Evolution of a platform - Challenges inside a Software Ecosystem

29.11.2012
Introduction

“What do you want to accomplish with a SECO?”

“What challenges have to be considered?”
Different Perspectives

Software Vendor Level

Software Supply Network Level

Software Ecosystem Level

Source [1]
Software Vendor Level

Own products and services only

- Product Line Planning, see also:
  “Software-Produktlinien-Entwicklung in Software-Ecosystemen” from Martin Russer
- Knowledge Management (e.g. guides, feedback, videos)
- Architecting for extensibility, portability and variability
- Development organization system integration
  (keep other interfacing systems in mind → automatization)
Software Supply Network Level

Include first-tier buyers and suppliers

• Establishing relationships in a SSN
  – Meetings, workshops, different web portals

• Release heartbeat and release timing
  – Developer want to reuse the newest features
  – End-users demand a stable system
  – Common solution: beta-channels

• Managing quality in the SSN
  – Certification, testing
Software Ecosystem Level

All interacting business units

- Characterisation and modelling of SECOs
  - e.g. size, livelihood, presence of standards
  - Currently no modelling formalism
- Developing policies and strategies within SECOs for SECO orchestration
- Determining a strategy to thrive and make profit in an SSN
  - e.g. iOS App Store 30% apple, 70% developer, yearly fee
Overview

- Intrinsics
  - Platform Architecture
  - Platform Governance
- Environmental fit
- Environmental Dynamics

Theoretical lenses:
- Bounded rationality
- Evolutionary selection
- Modular systems theory
- Real options theory

Evolutionary Dynamics

(compare Source [2])
Slightly simpler definition

- SECO = collection of the software platform and the modules specific to it
- Two different levels of analysis
  - SECO level or
  - module/platform level
- Applicable to each area of challenges
- Potentially different causal explanations and contributions
Evolutionary dynamics

- Supplement pervasive, classical notions of performance (e.g. efficiency, effectiveness...)
- Temporal distinction in short and long term evolutionary dynamics helpful
- Relative to the lifespan of the SECO
  - Operating systems about 5-10 years
  - Smartphone Apps about 6-24 months
Long term evolutionary dynamics

- Evolution rate
  - Android ~six months
  - iOS first ~yearly
- Envelopment
  - Integration of functionality from a different SECO
- Derivative mutation
  - Unanticipated creation of a spin-off
- Survival/mortality
- Durability
  - Persistence of market advantages and distinctiveness
  - e.g. Apple Siri
Short term evolutionary dynamics

- **Composability**
  - Ease with which functionality-extending changes can be made to a platform/module
  - Without compromising integration or functionality

- **Malleability**
  - Ease with which the platform/module can be reconfigured to refine/extend their behaviours
Platform architecture

- Decomposition
  - e.g. iOS four layers
  - Different modules
- Modularity
  - Avoid ripple effects between modules
- Design rules
  - Have to be stable and versatile at the same time
- Changes often irreversible or at least difficult to reverse
  → “Humpty-Dumpty” problem [2]
- see also: “Architektonische Herausforderungen in Software-Ecosystemen” from Michael Strotz
Platform governance

“Who decides when what?”

- Control
  - Formal and informal mechanisms
  - Can be bidirektional: platform owner ↔ module developer
- Decision Rights Partitioning
  - what, how, who?
- Proprietary versus Shared Ownership

see also: “Architectural Guidance & Governance” from Klaus-Benedikt Schultis

Theoretical lenses:
- Bounded rationality
- Evolutionary selection
- Modular systems theory
- Real options theory
Internal fit

„How are evolutionary dynamics influenced by the interplay of platform architecture and governance?“

- Architecture changes over time → coevolution of governance beneficial
- Complementarities
  - Architecture reinforces/diminishes the influence of governance
Example for internal fit

- At the module level
  - Modularity + control
    → better composability
    - Modularity decreases interdependencies
      → isolates ripple effects from errors
      → less coordination necessary
      → reduced module-to-platform integration effort
      → supports changes to single modules
      → better composability
    - Output control sufficient, process control obstructive

Theoretical lenses:
- Bounded rationality
- Evolutionary selection
- Modular systems theory
- Real options theory

Diagram: Platform Architecture, Platform Governance, Environmental Dynamics, Environmental fit, Internal fit.
Environmental dynamics

• Convergence
  – Emergence of complementary and substitutive technologies
  – Convergence of adjacent domains → envelopment

• Multihoming costs
  – Developer's costs of associating with multiple SECOs
  – Also consider documentation, toolkits, adapters...

• Influence exerted by complementors
  – influence/services from outsiders (e.g. AT&T → Apple)
“How does the interplay between the platform and its environment influence the evolutionary dynamics of the SECO?“

  - Mutual interaction as choices influence expectations
- How fast can opportunities be exploited/recognized?
- When to make/announce decisions?
Example for environmental fit

- At the SECO level
  - Modularity + convergence → envelopment
    - Convergence leads to envelopment opportunity
    - Modularity allows to actually exploit this opportunity
      → Convergence and modularity cooperate
  - Unflexible architecture obstructs envelopment
    (Regardless of existing opportunities)
How to find causal explanations?

- “Theoretical lenses”
  - four not widely used ones
    - Bounded Rationality
    - Evolutionary Selection
    - Modular Systems Theory
    - Real Options Theory
  - Could prove especially useful to find causal explanations
Bounded Rationality

Premise:
Developers only can process and interpret a bounded volume of information
→ leads to “good enough solution”
→ impact onto the evolution of the SECO

- Limit scope of information a developer must process
  - e.g. use stable design rules
- Contain ripple effects from errors
  - e.g. through modularization
Evolutionary Selection

Premise:
A higher evolution rate and diversity lead to a better environmental fit.

- Decomposition + influence of complementors → mortality
  - Higher grade of decomposition
    → faster evolution through recombination
    → better chance of survival
  - In contrast: increasing influence of complementors
    → constrains or even cancels out benefits of decomposition
    → increases likelihood of the SECO's mortality
Modular systems theory

Premise:
A system, which is composable into smaller subsystems, which only interact only through predefined, stable interfaces is more amendable to changes.

• Ideas especially interesting in SECOs:
  - Decreased need for coordination
  - Less effort to manage dependencies
  - Can replace process control
  - Allows for deeper specialization
Real options theory

“The right to do sth. without the obligation to do it”

- Entails future flexibility
- Has to be consciously embedded within the architecture
  - Upfront option acquisition cost
  - Opportunity to exercise option when beneficial
  - e.g. modular operators: splitting, substitution, augmentation, exclusion, inversion and porting
- More valuable with greater uncertainty in the environment
Conclusion

- Different perspectives (each with own challenges)
- Five big areas of challenges:
  - Platform architecture
  - Platform governance
  - Environmental dynamics
  - Internal fit
  - External fit
- Different theoretical lenses to deliver causal explanations
Underlying literature


End of Talk

Time for questions