specifications, system architecture, source code, and test cases
15. Company best practices for process modeling
16. Company best practices for software maintenance
17. Knowledge about the evaluation of the company software processes
18. Knowledge about the distribution of competencies among the company professionals
19. Reports on the best practices in the software industry
20. Knowledge about organizational objectives and goals
21. Company best practices for training
22. Company historical data related to project management
23. Company best practices for system operation and support to users
24. Knowledge about the types of software developed in the company
25. Company best practices for reuse of software items
### Survey Results

#### Type of Knowledge

1. Company best practices for coding
2. Company best practices for tests
3. Company best practices for software documentation
4. Company best practices for quality assurance
5. Knowledge about the company software processes
6. Company best practices for the problem solving process
7. Company best practices for specification and analysis of software requirements
8. Company best practices for software design
9. Knowledge about the application domain
10. Company best practices for project management
11. Company best practices for system engineering
12. Templates of documents, including real examples of their use
13. Company best practices for the evaluation and improvement of software processes
14. Software items such as specifications, system architecture, source code and test cases
15. Company best practices for process modelling
16. Company best practices for software maintenance
17. Knowledge about the evaluation of the company software processes
18. Knowledge about the distribution of competencies among the company professionals
19. Reports on the best practices in the software industry
20. Knowledge about organizational objectives and goals
21. Company best practices for training
22. Company historical data related to project management
23. Company best practices for system operation and support to users
24. Knowledge about the types of software developed in the company
25. Company best practices for reuse of software items

#### Best Practices
- Software Items and others
- Application Domain
- Organization
- Technical Literature
- Historical Data
- Clients
- Technical Partners
- Suppliers
### Survey Results

| Type of Knowledge |  
|-------------------|---
| 1k.                | Company best practices for coding |
| 1x.                | Company best practices for tests |
| 1y.                | Company best practices for software documentation |
| 1z.                | Company best practices for quality assurance |
| 2k.                | Knowledge about the company software processes |
| 2x.                | Company best practices for the problem solving process |
| 2y.                | Company best practices for specification and analysis of software requirements |
| 2z.                | Company best practices for software design |
| 3k.                | Knowledge about the application domain |
| 3x.                | Company best practices for project management |
| 3y.                | Company best practices for system engineering |
| 3z.                | Templates of documents, including real examples of their use |
| 4k.                | Company best practices for the evaluation and improvement of software processes |
| 4x.                | Software items such as specifications, system architectures, source code and test cases |
| 4y.                | Knowledge about the distribution of competencies among the company professionals |
| 4z.                | Knowledge about the distribution of software processes |
| 5k.                | Knowledge about the application domain |
| 5x.                | Company best practices for project management |
| 5y.                | Company best practices for system engineering |
| 5z.                | Templates of documents, including real examples of their use |
| 6k.                | Company best practices for the evaluation and improvement of software processes |
| 6x.                | Software items such as specifications, system architectures, source code and test cases |
| 6y.                | Knowledge about the distribution of competencies among the company professionals |
| 6z.                | Knowledge about the distribution of software processes |
| 7k.                | Knowledge about the application domain |
| 7x.                | Company best practices for project management |
| 7y.                | Company best practices for system engineering |
| 7z.                | Templates of documents, including real examples of their use |
| 8k.                | Reports on the best practices in the software industry |
| 8x.                | Knowledge about organizational objectives and goals |
| 8y.                | Company best practices for training |
| 8z.                | Company historical data related to project management |
| 9k.                | Company best practices for system operation and support to users |
| 9x.                | Knowledge about the types of software developed in the company |
| 9y.                | Company best practices for reuse of software items |

- **Best Practices**
  - Software Items and others
  - Application Domain
  - Organization
  - Technical Literature
  - Historical Data
  - Clients
  - Technical Partners
  - Suppliers
## Survey Results

<table>
<thead>
<tr>
<th>Type of Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>151</td>
</tr>
<tr>
<td>152</td>
</tr>
<tr>
<td>153</td>
</tr>
<tr>
<td>154</td>
</tr>
<tr>
<td>155</td>
</tr>
<tr>
<td>156</td>
</tr>
<tr>
<td>157</td>
</tr>
<tr>
<td>158</td>
</tr>
<tr>
<td>159</td>
</tr>
<tr>
<td>160</td>
</tr>
<tr>
<td>161</td>
</tr>
<tr>
<td>162</td>
</tr>
<tr>
<td>163</td>
</tr>
<tr>
<td>164</td>
</tr>
<tr>
<td>165</td>
</tr>
</tbody>
</table>

## Best Practices
- Software Items and others
- Application Domain
- Organization Technical Literature
- Historical Data
- Clients
- Technical Partners
- Suppliers
Interpretation of Survey Results

- Knowledge related to software product engineering were considered more important than knowledge related to project management

- Best practices were evaluated highly because they represent knowledge already consolidated

- Systems analysts and project managers considered knowledge about risk management as very important, while many groups of directors did not

General Order of Importance

- Best practices
- Knowledge about the application domain
- Description of the company software process, templates, examples and software items
- Knowledge about the company
- Knowledge from technical literature
- Historical data
- Knowledge about clients
- Knowledge about technical partners
- Knowledge about suppliers
Conclusion

- The results are of practical importance to Learning Software Organizations in Salvador
- They should not been seen as prescriptive
- They have been used to guide the elaboration and incorporation of content and the development of tools for the Taba workstation. This allows the configuration of Enterprise Oriented Software Development Environments for different companies.
Welcome to LSO 2005!
Session 3

Kaiserslautern, April 10, 2005

Panel Discussion: Spreading Software Engineering Experience –
Through Communities of Practice and Experience Packaging

- Introduction to Panel Theme
- Four Panel Statements
  - Outi Salo
  - Markus Nick
  - Marta Cimitile
  - Patrick Waterson
- Open Discussion
Panel Theme: Communities of Practice and Experience
Packaging for Spreading Experience

Two basic strategies to implement a learning software organization:

- Communities of Practice: Bring together a group of people and make them share their experience
- Experience Packaging: Document experience in some kind of artifact (report, databases, intranet portal etc.) and make people read and re-use the experience.

