From Agile Software Product Line Engineering Towards Software Ecosystems

PhD defence, November 30, 2010
Geir Kjetil Hanssen
Overview

• Research challenge
• The case
• Research design
• Study method
• Data
• Software product line engineering
• Agile software development
• Agile software product line engineering

• Software ecosystems
• Results and discussion
• A conceptual model of a software ecosystem
• Implications for practice
• Implications for theory
• Conclusions
• Future work
Research challenge

Objective: Understand the need for and implications of a more open approach to software product line engineering

RQ1: How can software product line engineering and agile software development be combined?
RQ2: How does a software ecosystem shape?
The case

Organization

- Main office and main segment of R&D is located in Oslo
- Two additional development locations in Eurasia
- Sales and support departments in U.K. and USA
- R&D is made up of fixed module teams

---

1) Due to a non-disclosure agreement with CSoft; company, product, and domain-specific details are omitted to maintain the anonymity of the case organization.
The case (the organizational development)

- Creative chaos
  - CSoft is established
  - QA function established
  - Introducing waterfall
  - 1996

- Waterfall - SPLE
  - EPG introduced
  - Evo 3 month trial
  - 2000

- Agile product line engineering
  - PMT established
  - Evo adopted
  - 2002

- Software ecosystem
  - External community
  - PSG replace PMT
  -Architecture team est.
  - Evo 3 month trial
  - External community
  - 2004

- Plan-driven and closed
  - 260 employees

- Reactive and open
  - 1998

- 2 employees
  - 2000

- 2002

- 2004

- 2006

- 2008

- 2010

www.ntnu.edu
Research question 1
How can software product line engineering and agile software development be combined?

Research question 2
How does a software ecosystem shape?

Adoption of Evo
(Agile SPL) + (Software entropy)

(Software ecosystems)


Study 1
P1
P2
Study 2
P3
Study 3
P4
Study 4
P5
P6

P6: G. K. Hanssen, An Emerging Software Ecosystem: A Longitudinal Case Study, accepted for publication Journal of Systems and Software.
Study method: characteristics

- **A case study**
  A preferred research strategy “…when a «how» or «why» question is being asked about a contemporary set of events over which the investigator has little or no control.”

- **An interpretative field study**
  Applying Klein & Myers seven principles of interpretive field studies. Based on the fundamental principle of the hermeneutic circle.

- **A longitudinal study**
  Suitable in the context of turbulent, flexible, and uncertain organizational and environmental conditions in today’s industry. Theoretically sound and practically useful research on change should explore the contexts, content, and process of change together with their interconnections through time.

---


Study method: case, data sources, analysis

• Selecting the case
  – Based on an existing relationship
  – An emergent opportunity to study the adoption of ASD in an SPLE organization

• Data sources
  – Interviews (single and group)
  – Documents (roadmaps, business plans and presentations)
  – Observations (product conference and stakeholder review meeting)

• Analysis
  – All data are made textual
  – Text was coded using NVivo™
    • Identifying concepts being important to the research questions
    • Extracting information on the case (organization, processes, product line and a retrospective of the organizational development
    • Identifying important concepts

