Emergence
- from the Point of View of Industrial Software

Frank J. Furrer
Prof h.c. TUD, Dr. sc. techn. ETH-Zürich

TU Dresden: Workshop «Emergence»
25. August 2016
Emergence is the manifestation of a system’s properties that cannot be anticipated in all cases from the properties of its constituent components or parts.
Example: Emergent property: „flying“

Constituent systems (CS) of an aircraft:
- engines
- body
- wings
- cockpit
- etc.

... none of the constituent systems is able to fly!

Assemble the essential constituent systems:

**Emerging property**: the assembly (= airplane) is able to **fly**!
Emergence?

Manifestation of a system’s properties that cannot be anticipated in all cases from the properties of its constituent components or parts.

Uncertainty, Risk, Accident Potential
Example: Emergent Property: Interface Failure

Constituent systems:
• Airplane (DC-8)
• Airport (Runway)

October 8, 1979: Swissair Flight 316 overran the Athens runway – 14 deaths

Cause: „Interface“ between the runway and the airplane
• Landing when braking action is less than good
• Crew mistakes
Objectives of this Presentation:

• Present the Industrial Software View of Emergence

• Show the potential Impact of Emergence on Industrial (= Cyber-Physical) Systems

• Investigate the Possibilities to manage Emergence

• Discuss open Questions
Content:

1. Definitions & Foundation
2. Industrial Software
3. Emergence
4. Impact Analysis
5. Managing Emergence
6. Conclusions
Definitions & Foundation
For emergent behaviour you need a *system of systems*

**System-of-Systems characteristics:**
1. Operational Independence of the Elements
2. Managerial Independence of the Elements
3. Evolutionary Development
4. Emergent Behavior
5. Geographic Distribution

[5 Maier criteria, 1998]

Total = more than the sum of its parts

Emerging behavior is the consequence of the *interactions* of the constituent components or parts of the system of systems
**SoS (System-of-Systems) Terminology**

- **Mission**
- **Constituent Systems (CS)**
- **Dependency**
- **System-of-Systems (SoS)**
- **Stakeholders**

**Lead System**
«GO» is a board game which was invented before 2'500 years in China.

Board: 19 x 19 lines, infinite number of white and black stones.

Objective: gain as much terrain as possible

Number of possible situations on the GO-board: $\sim 4,63 \times 10^{170}$

- Chess: $\sim 10^{43}$
- Number of atoms in the universe: $\sim 10^{80}$
Emergent Behaviour (Example 2/3)

Artificial Neuron

\[ y_i = f(\text{net}_i) \]

Neuronal Network
Impressive/frightening:
«AlphaGO» was not programmed, but was a learning program [Deep Learning] ⇐ Emergence!

March 2016: The AI-program «AlphaGO» wins against the multiple and current world champion Lee Sedol (4:1)
Cyber-Physical System

Processing

Sensors

Actuators

Cyber-World

Physical World
A cyber-physical system (CPS) consists of a collection of computing devices communicating with one another and interacting with the physical world via sensors and actuators in a feedback loop.
Example of a Cyber-Physical System of Systems (CPSoS):
Collaborating Robots
Industrial Software
Industrial Software

Processing

Sensors

Actuators

Software

http://www.dangerouscreation.com

25.08.2016
© Prof. Dr. Frank J. Furrer - Workshop Emergence
Industrial Software

• Mission-critical
• Safety-critical
• Real-time
• Security-critical
• etc.

Characteristics of Industrial Software

Software

```java
public class Arc {
    public String start;  public String end; }
```

```java
public class Colored {
    public String node;  public String color;
}
```
Emergence
Emergence

Manifestation of a system’s properties that *cannot* be anticipated in *all* cases from the properties of its constituent components or parts.
Emergent Behaviour Classification

Emergence

Desirable positive

Undesirable negative

Expected emergent behavior

Unexpected emergent behavior
### Emergent Behaviour Classification

<table>
<thead>
<tr>
<th>Emergence</th>
<th>Desirable positive</th>
<th>Undesirable negative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expected</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>emergent behavior</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Unexpected</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>emergent behavior</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# SoS Emergent Behaviour Classification