Both strategies cover a variety of different techniques.
They have different strengths and weaknesses.

What is a community of practice?

"Groups of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise in this area by interacting on an ongoing basis"
The History of Experience Packaging
- A Rough Guide

1980  1990  2000

- Code Reuse

Experience Factory & Comprehensive Reuse

The Experience Factory at the NASA GSFC Software Engineering Laboratory

Error Rates (development)

- Early Baseline: 8 similar systems
- Current: 7 similar systems

- Early Baseline: 4 projects
- Current: 4 projects

Cost (staff months)

- Early Baseline: ~490 staff months
- Current: ~210 staff months

- Early Baseline: 98 staff months
- Current: 277 staff months

Reuse

- Early Baseline: 8 similar systems
- Current: 8 similar systems

- FORTRAN: 3 systems
- Ada: 5 systems

- Average: ~79%
- IEEE: ~20%

- Decreased 75%
- Reduced 55%
- Increased 300%
The Experience Factory (Basili, Rombach, 1989)

**Project Organization**

1. Characterize
2. Set Goals
3. Choose Process

**Execution plans**

4. Execute Process

**Experience**

- Environment characteristics, customizable knowledge
- Project Support
- Experience Base
- Project analysis, process modification data, lessons learned
- Analyze

**Code Reuse**

- Experience Factory & Comprehensive Reuse
- Pattern Repositories
- SE Information Repositories
- Good Practice Repositories

The History of Experience Packaging - A Rough Guide

<table>
<thead>
<tr>
<th>Year</th>
<th>Code Reuse</th>
<th>Experience Factory &amp; Comprehensive Reuse</th>
<th>Pattern Repositories</th>
<th>SE Information Repositories</th>
<th>Good Practice Repositories</th>
</tr>
</thead>
<tbody>
<tr>
<td>1980</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Panel Theme: Communities of Practice and Experience Packaging for Spreading Experience

Questions arising in the context of experience sharing:

- What techniques do you find most important for spreading SE experience throughout a software organization?
- What would be the major fault when implementing a learning software organization?
- How would you combine communities of practice and experience packaging to establish a highly effective learning software organization?

Panel Discussion: Spreading Software Engineering Experience – Through Communities of Practice and Experience Packaging

- Introduction to Panel Theme
- Four Panel Statements
  - Outi Salo
  - Markus Nick
  - Marta Cimitile
  - Richard Waterson
- Open Discussion
In Agile Software Development...

- The software process improvement focus is on learning and process knowledge of software development teams (iterative team reflection workshops)

  "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts is behavior accordingly". (Agile Principle)

- Short software development iterations provide an opportunity to validate the learning (process improvements) during the project
- A large amount of valuable & validated process improvement knowledge can be created iteratively in individual Agile projects
- A huge potential is available for "Agile organizational learning" in a "bottom up" manner
• What would be a major fault when implementing a learning agile software organization?
  • Not establishing organizational structures and practices for learning "bottom-up" => team learning not utilized in organizational level
  • Not understanding the importance of two-way collaboration of project teams and organization during the knowledge creation process

• How would you combine communities of practice and experience packaging to establish a highly effective learning agile software organization?
  • Conducting iterative team reflection workshops in agile software development projects (including iterative validation of learning)
  • Systematic and structured tool supported documentation of team reflections for the purposes of both project team (guidance & follow-up) and organization (knowledge transfer, filtering, and analysis)

• What techniques do you find most important for spreading SE experience throughout a software organization?
  • Effective filtering, analysis & action taking on organizational level
  • Embedding learning into organizational memory (e.g., process descriptions & training)
Hypothesis: Experience Usage ⊆ Experience Maintenance

- Idea "Everybody does packaging and repackaging!"
- Idea "Closed loop": Users maintain experiences (their own and of others)
  - Users add new experiences, EF reviews
  - Users update experiences, EF reviews
  - Standardization of evaluation and maintenance
    - Evaluation through user feedback
    - Maintenance triggered by user feedback
    - Tight integration with day2day work processes and tools
- Result: Users do part of EF's job and are more satisfied ☺
„Spreading SE Experience through CoPs and experience packaging“ – Panel Contribution

LSO 2005 – April 11th 2005

Dr. Patrick Waterson
Fraunhofer Institut Experimentelles Software Engineering (IESE)
Tel.: 06301-707 220

Copyright © Fraunhofer IESE 2005

Barriers and hindrances in implementing a learning software organisation

• Many barriers and hindrances are well understood and researched
  - Reluctance to share knowledge, lack of trust in people and systems
  - Ease of use, relevance of information
  - Lack of a collaborative culture etc.
• Some less so
  - What motivates people to use CoPs and social software (weblogs, wikis)?
  - Various answers:
    • Initial motivation is often personal benefit („I need to find an answer to X“)
    • Once a CoP is more established, then members are more willing to help one another (share information, give-and-take)
    • Success of the CoP depends on initial altruism, reciprocity and support from others in general
  - The most frequent problems are often the most basic
    • Lack of initial content
    • Initial bursts of enthusiasm followed by apathy and decline
    • Motivating people to keep on participating, contributing
Possible solutions and ways forward

• Balancing social (i.e., human) and technical concerns
  - „Kick-starting“ the community with relevant content
  - Staging implementation and conducting iterative evaluations
  - Moderation, mentorship