→ Describing an emerging ecosystem
<table>
<thead>
<tr>
<th>Study</th>
<th>Data sources</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Study 1 (paper 1 &amp; 2)</td>
<td><strong>Interviews</strong>&lt;br&gt;• One group interview with six developers using the PMA technique (Birk, Dingsøyr et al., 2002)&lt;br&gt;• Three semi-structured interviews with the participating customers&lt;br&gt;• Five semi-structured interviews with the PMT members</td>
<td></td>
</tr>
<tr>
<td>Study 2 (paper 3)</td>
<td><strong>Interviews</strong>&lt;br&gt;• One semi-structured interview with the CTO&lt;br&gt;• One semi-structured interview with the PMT manager</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Documents</strong>&lt;br&gt;• Six business plans for product line modules and main features</td>
<td></td>
</tr>
<tr>
<td>Study 3 (paper 4)</td>
<td><strong>Reusing data from study 1 and 2. New data:</strong>&lt;br&gt;<strong>Interviews</strong>&lt;br&gt;• One group interview with two of three members of the architecture team. The interview was done after the architecture analysis as an open discussion about software entropy.&lt;br&gt;• One semi-structured interview with one of the team leaders&lt;br&gt;• One semi-structured interview with a team member/developer</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Other</strong>&lt;br&gt;• Architecture analysis using NDepend. Results are documented in (Hanssen, Yamashita et al., 2010; Hanssen, Yamashita et al., 2009; Smacchia, 2008).</td>
<td></td>
</tr>
<tr>
<td>Study 4 (paper 5&amp;6)</td>
<td><strong>Interviews</strong>&lt;br&gt;• One interview with the R&amp;D manager&lt;br&gt;• One interview with the manager of Professional Services&lt;br&gt;• One interview with the PSG manager&lt;br&gt;• Three interviews with PSG members&lt;br&gt;• One interview with a Technical Account manager&lt;br&gt;• One follow up interview with the PSG manager to clarify issues and notes after observation of customer review meeting (source #18).</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Observations</strong>&lt;br&gt;• Product conference, London 2008&lt;br&gt;• Customer review meeting, 2008</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Documentation</strong>&lt;br&gt;• Five module product plans, four presentations at the 2008 product conference (CSO’s, CEO’s, VP Product Marketing’s presentation and customer’s presentation)</td>
<td></td>
</tr>
</tbody>
</table>
Software product line engineering

SPLE is the sum of developing reusable core assets and developing products using core assets. A line of products display commonalities and variability.

“A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.”  

“Software product line engineering is a paradigm to develop software applications (software-intensive systems and software products) using platforms and mass customization.”


Agile software development

Emphasize a low level of up-front planning or development, frequent feedback and correction, constant customer interaction, incremental and iterative development.

The most known and practiced agile methods are Scrum and Extreme Programming (XP).
Agile software product line engineering

- The combination of SPLE and ASD have recently gained some interest.
- The basic idea is to address the weaknesses of the one with the strengths of the other.
  - Increase flexibility and customer proximity in SPLE.
  - Manage a large diversity of products, a large customer base and long-term development.
- Recent studies show:
  - A combination is feasible, from a practical perspective.
  - SPLE and ASD work together, supporting strategic and tactical objectives, respectively.
  - Day-to-day operational processes constitutes the experience-bearing link from the tactical to the strategic process.
  - A process combination affects many aspects, such as product planning, knowledge management, organizational aspects, and innovation.
Software ecosystems

- A new concept with no exact agreed definition (yet)
- To fundamental concepts
  - A network of organizations and actors
  - A common interest in the development and use of a central software technology
- Three high-level roles
  - A keystone organization
  - End-users
  - Third-party organizations (extending the technology beyond the keystone organization)

A simple software ecosystem taxonomy

<table>
<thead>
<tr>
<th>end-user programming</th>
<th>MS Excel, Mathematica, VHDL</th>
<th>Yahoo! Pipes, Microsoft PopFly, Google’s mashup editor</th>
<th>none so far</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>MS Office</td>
<td>SalesForce, eBay, Amazon, Ning</td>
<td>none so far</td>
</tr>
<tr>
<td>operating system</td>
<td>MS Windows, Linux, Apple OS X</td>
<td>Google AppEngine, Yahoo developer, Coghead, Bungee Labs</td>
<td>Nokia S60, Palm, Android, iPhone</td>
</tr>
</tbody>
</table>

Results and discussion:
An agile software product line engineering process

• Actions taken to combine the processes
  – Simplification of the process and organization
  – Engagement of customers

• Effects of the combined process
  – Risk reduction
  – Organizational development
  – Reduced maintainability
  – Community building
  – Openness and visibility
  – Company culture

• Contextual factors
  – Domain volatility
  – The SAAS delivery model
Results and discussion:
An emerging ecosystem

The change from an agile SPLE to a software ecosystem:

– Engaging externals – Increasing visibility
– Learning by doing – Increasing extensibility

