Composition and Interoperability of IT-Systems

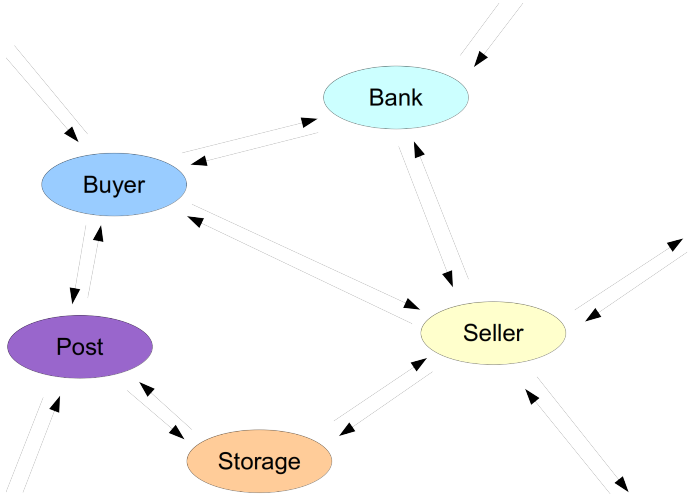
based on *"Komposition und Interoperabilität von IT-Systemen"*, Informatik Spektrum, Springer 2021 [6]

Johannes Reich, johannes.reich@sap.com

2024-07-04

Content

Motivation (I): The world is a network of interactions



Motivation (II)

Kenneth J. Schlager (1956): *"The first need for systems engineering was felt when it was discovered that satisfactory components do not necessarily combine to produce a satisfactory system. In a complex system it is often found that even though individual components satisfy all specifications, the system as a whole will not work."* [7]

The German government (2019) names *"interoperability"* as the third pillar of the design of digital ecosystems alongside sovereignty and sustainability in its "Leitbild 2030"[1]

(One) problem: Leslie Lamport's "Whorfian syndrome": What are agents, programs, processes, calculi, services, objects, classes, methods, digital twins, asset adminstration shells, etc.? [4]

Necessary milestones for a better understanding of 'Interoperability':

1. What entities are we talking about?
2. What do we precisely mean by 'x interoperates with y' where $x, y \in \{\text{all entities we are talking about}\}$?

Interoperability - Definition

lat. opera "work" und inter "in between"

My definition:

Two systems are interoperable if they are able to work together by interaction according to a defined success criterion.

I.e. we must know:

- What is a *system*?
- What is an *interaction*?
- What is a *successful* interaction?

What do we mean by "success"?

The answer: System composition.

Be $\mathcal{S}_1, \ldots, \mathcal{S}_n$ n systems, then we can denote their composition into a supersystem $\mathcal{S}_{tot}$ by means of a corresponding composition operator $comp_{\mathcal{S}}$ as a partial function on systems:

$$\mathcal{S}_{tot} = comp_{\mathcal{S}}(\mathcal{S}_1, \ldots, \mathcal{S}_n). \tag{1}$$

1st benefit: classification of system properties

A property $\alpha : S \to A$ of a system $\mathcal{S} \in S$ ...

... is a partial function which attributes values of some attribute set A to a system $\mathcal{S} \in S$.

Compositional vs. emergent properties

I call a property α of a composed system $\mathcal{S}_{tot}$ *"(homogeneous) compositional"* with respect to the composition $comp_{\mathcal{S}}$, if there exists an operator $comp_{\alpha}$ such that

$$\alpha(comp_{\mathcal{S}}(\mathcal{S}_1, \ldots, \mathcal{S}_n)) = comp_{\alpha}(\alpha(\mathcal{S}_1), \ldots, \alpha(\mathcal{S}_n)) \tag{2}$$

Otherwise I call this property "*emergent*".

Examples

- Mass of a system: $m_{tot} = m_1 + \ldots m_2$. $\Rightarrow$ mass is compositional.
- Resonant frequency of an LC-circuit: $f_0 = \frac{2\pi\sqrt{LC}}{1}$. $\Rightarrow$ f_0 is emergent.

Computability as a compositional property [3]

Starting from given set of elementary operations, all further computable operations can be constructed by the following 3 rules:

1. *Comp:* parallel and sequential composition: $f = h(g_1, \ldots, g_n)$
2. *PrimRec:* 'For-loop' .
3. μ-*Rec:* 'While-loop'

2nd benefit: interfaces and components

An interface of a system ...