• Some promising directions (e.g., Ras et al., 2005)
  - Use of Wiki technologies, integrating learning and KM activities
  - Advantages: smaller than CoPs, members can tailor functionality, easy to implement and extend
  - Supporting an experience-based information system using (individual) weblogs and project (work group) reports:
    - Experience packages are enriched by learning elements (e.g., term definitions, cases) and support learning

Welcome to LSO 2005!
Session 4

Kaiserslautern, April 10, 2005

Session IV: Technology for the Learning Software Organization

- Experience-based Support for Code Inspections
  M. Nick, C. Denger, T. Willrich (Fraunhofer IESE)

- Using a Software Development Environment with Knowledge Management to Support Deploying Software Processes in Small and Medium Size Companies
  G. Santos, M. Montoni, A.R. Rocha, S. Figueiredo, S. Mafra, A. Albuquerque (Federal University of Rio de Janeiro), B. Diaz Paret, M. Amaral (Riosoft)

- New generation E-Learning technology by Web Services
  G. Dedene, M. Snoeck, M. De Backer, W. Lemahieu (Katholieke Universiteit Leuven)

- Discussion
Experience-based Support for Code Inspections

Markus M. Nick
Fraunhofer IESE
Sauerwiesen 6
67661 Kaiserslautern
Germany

Fraunhofer IESE
Fraunhofer Institute for Experimental Software Engineering
• founded 1996 in Kaiserslautern
• ca. 150 employees
• IESE consults and supports companies with software-intensive products and services regarding all issues of Software Engineering
• Customers from domains such as automobile, mechanical engineering, trade/banks/insurance, telecommunications, government, software developers, ...
Experience-based Support for Code Inspections

Inspections in Software Development

- Problem
- Requirements
- Design
- Implementation
- Test
- Deploy
- Operation + maintenance

- Requirements inspections
- Design inspections
- Code inspections

Software Inspections
- Defects are found early
  -> avoid rework

Tests
- Defects are found late
  -> rework is expensive
  (e.g., defects in requirements)

Software Inspections and Quality

- Efficient quality assurance technique
- Early defect detection is regarded as a necessity

Although benefits are evident: [Visek-ISERN Study 2002]
- Application by industry not yet systematically enough
- Most frequent reason: Insufficient tool support
  - particularly for defect detection [McDonald & Miller, 99]

Most important success factor: [Wong & Wilson, 2003]
- Experience => systematic learning across inspections,
  project phases and projects
Each Software Inspection has Four Steps:

- Planing
- Defect detection
- Collection (Meeting)
- Correction

#include <stdio.h>

```c
void foo()
{
    int fd = open();
    if (...) {
        close();
    } else {
        /* code analysis */
    }
}
```

Checklist for Inspection
- Close for every open?
- Open for every close?
- ...+

How do people actually do the defect detection?
- Little detail is known
- ISI as experimentation platform
  - identify factors
  - improve tool support

Development Context
- Quality requirements
- Architectural context

Inspection Experience Base
- ISI Tool provides input for
- helps supports
Experience-based Inspections-Process

- Development context & problem's situation:
  - MS Access is used as database and accessed from Java via the JDBC-ODBC bridge
  - Related problem: The value of a ResultSet of SQL queries is emptied by reading – in contrast to other JDBC drivers.
- Solution: [...] 

Experience-based checkitem:
Is an attribute value of a SQL query's ResultSet accessed more than once?

Code Recognition Patterns:
Calls of getString, getInt, etc. for a ResultSet instance

Statistics about ISI's Experience Base

<table>
<thead>
<tr>
<th>Checklists</th>
<th>#Checkitems available</th>
<th>% suitable for automatic locating</th>
<th>Locating of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard checklists</td>
<td>117</td>
<td>69%</td>
<td>• Language elements</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Method calls</td>
</tr>
<tr>
<td>Error management database</td>
<td>36% -&gt; code</td>
<td>100%</td>
<td>• Interface languages</td>
</tr>
<tr>
<td></td>
<td>31% -&gt; design</td>
<td></td>
<td>• Architectural dependencies</td>
</tr>
<tr>
<td>Security checklists and</td>
<td>127</td>
<td>100%</td>
<td>• Library functions</td>
</tr>
<tr>
<td>related material</td>
<td></td>
<td></td>
<td>• Array definition</td>
</tr>
</tbody>
</table>
Summary, Conclusions, Next Steps

- **3rd Generation Inspection Tool**
  - 2 closed loops for experiences
  - Improvements regarding checklists:
    - Locating where to check what
    - Adaptation to development context
    - Better quality focus

- Availability and suitability of checkitems shows feasibility

- Next steps
  - ISI as experimentation platform
  - Empirical studies
Using a Software Development Environment with Knowledge Management to Support Deploying Software Processes in Small and Medium Size Companies

Gleison Santos¹, Mariano Montoni¹, Ana Regina Rocha¹, Sávio Figueiredo¹, Sômulo Mafra¹, Adriano Albuquerque¹, Benito Diaz Paret², Márcio Amaral²

¹ COPPE – Federal University of Rio de Janeiro
² Riosoft – Núcleo Softex do Rio de Janeiro

Agenda

- Introduction
- Qualisoft Project
- Software Process Definition
- Software Process Training
- Taba Environments and CASE Tools
- Lessons Learned
- Conclusions and Future Work
Introduction

- Software Development Environments (SDE) are important to support the execution of software processes
  - SDEs are evolving to integrate knowledge management activities within software processes
  - SDEs with KM support help the institutionalization of a learning software organization
- Use of a software development environment to support the deployment of software processes in small and medium size Brazilian companies