<table>
<thead>
<tr>
<th>Emergence</th>
<th>Desirable/positive</th>
<th>Undesirable/negative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expected</strong></td>
<td>Reason for building the SoS (SoS objective)</td>
<td><strong>Mitigate</strong> by appropriate design measures, such as threat/risk analysis and countermeasures</td>
</tr>
<tr>
<td>emergent behavior</td>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Unexpected</strong></td>
<td>Sometimes (however, quite rarely) an SoS shows unexpected, beneficial behaviour</td>
<td>Unexpected &amp; undesirable negative emergent behavior is one of the critical risks of most SoS</td>
</tr>
<tr>
<td>emergent behavior</td>
<td><img src="image3" alt="Smiley" /></td>
<td><img src="image4" alt="Image" /></td>
</tr>
</tbody>
</table>
SoS Emergent Behaviour Classification

deterministic

temporary  predictable

by degree (incremental)

expected  unexpected

unpredictable  persistent

discontinuous

stochastic
Impact Analysis
Which Impact has Emergence on our CPSoS?
# Workshop EMERGENCE

## Emergence

<table>
<thead>
<tr>
<th>Expected emergent behavior</th>
<th>Desirable positive</th>
<th>Undesirable negative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Software</strong></td>
<td><strong>Design Goal: Attain</strong></td>
<td><strong>Design Goal: Avoid</strong></td>
</tr>
</tbody>
</table>

### Expected emergent behavior

**Class Graph**
```java
class Graph {
    private Set<Arc> arcs = new HashSet<Arc>();
    ...
}
```

**Expected emergent behavior**

**Goal:**
- **Expected emergent behavior**

**Software**
```java
class Graph {
    private Set<Arc> arcs = new HashSet<Arc>();
    ...
}
```

### Unexpected emergent behavior

**Software**
```java
class Graph {
    private Set<Arc> arcs = new HashSet<Arc>();
    ...
}
```

**Goal:**
- **Unexpected emergent behavior**

**Software**
```java
class Graph {
    private Set<Arc> arcs = new HashSet<Arc>();
    ...
}
```
<table>
<thead>
<tr>
<th>Emergence</th>
<th>Desirable positive</th>
<th>Undesirable negative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expected</strong></td>
<td><img src="image1.png" alt="Good Job!" /></td>
<td><img src="image2.png" alt="ERROR!" /></td>
</tr>
<tr>
<td>emergent behavior</td>
<td>Missing Care!</td>
<td>Accidents</td>
</tr>
<tr>
<td><strong>Unexpected</strong></td>
<td><img src="image3.png" alt="Person Confused" /></td>
<td><img src="image4.png" alt="Bomb" /></td>
</tr>
<tr>
<td>emergent behavior</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Managing Emergence
Managing Emergence in Industrial Software

<table>
<thead>
<tr>
<th>Emergence</th>
<th>Desirable positive</th>
<th>Undesirable negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected emergent behavior</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Unexpected emergent behavior</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Vital Questions for SoS-Engineering:

1) How can we **avoid** **unexpected** emergent behaviour?

2) How can we **avoid** **undesirable** emergent behaviour?

3) How can we **prove** that there is no **unexpected** emergent behaviour in our system?

4) How can we **prove** that there is no **undesirable** emergent behaviour in our system?
Emergence is today an active research challenge in various fields.
Emergenent Properties:

What can we do in Industrial Software?

New Field: Systems-of-Systems Engineering (SoSE)
Managing Emergence in Industrial Software

⇒ CPSoS-Engineering
Interface Contracts

Collaboration Contract
- functional
- operational
- commercial
- legal

Governance Authority 1

Governance Authority 3

Collaboration Contract
- functional
- operational
- commercial
- legal

Governance Authority 2

Collaboration Contract
- functional
- operational
- commercial
- legal
Execute a formal **risk assessment**
(Various proven methodologies available)

Risk $\Rightarrow$ Assessment $\Rightarrow$ Mitigation (Software Development)
Monitoring

Continuously monitor Inputs/Outputs

Continuously monitor Interface Contracts
Conclusions
Conclusions

Emergence (emergent behaviour) can be a serious risk for Industrial Software

Emergence is not yet well understood

Potential Emergence in Industrial Software needs to be managed

Requires an effective SoSE-Process
Research in Emergence Management for Industrial Software is an interesting and fruitful Topic (Master, Diploma, PhD)
Thank You and Questions Please