... sums up everything a composition operator needs to know about a system.

$\Rightarrow$ The interface notion becomes decidable. The recipe is:

1. provide a system model,
2. show what the composition operator looks like, and
3. show which parts of the system model belong to the interface.

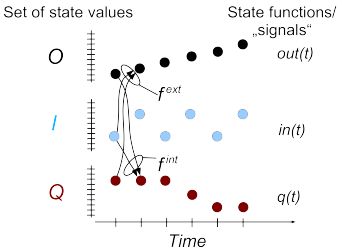
A component ...

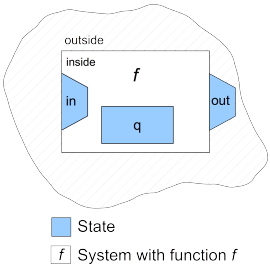
1. ... is a system that is intended for a particular composition and therefore has correspondingly well-defined interfaces that, by definition, express the intended composition.
2. ... represents a complexity border.

States and systems and time

A **state [or signal]** is a function $T \to D$

A **system** consists of states (in, out, q) and a system function.





A **(simple) discrete system** is characterised by its system function $f : Q \times I \to Q \times O$ with $f = (f^{int}, f^{ext})$ s.t.

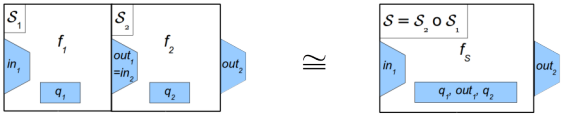
$$\begin{pmatrix} q(t') \\ out(t') \end{pmatrix} = \begin{pmatrix} f^{int}(q(t),i(t)) \\ f^{ext}(q(t),i(t)) \end{pmatrix}$$

or

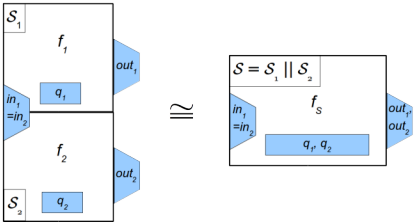
a (det.) transition relation $\Delta \subseteq I \times O \times Q \times Q$ where we write instead of $(i, o, p, q) \in \Delta$ also $p \overset{i/o}{\to} q$.

System composition - computability

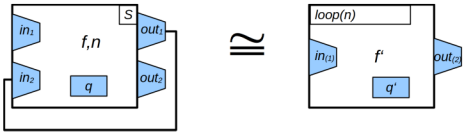
1a) Sequential composition



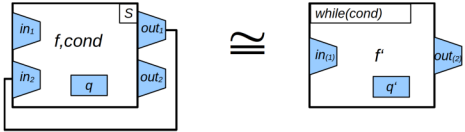
1b) Parallel composition



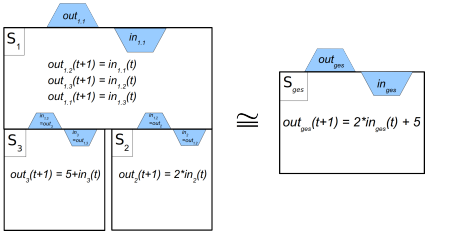
2) Primary recursive composition



3) μ-recursive composition



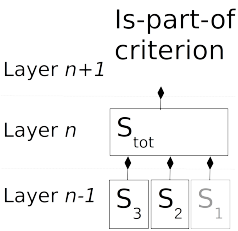
A simple composition example: $f(x) = 2x + 5$



≅

C-Code:

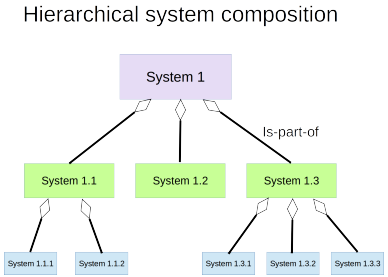
```
int S2(int n) {return(2*n);}
int S3(int n) {return(n+5);}
int S(int n) {return(S3(S2(n)));}
```