Qualisoft Project

- Aims to increase the capability of organizations through the adequate use of Software Engineering techniques in their software processes
- Use of a software development environment to support the deployment of software processes in small and medium size Brazilian companies
Qualisoft Project

- **Focus:**
  - small and medium size organizations

- **Participants:**
  - Pool of organizations with similar characteristics

- **Project Coordinators:**
  - RioSoft - a non-governmental organization that integrates the Softex Program
  - COPPE/UFRJ – Federal University of Rio de Janeiro

- **Phases:**
  - 1st Phase: 2003 – 10 organizations
  - 2nd Phase: 2004 – 9 organizations

---

Qualisoft Project

- **Activities:**
  - definition of software development and maintenance processes
  - training in Software Engineering
  - training in the software processes defined
  - use of CASE tools
  - follow-up of the companies
Software Process Definition

- Identification of the characteristics of the organizations
  - Questionnaire
    - Organizational culture
    - Software process stages
    - Quality management systems
    - Software development practices
    - Main problems in the current software development and maintenance
    - Organizational objectives related to SPI
  - Interview
    - Visits to the organization

Software Process Definition

- Definition of standard software processes
  - Development
  - Maintenance

1st Phase:
  - based on ISO/IEC 12207

2nd Phase:
  - based on ISO/IEC 12207 and adherent to CMMI Level 2
Software Process Training

- **1st Phase:**
  - 32 hours
  - 80 professionals
  - **Topics:** Software Engineering, Software Process, Requirements Engineering, Configuration Management, Project Management, Software Products Quality

- **2nd Phase:**
  - 44 hours
  - 70 professionals
  - **Topics:** Peer-review, Tests, Measurement and Analysis, Supplier Agreement, Knowledge Management, (plus 1st phase topics)
  - Standard software processes defined
  - Taba Workstation CASE tools

Taba Environments and CASE Tools

- Process-centered environments
- Supported by Knowledge Management
- Until 2003 used only on academic projects
  - More than 50 master and doctoral theses
- Provide automated support to facilitate the deployment of the processes
- Available at no costs to the organizations
Taba Environments and CASE Tools

- **Configuration of the environments**
  - Tailoring of the processes to each organization
    - Identification of types of software developed
    - Documents produced
    - Software development paradigms adopted
  - Use in pilot projects
    - Tailoring of the processes to each project

---

Taba Environments and CASE Tools

- **CASE Tools**
  - Tailoring of processes for a project
  - Definition of the organizational structure
  - Acquisition, filtering, packaging and dissemination of organizational knowledge
  - Planning the organization of specific projects
  - time, costs, risks, human resources planning, monitoring and control
  - Configuration Management planning and execution
  - Identification of software product quality
  - Documentation planning
  - Measurement and analysis based on the GQM method
  - Requirements management
  - Post mortem analysis
  - etc
Lessons Learned

- Identified during regular meetings
- Confirmed through reports
- Lessons Learned:
  - the deployment of software processes in a group of organizations in a common project and at low cost is feasible
  - the configured environments facilitate training, deployment and institutionalization of software processes
  - the knowledge management approach adopted in the environments is determinant for the success of the approach
  - the knowledge acquisition approach integrated into the CASE tools enables the gradual evolution of the knowledge repository with acquisition and dissemination of lessons learned, best practices and ideas to improve the software processes.

Conclusions and Future Work

- Results of the Qualisoft Project
  - The experience is successful
  - Software deployment at low costs
  - Technology transfer between university and industry
  - 4 organizations were ISO 9001:2000 certified
- Other results
  - More than 30 Brazilian companies use the TABA environments
  - 1 organization has been evaluated as CMMI Level 2
- A 3rd Phase of Qualisoft Project will be conducted in 2005
  - 10 new organizations
  - 3 organizations will be evaluated as CMMI Level 3
Conclusions and Future Work

- A formal study was conducted with Qualisoft participants
  - Survey with 16 participants
  - Questions related to:
    - Characterization of the participant
    - Software process deployment
    - CASE tools support
    - Activities and procedures related to software processes

- There were adequate sensitization in the company concerning the importance of the use of the process.
- There were resistance to the deployment of the process.
- The projects were developed according to the defined process.
- The activities for Project Planning were adequate.
- The activities for Project Monitoring and Control were adequate.
- The activities for Requirement Management were adequate.
- The procedures for Configuration Management were adequate.
- The activities for Product Quality Assurance were adequate.
- The activities for Process Quality Assurance were adequate.
- The automated support reduced the effort for executing the processes activities.
- The automated support was adequate.
- The process was easily understood and used.
- The process were adequate to the projects.
- There were resistance to the deployment of the process.
- There were adequate sensitization in the company concerning the importance of the use of the process.
New Generation E-Learning Technology by Web Services

Prof. Dr. Guido Dedene
Prof. Dr. Monique Snoeck
Drs. Manu De Backer
Prof. Dr. Wilfried Lemahieu

LSO 2005, Kaiserslautern

Katholieke Universiteit Leuven
Department of Applied Economics
Naamsestraat 69
B-3000 Leuven

University of Amsterdam
PrimaVera Research Institute

Vlerick Leuven Gent Management School

Information & Communication Technology Teaching at K.U.Leuven

Faculty of Sciences:
COMPUTER SCIENCE

Faculty of Applied Sciences:
ENGINEERING
COMPUTER SCIENCE

Faculty of Economics & Applied Economics:
BUSINESS ENGINEERING in INFORMATION SYSTEMS

30 years
Business Information Systems Group Research Topics

- The development of Software Applications, including Systems Analysis and Design.
- Database Management Systems, Data Mining and Data Warehousing.
- Knowledge Based Systems, including Decision Tables and Fuzzy Logic.
- Groupware, Office Systems and Workflow Management.
- Economics of Information Systems, and Systems Management

