Towards Mission-Critical Software-Intensive Ecosystems

Klaus Marius Hansen
Professor, PhD
Department of Computer Science (DIKU)
University of Copenhagen, Denmark
klausmh@diku.dk
Architectural solutions for growing a platform

Create a *product line*, manage a set of core assets, produce applications from these?
- Has worked for various organizations
- Most successful for closed, technical/embedded systems (cf. Nokia)

Create and control common platform, source *components* from suppliers, reuse existing components?
- Has been tried over and over again; seldom works
- A long and somewhat painful history (e.g., EPR in Denmark, UK, ...; IBM IFW)

Nurture a *software ecosystem*, allowing suppliers and users to create applications in this
Software ecosystems

Working definition
- A software ecosystem consists of a set of actors and software elements in a domain interacting as a system

Many existing keystone examples,
- Microsoft Windows
- Apple iOS
- Google Android

Ecosystems go through birth, expansion, leadership, self-renewal (Moore, 1993)
Domains of software-intensive ecosystems

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

(Bosch, 2009)
Operating system-level ecosystems

Examples
  - Windows, Google AppEngine, iOS

Characteristics
  - Domain-independent
  - Often stand-alone applications
  - Development tools central

Challenges
  - Complexity of configurations – hardware and software
  - Weariness of dominant players

Success factors
  - Minimizing effort in building applications
  - Ability to accommodate functionality early
  - Number of users of operating system
Application-centric software ecosystems

Examples
- AutoCad, SalesForce

Characteristics
- Platform built on successful application
- Hosting or similar by platform provider
- 3rd party developers extend functionality
- Integration of data, workflow, user experience

Challenges
- Transitioning from a product strategy to a platform strategy
- Hard to establish a viable business model for 3rd party developers

Success factors
- Potential large set of end customers for 3rd party developers
- Simplify contribution by 3rd party
- Ability to extend data models and workflows
- Platform expose 3rd party contributions to end customers
End-user programming software ecosystems

Examples
- Microsoft Excel, Lego Mindstorms, Yahoo!
- Pipes

Characteristics
- Often take the form of a Domain-Specific Language (DSL)
- Focus on domain experts with no CS/engineering background
- Often based on pipes-and-filters architecture
- Composition of pre-existing building blocks

Challenges
- Modeling the application domain and creating the DSL

Success factors
- Amount of value created by end user
- Providing efficient ways of sharing applications
Mission-critical software systems

*Software systems in which a software failure may mean business failure*

**Examples**
- Healthcare  
  - E.g., telemedical applications
- Industrial  
  - E.g., process control
- Finance  
  - E.g., (internet)banking

**Observation**
- Many software systems in this category are built as a product on top of a platform using service-oriented principles

**How can such platforms be opened up to become an ecosystem?**
Strategies for software ecosystems

Assumption
- The is an existing service-oriented platform for the (mission-critical) applications owned by an organization

Internal/closed
- The organization allows new products to be composed from the platform and new, internally developed components
- Incentives are established to build and integrate new components
- E.g., build and reuse a loan processing component in an organization that builds multiple banking systems for internal customers

External/open
- Externally developed components may be composed with the platform
- Organization publishes (and hosts) externally developed components
- Revenue flows to developers of external components
- E.g., build a Facebook plugin that allows money transfer without giving out account numbers

The external/open strategy has higher potential of open innovation
Characteristics of software ecosystems

1. Controlled (components)

2. Reused (platform)

3. Homogeneous (actors)

Legend:
- **internal**
- **external**

- Autonomous
- Composed
- Pluralistic
Software development challenges

Mission-critical systems often rely on standards (and standardized systems)
  • (Open) software ecosystems require the ability to externally develop components
  • Barrier to entry should be low
  • Opportunity to create a façade for standards – a DSL for components

Failures should be avoided
  • A failure is the
    • termination of the ability of a functional unit to perform a required function
    • which is ultimately caused by a fault (defect/bug) that is an
      • abnormal condition that may cause a reduction in, or loss of, the capability of a functional unit to perform a required function
  • (IEC 61508)
Software development challenges

Failure avoidance includes

- Fault avoidance: avoid faults being introduced in a component (e.g., mandate single-threaded programming)
- Fault prediction: predict if faults are present in a component (e.g., through model checking)
- Fault tolerance: avoid faults manifesting as failures at runtime (e.g., use transactions)

This is a challenge in a compositional world

- With respect to functionality
- With respect to quality
  - Security, availability, testability, performance, ...
Summary

A *software ecosystem* is set of applications and actors (building and using these) that are considered a whole

- Software ecosystems may be seen as an evolution from product lines and component platforms
- There are technical and business challenges in supporting an ecosystem strategy

*Mission-critical* systems are systems whose failure may mean business failure

- Need a focus on functional and non-functional reliability
- This is a particular challenge in a software ecosystem setting

Part of the Net4Care project, funded by the European Union and the Central Danish Region through Caretech Innovation, [http://www.net4care.org](http://www.net4care.org)