Conclusion

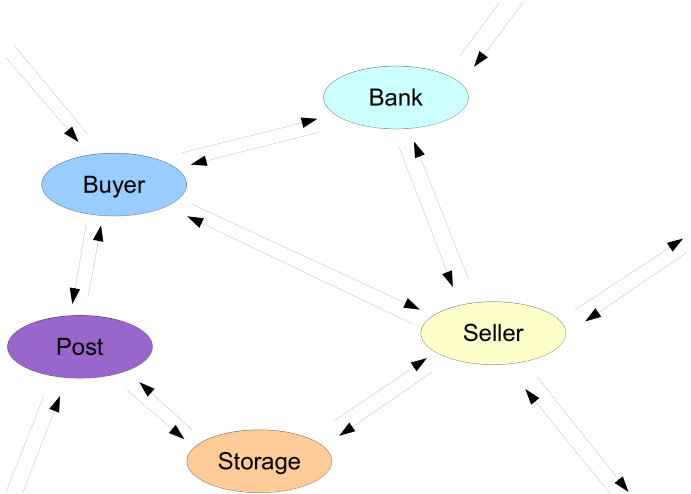
- There is no system $\mathcal{S}_1$ mentioned in our detailed C-code.
- There is no interaction between $\mathcal{S}_{tot}$ and $\mathcal{S}_1$, $\mathcal{S}_2$ or $\mathcal{S}_3$.
- $\Rightarrow$ The interface concept "operation" is "compositional" and not "interactional", i.e. it always represents the complete system in a compositional sense and hides any interaction.

(Hierarchical) system composition - summary

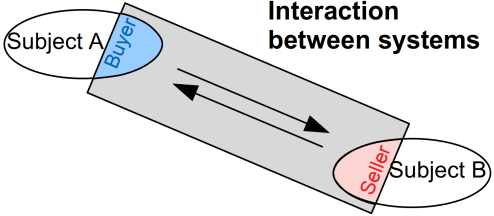


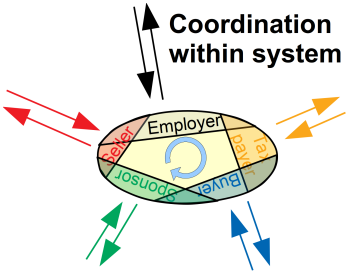
- The interaction of systems may lead to the creation of super systems, if we can identify the super system function.
- The hierarchy is an immediate consequence of the homogenity of this form of composition: systems become parts of super systems.
- There is **no** interaction between the super system and its subsystems!

Interactive systems in nonhierarchical interaction networks



The role concept of interactive systems: interaction and coordination



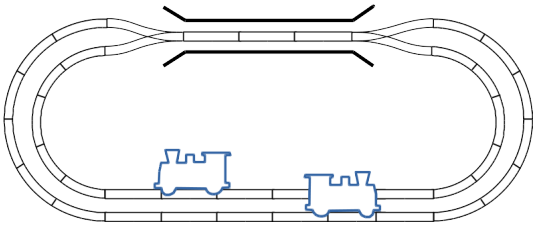


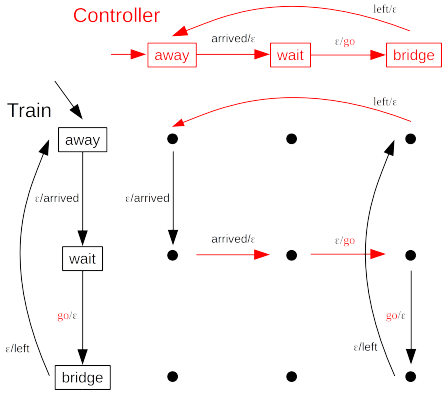
A role is the projection of an interactive system onto its interactions.

External coupling of roles by interaction: protocols

Composition of roles to protocols

$$\mathcal{P} = C_{\mathcal{R}}^{Prot}(\mathcal{R}_1, \ldots, \mathcal{R}_n) \tag{3}$$