INTRODUCTION TO INFORMATION TECHNOLOGY & SYSTEMS

- First year class
- 60-hours, compulsory computer course for applied economics and business engineering students
- 2 major goals:
  * Insight in current state-of-the-art of I.C.T., sufficient to judge in business applications.
    Emphasis on Platforms and Networking
  * Learn basic mechanisms to understand and construct computer applications.
    There are 2 additional courses on Databases and (standard) Office & Groupware software
Student Environment as *Learning Software Organisation*

- The users need a minimal hardware and software infrastructure on their workstations (processor speed, disk space, memory size, workstation operating system version, sufficient Internet access bandwidth, etc...)
- The (educational) support staff may suffer from a serious load in terms of installation questions, in particular for a large number of students. In the extreme case this may even become a bottleneck for the actual educational support.
- The cost for setting up the distribution mechanism (CD production, or Web-site(s) for downloading.
- The CD as well as Web-Download distribution results in multiple, potentially conflicting versions of the education environment that must be supported.
- The CD distribution typically requires additional Web-distribution for corrections, modifications and additional exercises.

New Generation Education Environment *Requirements*

- The education environment should have a simple installation and configuration procedure.
- The education environment should provide maximal Usability for the students, based on intuitive as well as defensive style of User Interfaces.
- The education environment should provide clearly structured, step-by-step simple exercises.
- The education environment should allow the student to construct solutions for the exercises, whereby the education environment should contain hints and/or guidance in the construction of the solutions.
- The education environment should support maximal automation in the process of creating, changing, maintaining and distributing exercises.
- The education environment should require minimal maintenance from a technical infrastructure point of view.
- The education environment should integrate in a frictionless fashion with other Internet or Intranet based education tools.
Context Requirements for Web Service-based E-duction

- when the education environment has clearly isolated “user sessions” and has the capability of distinguishing and maintaining the usage data of multiple users within the same environment.

- in case a simple user interaction, preferably on the basis of forms is used. In particular, not all current Windows-based user-interactions can be easily transformed into a Web-interface.

- when XML documents can be processed by the education environment, or XML formats can be used to import and export information with the education environment.

- when there is no need to store and maintain local data about the user on the user workstation.

Total Cost of Ownership Shifts

<table>
<thead>
<tr>
<th>Potential TCO reductions</th>
<th>Student</th>
<th>Educational Staff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Equipment Costs</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Lower Software Costs</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>No Installation Costs</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>No Installation Support Costs</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Flexible Access to the E-Learning application (any thin client Web Browser)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Better reliability of the E-Learning application</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Lower Software Distribution Costs</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Flexible maintenance of the E-learning software and exercises</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>E-Learning environment can reach more students</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Type of exercises: User Interface

• Description of the exercise:
  – “Create a program that displays Hello World”

• Part of the program:

```eiffel
Class
  ROOT_CLASS
Creation
  make
Feature – Initialisation
  make is
    -- Creation Procedure
    do
      ...
    end
End – class ROOT_CLASS
```

EIFFEL EDUCATION ENVIRONMENT

• “Fill-in exercise”
Welcome to "http://eee.econ.kuleuven.ac.be"

Contact Information
For technical information, contact:

For more information, contact:

Start Login
Username:
Password:
Sign in

Welcome, you have found and accessed the virtual education environment of the Faculty of Economics and Applied Sciences. EE is a training environment for the Sifting programming language. The following pages offer a wide range of exercises for the first year and second year students for the course "Introduction to Informatics" and "Introductie Informatiekunde".

Compiler

class ROOT_CLASS
creation make
feature -- initialization
    make = Creation procedure.
    in
        o.putstring("Hello World")
    end
end -- class ROOT_CLASS
4 fundamental ASP.NET pages
- Login.aspx
- ExerciseSummary.aspx
- Exercise.aspx
- Run.aspx
- Compile.asmx

Log in until successful
Other exercise
Log in successful
Other exercise
Retry exercise
Compilation successful
Start compilation
Return compiler output
Select exercise

- 1 Web Service

The Web Service based environment can serve a community of 900 students (with typically 30 to 50 concurrent sessions) with a 500 MHz-based Pentium Server, at 512 MB of main memory. Disk space is managed by “flushing” all the user files overnight (compiled materials only).

Significant improvements have been observed in the student results after the introduction of the Web Services based E-Learning environment. The Department of Applied Economics has 2 types of students, Applied Economics and Business Engineering students. The following table summarises the improvements for the same type of exams:

<table>
<thead>
<tr>
<th></th>
<th>Before E³</th>
<th>After E³</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applied Economics</td>
<td>8.1 / 20</td>
<td>9.3 / 20</td>
<td>15%</td>
</tr>
<tr>
<td>Business Engineers</td>
<td>9.3 / 20</td>
<td>11.1 / 20</td>
<td>19%</td>
</tr>
</tbody>
</table>

The Total Cost of Ownership reductions are quite dramatic in this case, mainly due to the reduced support costs on behalf of the educational assistants. The TCO-cost reduction can be estimated to be about 70%. This shows also how the introduction of Web Services provides a significant return on investment.
Welcome to LSO 2005!
Session 4

Kaiserslautern, April 10, 2005

Session V: Future Studies and Novel Approaches

- Software Process Improvement from a FLOW Perspective
  K. Schneider (University of Hanover)
  Discussion

- Local Software Organizations with Global Customers: a Survival Quest
  K. Smolander, P. Ovaska, P. Juvonen (South Carelia Polytechic)
  Discussion

- Causal Mapping Method as a Learning Approach for the Management of Risks
  in IS/IT Projects
  A.J. Al-Shehab, R.T. Hughes, G. Winstanley (University of Brighton)
  Discussion