<table>
<thead>
<tr>
<th></th>
<th>supplier</th>
<th>customer</th>
<th>3rd party</th>
</tr>
</thead>
<tbody>
<tr>
<td>supplier</td>
<td></td>
<td>Sales of licenses and services.</td>
<td>Maintaining focus on the core product line. Input to plans and development.</td>
</tr>
<tr>
<td>customer</td>
<td>Business enabling tools and services. Early insight into technology development.</td>
<td></td>
<td>Value adding solutions and services.</td>
</tr>
<tr>
<td>3rd party</td>
<td>Access to an established customer base and a technical platform. Early insight into technology development.</td>
<td>Sales of licenses and services.</td>
<td></td>
</tr>
</tbody>
</table>
A conceptual model of a software ecosystem

- Supplier
  - develops
  - guides
  - learns about
  - participates

- Product line
  - use
  - extends

- Strategy
  - sees

- Customers
  - serves

- 3’d parties
  - learns about

External environment
Implications for practice

• Support the external environment by being open. Sharing information on plans, strategies, and development may create a fundament for collaboration and new patterns of innovation. If appropriate, encouraging and supporting a third party community can be a valuable extension to the normal development of the product line.

• Reducing the variability in the product line consequently reduces the costs of maintaining and managing variability. The potential drawback of reduced variability can be compensated through offering interfaces for extending the product line by externals, and thus increasing the overall variability.

• Establishing and benefiting from a software ecosystem takes time. A successful development relies on repeated cycles of experimentation and learning. This learning process needs to involve all types of actors.
Implications for theory

• Building on the concept of organizational ecology\(^1\) to develop a concept of software ecosystems:
  1. *Member organizations in a software ecosystem are linked to a key organization among them, which acts as a central referent organization*, doing so even though many of the *members* are only partially under its control or linked to it only through *interface relations*.
  3. *Software ecosystems have a networked character*.
  4. *Software ecosystems exist through the use of information and communication technology (ICT)*.
  5. *Software ecosystems exhibit shared values*.

In addition, we propose the following extensions to Trist's concepts:

6. *The shared value of a software ecosystem is both the software product and the business domain*.

7. *As a software ecosystem emerges, control moves from the supplier of the software to its users*.

Conclusions

• RQ1: How can software product line engineering and agile software development be combined?
  – A strategic process takes care of long-term planning, resulting in high-level plans for the product line.
  – A tactical process encompasses the development of new releases, organized as incremental and iterative processes per module in the product line.
  – A day-to-day operational process connects the strategic and tactical process with feedback from the use of the product line.
  – Two important enablers for this approach is 1) the active engagement of external actors both in long-term planning and in development, and 2) openness and visibility in plans and development.
Conclusions

• RQ2: How does a software ecosystem shape?
  – Opening
    • Through the opening of the information flow internally and externally
    • Through the opening of the innovation processes by collaborating with external actors at various levels
    • Through the opening of the technical interfaces, enabling external actors to use the product line as a platform for value adding services and solutions
  – Collaboration with external actors
  – Active support of the external community
  – An experience-driven change
Future work

- We need to see more empirical studies of various types of software ecosystems, how they develop, and the effects they produce. Such studies should naturally be focused towards the industry, and be longitudinal as well as exploratory.
- We advise a further refining of a theory of software ecosystems to build a common understanding of software ecosystems: how they shape, how they work, and what their effects are.
- We need to see more dedicated studies of new business models affecting intellectual property rights, economic models, competition etc. to realize the potential of ecosystems.
- Software ecosystems are closely related to the more mature concept of open source software development. We need to better understand the similarities and the differences in order to transfer knowledge between these two related domains.
- The engine of a software ecosystem is the collaboration with external actors. We have showed some examples through our studies, but this is a broad topic that needs further investigation.
- The study of software ecosystems potentially relates to several disciplines such as business strategy, sociology, technology and innovation management, economy, and others. We have briefly touched a few of these and we see a need to investigate these links further.
- Software ecosystems affect the shape of control structures. We believe that control shifts from the supplier towards the users, a transition that needs to be better understood.