Normalized Systems

Re-creating Information Technology based on Laws for Software Evolvability

Prof. dr. Herwig Mannaert
Management Information Systems Department

Contents

• Setting the Scene
• Supporting Theories
• Towards Evolvable Information Systems
• Towards Evolvable Enterprise Architectures
• Conclusions
The Business Challenge

- The **Agile Organization**
  - Continually scans its ecosystem
  - Reacts quickly to opportunities and is innovative
- Has 2 Characteristics
  - **Complexity**
    - Multi-channel vs. single channel
    - Diversify offerings/Additional services
  - **Change/Evolvability/Flexibility**
    - "These things are changing so fast it's invention in the hands of the owner." (Hansen et al., 2007)

The ICT Challenge

**The Law of Increasing Complexity**

Manny Lehman

"As an evolving program is continually changed, its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it."

Reference Frame

- Common ground between enterprises and IT is the structure or modularity
- Combination of modularity and change means \[\rightarrow EVOLVABLE MODULARITY\]
- Enterprises need to be architected, engineered, and designed
- Combination of design and change means \[\rightarrow DESIGN FOR CHANGE\]

Contents

- Setting the Scene
- Supporting Theories
  - Evolvability of Artefacts
  - Systems Theoretic Stability
  - Statistical Mechanics Entropy
- Towards Evolvable Information Systems
- Towards Evolvable Enterprise Architectures
- Conclusions
The Theories – Part 1

- Stability in System Dynamics:
  - In systems theory, the dynamic evolution of a system is studied based on a differential or difference equation.
  - A system is stable if and only if:
    • a bounded input results in a bounded output
    • it has poles in the left plane or inside the unit circle:
  - For a first order model, stability $\iff a < 0$:
    • $\frac{dy(t)}{dt} = x(t) + ay(t) \iff Y(s)/X(s) = 1/(s-a)$
    • $y[k+1]-y[k] = x[k] + ay[k] \iff Y(z)/X(z) = 1/(z-(1+a))$
  - This means that the increase cannot have a positive contribution from the size of the system.
Example: Enterprise Service Bus

- The effort to include an additional component may or may not vary with the system size

\[ \text{Impact} = N \quad \text{Impact} = 1 \]

Source: http://nl.wikipedia.org/wiki/Enterprise_Service_Bus

The Theories – Part 2

- Entropy in Thermodynamics:
  - The dynamic evolution of a system is represented by its entropy, a measure for how (dis)organized a system is
  - An isolated system will always increase its entropy, which basically represents the irreversibility in nature
  - In statistical thermodynamics, Boltzmann entropy is the number of possible microstates for a given macrostate, e.g.:
    - a number of coins with or without partitions
    - gas container with or without partitions
  - In information theory, Shannon defined entropy in a similar way as the expected value of uncertainty, i.e. inversely related to the amount of information we have:
    \[ - \sum_i p(x_i) \log(p(x_i)) \]
Example: Workflow Controllers

- The effort to debug a system after adding another component may or may not increase.

Entropy/Uncertainty = 4

Entropy/Uncertainty = 1

Contents

- Setting the Scene
- Supporting Theories
- Towards Evolvable Information Systems
  - Normalized Systems
- Towards Evolvable Enterprise Architectures
- Conclusions
Stability and Normalized Systems

- Systems theoretic stability:
  - bounded input results in bounded output for infinite time
- Software stability:
  - bounded amount changes results in bounded impacts for infinite time
- Assumption of unlimited system evolution:
  - number of all primitives and all dependencies between them become unbounded
- Normalized systems:
  - information systems that are stable wrt defined set anticipated changes
- Postulate:
  - Information systems need to be stable wrt defined set of anticipated changes

Model and System Stability

- $Y[k+1] - Y[k] = X[k] + aY[k]$
  - $Y[k]$: the number of all software entities at $k$, including the various versions
  - $X[k]$: the number of (versions of) software entities to be added to the system at $k$
  - $Y[k+1]$: the number of all software entities at $k+1$ when the system works again properly
  - Stability: the output function $Y$ stays bounded for every bounded input function $X$
  - $aY[k] = \text{combinatorial effects}$
Model and System Entropy

- **Macrostate**: an observable output and or state of the information system
- **Microstate**: the whole of all states and results of all software entities of the running system
- **Partitions**: software entities that externalize the system state of control and/or workflow, i.e. transactions

NS Principles

- **Modularity x Change** → **Combinatorial Effects (CE)**!
  - CE = (hidden) **coupling** or dependencies, increasing with size of the system!
  - **NS Principles** identify CE at seemingly orthogonal levels
    - **SoC**: Which tasks do you **combine** in a single module?
      - "An action entity can only contain a single task."
    - **DVT**: How do you **combine** a data and action module?
      - "Data entities that are received as input or produced as output by action entities, need to exhibit version transparency."
    - **AVT**: How do you **combine** 2 modules?
      - "Action entities that are called by other action entities, need to exhibit version transparency."
    - **SoS**: How do you **combine** modules in a workflow?
      - "The calling of an action entity by another action entity needs to exhibit state keeping."
  - CE are due to the way tasks, action entities and data entities are **combined or integrated**!
A necessary condition:
Fine-grained Modular Structure

E.g. SoC: a module can know only 1 technology
→ for every technology, a different module is required!

Building NS Applications

NS Application = \( n \) Instances of Elements

Universiteit Antwerpen
Contents

- Setting the Scene
- Supporting Theories
- Towards Evolvable Information Systems
  - Normalized Systems
- Towards Evolvable Enterprise Architectures
- Conclusions
Contents

• Setting the Scene
• Supporting Theories
• Towards Evolvable Information Systems
  - Normalized Systems
• Towards Evolvable Enterprise Architectures
• Conclusions

Summary

<table>
<thead>
<tr>
<th>Current</th>
<th>Step 1: Principles</th>
<th>Step 2: Elements</th>
<th>Step 3: Meet-in-The-Middle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lehman</td>
<td>Fine-grained</td>
<td>Expansion</td>
<td>Aggregate <strong>Reusable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>And <strong>Evolvable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Building Blocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>➔ <strong>DETERMINISM</strong> !</td>
</tr>
</tbody>
</table>