- Using Open Space Technology as a method to Share Domain Knowledge
  T. Dingsøyr (SINTEF), F.O. Bjørnsson (NTNU)
  Discussion
Software Process Improvement from a "FLOW" Perspective

Kurt Schneider

Overview

- Current Process Models in Computer Science
  - Current Software Process Assumptions

- Focus on Flows
  - Information containers beyond "documents"
  - Information types beyond project information

- Experience FLOWs as Catalyst

- Examples of Modeling with FLOWs
  - A Legacy Situation
  - Variations of Reviews
Current Process Models

Project Information Flows

- Information flows through activities/phases

- Frequent implication
  - Information is contained in documents
  - Writing/reading relationships connects phases
  - Un-documented information does not exist/is not relevant

- However, explicitly
  - There may be other containers and media of flow!
Direct Information Flows

- Information is encoded in documents (write)
- Later, information is reconstructed from those documents (read)
- These two steps take a while
- Documents are persistent containers
- Information flows directly when people talk
- Encoding and decoding at one time
- Very fast, efficient, and "natural" information transfer
- Human brains forget

It is reasonable to use each mode when most appropriate

Micro Example
Many direct info items, a few documented
Agile Methods
build on direct information flows

- Incremental Development

- Continuous Flow of Information

- Appreciation of “tacit knowledge” (dashed lines)

Project Info and Experience Catalysts

Project 1

Project 2 (parallel)
FLOW Models Using Extended IDEF0

Design Experience (when to do what)

Requirements

Constraints

Design Activity

Basic Knowledge (incl. Design Patterns)

Software Design (to Implementation)

Variations of FLOW

• Comparing the flow profile of quality techniques (WCSQ-3)
• Studying influence of „flow“difs“
• Actively searching for most appropriate variant
Software Engineering FLOW Stories

Fortran Code is developed

User1  Dev1

User2  Dev2

Extensions for new users

Users want Java, now

User2  Dev2

Java Code

But wait! That’s a bigger thing

User2

Porting to Java

Dev3

Better yet: Use opportunity to learn

User2

Porting to Java and Reverse Engineering

Exp./Rationale

Dev3

Users want Java, now

User2

Java Code

Conclusions

• Information as glue of software processes
  – beyond documents
  – beyond phases
  – beyond mere project information

• Experience is a catalyst

• Taking direct flow seriously
  – From human to human
  – Fast and efficient
  – However: non-persistent

• FLOW models help to find good compromises in SPI
  – For specific and tailored processes
  – For a balance of direct and documented communications
  – For a mix of storage media and containers

Notation calls for more applications in order to improve!
Software Process Improvement from a “FLOW” Perspective

Kurt Schneider

Overview

• Current Process Models in Computer Science
  – Current Software Process Assumptions

• Focus on Flows
  – Information containers beyond "documents"
  – Information types beyond project information

• Experience FLOWs as Catalyst

• Examples of Modeling with FLOWs
  – A Legacy Situation
  – Variations of Reviews
Current Process Models

- Analysis (Requirements)
- Design
- Coding
- Test and Integration
- Operation

Emphasize activities and documents

Project Information Flows

- Information flows through activities/phases
  - Frequent implication
    - Information is contained in documents
    - Writing/reading relationships connects phases
    - Un-documented information does not exist/is not relevant
  - However, explicitly
    - There may be other containers and media of flow!
Direct Information Flows

- Information is encoded in documents (write)
- Later, information is reconstructed from those documents (read)
- These two steps take a while
- Documents are persistent containers
- Information flows directly when people talk
- Encoding and decoding at one time
- Very fast, efficient, and "natural" information transfer
- Human brains forget

It is reasonable to use each mode when most appropriate

Micro Example
Many direct info items, a few documented
Agile Methods
build on direct information flows

- Incremental Development
  
<table>
<thead>
<tr>
<th>Analysis</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>Code</td>
</tr>
<tr>
<td>Analysis</td>
<td>Code</td>
</tr>
</tbody>
</table>

- Continuous Flow of Information

- Appreciation of „tacit knowledge“ (dashed lines)

Project Info and Experience Catalysts

- Project 1
  - From past projects
  - Exp. Base
  - Patterns

- Project 2 (parallel)
  - From past projects
  - Exp. Base
  - Patterns
FLOW Models Using Extended IDEF0

Design Experience (when to do what)

Requirements

Constraints

Design Activity

Software Design (to Implementation)

Basic Knowledge (incl. Design Patterns)

Variations of FLOW

- Comparing the flow profile of quality techniques (WCSQ-3)
- Studying influence of "flow"diffs"
- Actively searching for most appropriate variant
Software Engineering FLOW Stories

Fortran Code is developed

User1 - Dev1

User2 Dev2

Extensions for new users

Users want Java, now

User2 Dev2

Java Code

But wait! That’s a bigger thing

Porting to Java

Dev3

Better yet: Use opportunity to learn

Porting to Java and Reverse Engineering

Dev3

Conclusions

- Information as glue of software processes
  - beyond documents
  - beyond phases
  - beyond mere project information

- Experience is a catalyst

- Taking direct flow seriously
  - From human to human
  - Fast and efficient
  - However: non-persistent

- FLOW models help to find good compromises in SPI
  - For specific and tailored processes
  - For a balance of direct and documented communications
  - For a mix of storage media and containers

Notation calls for more applications in order to improve!
Local Software Organizations with Global Customers: a Survival Quest