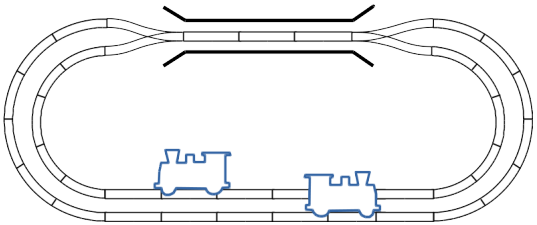


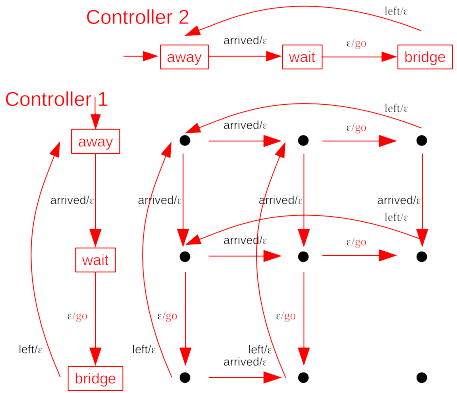


Internal coupling of roles: coordination

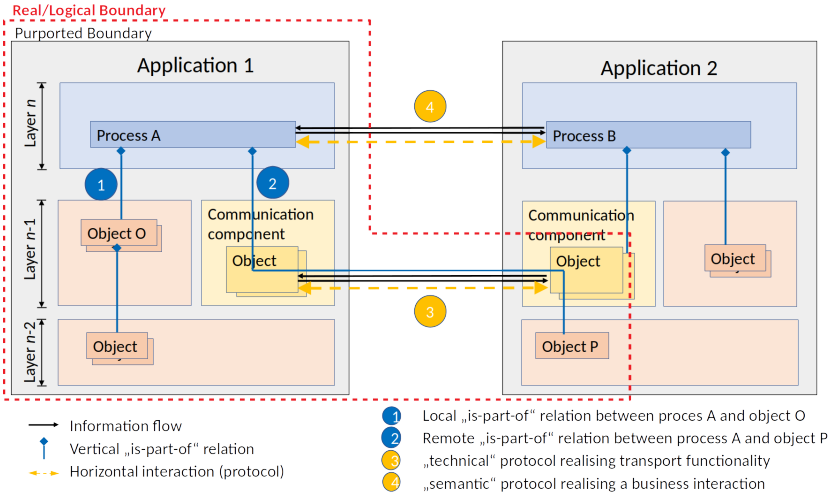
Composition of roles to systems

$$\mathcal{S} = C_{\mathcal{R}}^{Coord}(\mathcal{R}_1, \ldots, \mathcal{R}_n) \tag{4}$$





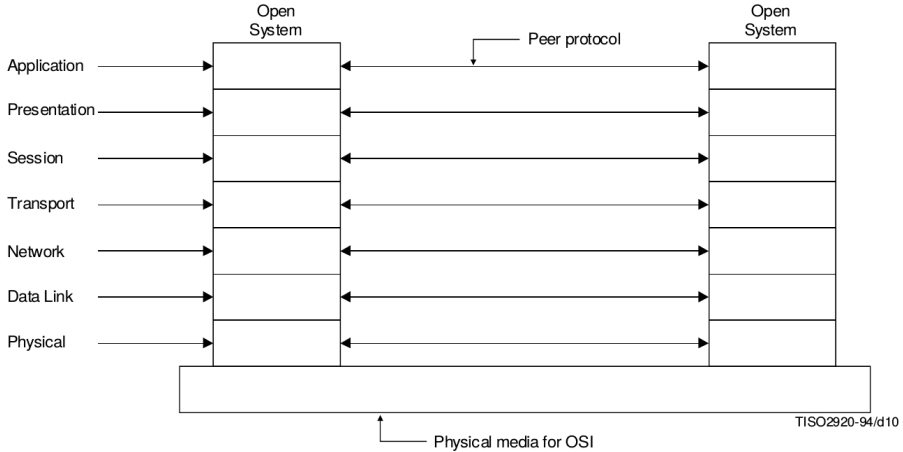
Correctly layered application model



Properties of the interfaces of operation and role

	Operation	**Rolle**
Composes to ...	... more complex operations (homogeneous composition)	... Protocols (inhomogeneous composition)
Transformation	Deterministic	Non-deterministic
State	generelly stateless; state is only temporarily required for recursion.	generally stateful, as non-determinism together with statelessness implies randomness. Also, without state, the internal coordination of roles doesn't make sense.
'Intelligence'	lies within the composition itself, hence in the developer.	Lies in the decisions and coordination of roles.
Hides:	the structure of the calculation (the 'algorithm')	How decisions are taken and how coordination is performed.
Reveals:	the (deterministic) mapping behavior (system function)	the (non-deterministic) mapping behavior (rules according to which the sytem behaves)
Laterality	Unilateral: Interface relates only to the 'system of interest'	multilateral: the consistency of a protocol interaction rests on all roles of a protocol.
Is the base for ...	... Reuse; (vertical) 'is-part-of' relation within a system hierarchy.	... (horizontal) interactions between independent systems.

