Network-like System Relations and Their Meaning for IT-System Architecture

Johannes Reich, johannes.reich@sophoscape.de

2013-12-02
To be semantic or not to be?

Claude E. Shannon (1948): A Mathematical Theory of Communication

"The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have meaning; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem."

Frege Principle of Semantics

Two components are semantically equivalent if they can be exchanged.
We are living in an open world of network-like relations between systems.
We are living in an open world of network-like relations between systems: Focus on processes
Systems

Network-like System Relations

Inside: $X$

Outside: $f$

In

Out
Two Simple Example Systems

A simple multiplier

No internal state

Input state: \( x, y \in \{-2^{31} \ldots 2^{31} - 1\} \)

Output state: \( z \in \{-2^{31} \ldots 2^{31} - 1\} \)

System function: \( z' = f(x, y) = x \times y \quad //\text{ watch for overflow!} \)

A simple counter

No input and no internal state

Output state: \( z \in \{0 \ldots 2^{32} - 1\} \)

System function: \( z' = f(z) = z + 1, \quad //\text{ watch for overflow!} \)
System Composition/Super System Formation

Sequential Composition \((S_2 \circ S_1)\)

Parallel composition \((S_2 || S_1)\)
Richer Interaction Semantics

"Interaction-Diagramm"

System 1

Message $n_1, n_2, \ldots$

System 2

Message $m_1, m_2, \ldots$

System 1

System 2

System 1

 collaborate

System 2

System 3

determines
System 2

System 1_2

...
Recursive System Relations

System $\mathcal{U}_1$

```c
int fac1(int i) {
    if (i==0)
        return 1;
    else
        return i*fac2(i-1);
}
```

System $\mathcal{U}_2$

```c
int fac2(int i) {
    if (i==0)
        return 1;
    else
        return i*fac1(i-1);
}
```
We are living in an open world of network-like relations between systems: Focus on interactions
Tic tac toe as an extensive form game

1: 

\[ b[1] = 'x' \]

\[ \ldots \]

\[ b[9] = 'x' \]

2: 

\[ x \]

\[ \ldots \]

\[ 2: \]

\[ x \]

\[ \ldots \]

\[ \ldots \]

2: 

\[ b[2] = 'o' \]

\[ \ldots \]

\[ b[7] = 'x' \]

\[ \ldots \]

1 has won

\[ x \]

\[ o \]

\[ x \]

\[ o \]

\[ x \]
Tic tac toe as an interaction

Two systems playing tic tac toe. System 1 makes the initial move.
Games and Protocols/Processes

Relation between Games and Protocols

Protocols + Decisions = Games - Payoff

Consequences

- Game theory and protocol/process theory should use the same interaction model!
- Processes [in game-like interactions] interact via shared states (Shannon-channels).
- Prozesses [in game-like interactions] interact statefully.
Example: the Protocol of Mutual Exclusion
Causal Relation Between Output and Input of Different Systems - Channel Based Restriction
Example: Man in the Middle of the Protocol of Mutual Exclusion
Causal Relation Between Input and Output of the Same System - Condition Based Restriction
Semantic is key!

Common phrases and what they imply . . .

- "The process is in the objects"
- "message exchange patterns"
- "communication is about sending data from one system to another"
- "message based integration"
- "loose coupling is just asynchronous message exchange"
- "process interactions should be based on idempotent, stateless methods with no side effects"
- "the actual meaning of an interface is independent of its implementation"
- "for integration, write services, expose your object model!"
Architectural Principles

A process oriented application architecture is based on
- Clear system borders, clear layering
- A dedicated top process layer - separating reusable from non-reusable parts
- An internal event model - formalizing upward communication
- Sending/receiving documents - the basis for non-deterministic interactions
- Roles implementing protocols
- An adequate component model with protocol signatures

A process oriented application architecture simplifies
- Integration
- Reuse
- Security
Thank You!

Any questions?

Johannes.Reich@sophoscape.de
Literature