Kari Smolander
South Carelia Polytechnics

Introduction: the local business environment

- Southeast Finland has one of the world’s densest concentrations of forest industry
- Many software organizations serving the needs of the forest industry locally
- Strong globalization of the forest industry
- One large, globally operating software company in the area – the others are small and local
- An asymmetrical relationship between local software organizations and forest industry
  - their resources and sizes are in different category
  - one-way dependency – the global customer can easily replace its local software supplier
The objective of the research

• How a local software organization operates in such an asymmetrical situation?
  – What kind of development methods and processes are usable?
  – What kind of skills are required from software professionals?
  – What kind of learning and development can be introduced in practice?
  – What kind of tools and technologies are used now and in the future?
• The goal is to develop organizations and software processes in an asymmetrical business environment

The research process

• Two phases with a common plan
  – Visiting all software organizations in the region
    • key figures, business models, shortcomings in software processes
    • ~100 organizations visited
    • done by Lappeenranta University of Technology (Jul-Oct/2005)
  – Deeper inquiry into those organizations dealing with a global industrial customer
    • 19 theme interviews in 14 organizations (Sep-Nov/2005)
      – IT managers, CEOs, project managers
    • 11 interviews in local software organizations, 5 in forest industry enterprises, 3 in local IT service organizations
    • interviews were recorded and then fully transcribed as text
  – Qualitative research: grounded theory as the analysis method
  – 2 conference papers published this far
Data analysis

• Available data
  – Interview transcripts
  – Key figures, information about customers, competitors, partners, etc.
• Three researchers with separate viewpoints to the data
  – problems and good practices in local software organizations
  – business partnerships between local software organizations and their global customers
  – initialization of software projects in studied organizations

Analysis

• Open coding: the data was searched for mentions about
  – Problems and good practices
  – Current relationships between the organizations
  – Weaknesses and strengths of partnerships
  – Future expectations about the relationships
• Seed categories of “problems”, “good practices”, “current relationships” and “future expectations”
• Axial coding: forming the relationships between identified categories
• Selective coding: explaining the core category, local software development for global industry
Characterizing the business situation

- Four characterizing dimensions were identified for local software organizations:
  - Technological sophistication
  - Resource providing
  - Industrial domain knowledge
  - Software engineering knowledge
- The strategy of each organization varied according to these dimensions
  - None of the organizations had high values on all these dimensions

Four dimensions of current business strategies
The possibilities of small local organizations with global customers

• Challenges
  – Globalization, consolidation of industries
  – Competition with strategic global partners
• Small organizations can not operate as partners in the area of operative business systems
  – Lack of credibility of small organizations
  – Risk avoidance in business critical systems
  – Large industrial customers want to cooperate with big software organizations
  – Standardization in technology infrastructure and information systems
  – Large industrial customers want reduce their number of partners
• However, several possibilities for small organizations were expressed by industrial customers

Four dimensions and future trends

- **industrial domain knowledge**
- **tailored technology solutions**
  - the use of mobile and internet technologies
  - the interfaces between forest industry systems
  - data mining and information processing from automation systems
- **software products**
  - administrative systems targeted for different business domains
- **operative business systems**
  - global partner
- **resource pool**
  - local "hands and legs"
- **software engineering knowledge**
  - technology
  - resources
Small as a technology developer

- Requires innovation capabilities
- Mentioned examples included
  - mobile and internet applications in the industrial domain
  - interfaces between systems
  - data mining from automation systems
- Challenges
  - close cooperation with industry customer and universities and other research institutions
  - finding application domains where global software enterprises do not compete
  - learning new technologies, keeping skills up-to-date

Small as a resource pool

- "Local hands and legs" in implementation and maintenance
- Subcontractor for large software organization – also directly for industry
- Flexibility, agility as strengths
- Local presence in startups and maintenance
- Challenges:
  - networking with large software contractors
  - adopting software engineering practices used by main contractors
  - competing with organizations from lower labour cost countries (in Finland: Russia, Baltic and other East European countries)
  - keeping fixed costs low
Small as a software product developer

- Making software products for several business domains
- Mostly administrative business systems
  - domain independent systems, reporting systems
- Challenges
  - many suppliers, hard competition
  - small companies have only little resources for marketing
    - both human and financial resources needed
  - how to network with other complementary small software organizations to give more added value to customers
  - may require an entry to international markets (especially in small markets such as in Finland)

General challenges for small software organizations

- Adopting better software engineering practices and more efficient tools
- Need for better quality assurance and management
- Need for better project management skills
- Learning to use foreign languages
- Networking with other small organizations
  - enhancing competitive abilities with large customers
  - gain a win-win situation with those participating in the network
  - Everyone is interested in participation – few examples exist
Summary

• Current situation, possibilities, challenges, and future trends of small local software organizations identified
• Three possibilities for small organizations
  – work as a local subcontractor for larger organizations
  – specialize in some new technology
  – make products for multiple business domains
• Each has its own challenges + general challenges for small organizations

Future research

• Current research in software engineering is mostly targeted for large organizations
• Research serving the needs of the small is needed
  – methods, tools, and processes for small organizations
• Instructions, descriptions, and cases of networking between small technology organizations are needed
  – best practices, success stories, requirements for success
  – globalization and internationalization issues
  – a possibility for international research cooperation
Causal Mapping Method as a Learning Approach for IS/IT Risk Management

Abdullah J. Al-Shehab, Robert T. Hughes, Graham Winstanley

School of Computing, Mathematical and Information Sciences

University of Brighton, UK

LSO 2005 Conference
12 April 2005

Content

• Learning in Risk Management.
• Causal Maps (CM).
• Work under progress.
Aim of paper

• Presenting risks and their relationships visually.
• Use diagrammatic techniques to facilitate group working.
• Apply CM as a learning approach.

Learning, Evaluating & Documenting

• Learning in the Risk Management process.
  – “Risk Leverage”, (Robin et al 2002)

• Project Evaluation
  – Post-Project Review
  – Post-Mortems!
  – Project Post-Evaluation
  – (report, interviews, email, questionnaires…….)

• Could we add an approach to introduce a more clear representation.
Causal Maps

• A word-and-arrow diagram (Bryson 2004)
• can represent how a system works. (Scavarda et. al 2004).
• used frequently in operations management (Scavarda et al 2004).
• can portray information about a system efficiently. (Huff 1990)

Initial Experimentation

• Master level student MSc in Information System
  – Apply case study to 4 groups
• The experiment showed a positive feedback from the students.
Work Under Progress

Applying Causal Maps to Case Study

Longitudinal Case Study in IS Project Failure

• Investigating the problems led to failures of the project for 2 groups who worked on the project.
  – Managers.
  – Technical Staff.
• Maps were drawn individually and in group sessions. (30 different maps)
• 2 final combined maps
Experimenting with new tool

- **Visio (Microsoft)**
  - Drawing tool
  - Limitation on merging & combining
  - No analysing functions
- **Decision Explorer (Banxia)**
  - Drawing tool
  - Can capture mental perception.
  - Ability to merge and combine concepts
  - Analysing functions
DE diagram
Observations

- Mapping two groups (managers and Staff)
- We found many differences in the maps depending on:
  - role of stakeholder in the project
  - experience in the field
Future Work

• **Historical Repository of SD risk.**
  – Use of post-evaluation output as repository.
  – Project problems frequently repeat. 
  *Demarco 2003.*

• **Risk Pattern.**
  – One problem of risk, “it is not foreseen as a risk at the time”. *Bartlett, 2002*

• **A common ontology for risk.**
  – High level description vs. General design.

Future Work

• Developing an *initial generic model for risk management.* (base to assists PM’s)

• Analyse the case study to establish *relationship patterns* among the concepts that could be found in a ‘real world’ project as a *risk identification* approach.

• Examining the feasibility of *measuring the strength/weight* of these *relationships.*
Framework for LSO

- Stakeholders in a project could use the framework for PPR.
- Visual simulation, propagation and evolution of risks throughout the project life cycle to identify risk as a forecasting approach.
Using Open Space Technology as a method to Share Domain Knowledge

*Learning Software Organizations Workshop 2005, Kaiserslautern, Germany*

Torgeir Dingsøyr, SINTEF
Finn Olav Bjørnsson, NTNU

---

Overview

- Understanding Requirements
- Large Group Interventions
- Open Space Technology
- Suggested Research Project
Understanding Requirements

- A critical process in software engineering
- Up to 200 times more expensive to detect and repair errors in the maintenance stage*
- Most work in software engineering focus on
  - A clear, concise requirements document
  - Checking for consistency
- Little work on the developers understanding of the requirements and the customer’s domain


Software Development and Knowledge Integration*

Software Development Effectiveness versus Knowledge Integration

Software Development Cost Overrun versus Knowledge Integration

Large Group Interventions

- Large Group Intervention
  - Involve a large number of people in planning and implementing major change efforts
- Four aims*
  1. Enhance the amount of information brought to bear on a problem
  2. Build commitment to problem definitions and solutions
  3. Fuse planning and implementation
  4. Shorten the time needed to conceive and execute major projects


Large Group Intervention Processes*

<table>
<thead>
<tr>
<th>SIMPLER</th>
<th>STRATEGIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sophisticated</td>
<td>Strategic Options</td>
</tr>
<tr>
<td>Sophistication of problem and solution framing and analysis tools</td>
<td>Development</td>
</tr>
<tr>
<td></td>
<td>Analysis</td>
</tr>
<tr>
<td></td>
<td>Strategic Choice</td>
</tr>
<tr>
<td></td>
<td>Future Search</td>
</tr>
<tr>
<td></td>
<td>Search Conference</td>
</tr>
<tr>
<td></td>
<td>Technology of Participation</td>
</tr>
<tr>
<td>Simple</td>
<td>Clear</td>
</tr>
<tr>
<td>Real Time Strategic Planning</td>
<td></td>
</tr>
<tr>
<td>Open Space Technology</td>
<td></td>
</tr>
</tbody>
</table>

Open Space Technology Assumptions*

- Event focus on an issue of concern, and when the purpose becomes clear, the appropriate event and project structures will follow as a natural expression of the purpose.
- People will self-organize based on their interests. Everyone has the right and responsibility to put items on the agenda. Everyone has creative potential.
- Experts and analysts are needed among the participants. There should be no expert help from outside the group.
- You can work with the chaos of these events. It represents an opportunity for growth, organizational learning, and improved effectiveness.


Open Space Technology Principles*

- The law of two feet:
  - Participants have the right and responsibility to use their feet to go to a productive and meaningful conversation or place for themselves.
- Four principles:
  1. Whoever comes are the right people
  2. Whatever happens is the only thing that could have happened
  3. Whenever it starts is the right time
  4. Whenever it is over, it is over

Open Space Strengths and Weaknesses

Strengths
- Flexible
- Responsibility
- Ownership

Weaknesses
- Lack of control
- Investment in time

Suggested Research Project

Research questions
- Does Open Space Technology give developers better insight in domain knowledge
- Does Open Space Technology give developers a better opportunity to find domain knowledge after the requirement process is over

Research method
- Explorative action research study in a medium-size software consulting company
- Select one project where Open Space is used in requirement process
- Select one project where requirement process is done "normally"
- Data collection through semi-structured interviews, discussion and observation
**Conclusion**

- Requirements engineering needs improvement
- Large-group interventions candidate to increase domain knowledge
- Open Space technology particularly interesting technique: "simple and unclear"
- Suggested research design for explorative study of Open Space in the requirements process

---