Ref. Model of Open Systems Interconnection (OSI-Model)



The inconsistencies of the OSI-model

- *"[The OSI-model] is concerned only with the exchange of information between open systems"* - and not with the internal *"coordination [of] its interactions"/"internal functioning of each individual real open system)"* ([2], sections 4.2.6 & 7).
  - ↯ Without reference to the structure of the processing, no statement about the structure of the application is possible.
- Ambiguity regarding hierarchy: interacting systems vs. "is-part-of" relationship.
  - ↯ Assumption that applications always relate to each other on the same level is wrong.
- The state of interaction cannot be assigned to a single session layer in every case.
  - **Vertical "is-part-of" relation**: Here there may be a communication state that can be held in the communication layer ("context of use").
  - **Horizontal interaction**: Is by definition stateful.

The concept of a SOA-Service: well or ill-defined?

WSDL 1.1: defines the "operation" of a service by 4 "transmission primitives" a so called "endpoint" has to support. No reference to any transformational semantics

- "One-way": The "endpoint" receives a message.
- "Request-response": The "endpoint" receives a message and sends back a correlated message.
- "Solicit-response": The "endpoint" sends a message and receives a correlated message.
- "Notification": The "endpoint" sends a message..

WSDL 2.0: defines (Section 2.2.1) an "Interface component" as a "*sequences of messages that a service sends and/or receives*". According to WSDL 2.0, an "operation" is an "*interaction with the service consisting of a set of (ordinary and fault) messages exchanged between the service and the other parties involved in the interaction*". No reference to any transformational semantics

BPMN 2.0.2: Is based on WSDL 2.0 as non-normative reference. Defines (section 8.5.3) "*An Operation defines Messages that are consumed and, optionally, produced when the Operation is called.*" No reference to any transformational semantics

A: No.

The concept of a REST-service: well or ill-defined?

A REST-service is supposed to follow 2-3 "principles":

- Addressability: each resource has to have a unique URI.
- Statelessness: each REST-message is supposed to contain all the information that is necessary for the processing which it initiates.
- Idem potency (sometimes): the called service is supposed to have an identical effect, no matter when it is called.

Strange consequences if we transfer "principles" that simplify information transportation to the domain of information processing. Example Bank transfer: The bank would not know my account balance...

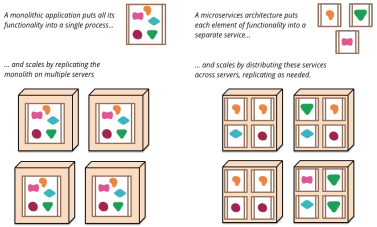
A: No.

The concept of a microservices: well or ill-defined?

Properties of a microservice architectur [5]

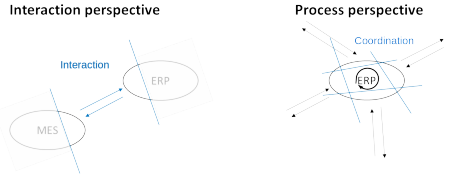
- Componentisation via Services (Libraries vs. Services)
- Organised along business capabilities (Conway Law, 1967)
- Products, not projects (full lifecycle ownership)
- Smart endpoints, dumb channels
- Decentralised governance (teams take responsibility for development and operation of their software)
- Decentralised data management
- Infrastructure automation
- Robust design
- Evolutionary design

A: No.



Source:http://martinfowler.com/articles/microservices.html, Figure 1: Monoliths and Microservices

Idea of an **"interaction oriented application architecture"**



What's the problem with determining the functions of processes explicitely a la "workflow"?

1. Building interactive IT systems: tension between (1) describing systems only via functions $\Leftrightarrow$ (2) these systems integrate into interaction networks only via non-deterministic, stateful, and asynchonous interactions.
2. Forcing everything in an explicitely formulated system function a la "workflow" destroys all consistency guarantees of the roles.

$\Rightarrow$ Idea of an **"interaction oriented application architecture"** where coordination complements interaction.

Some immediate consequences

1. Different composition classes require different interface classes.
2. Operations support "is-part-of" relation. "There is no way out of a system by calling an operation". Operations are of little value for describing horizontal interactions (IoT)
3. Roles are the interfaces of loose coupling, composing externally to protocols
4. Roles can also compose internally by coordination to systems ("interaction oriented architecture").
5. Component models that do not support role declarations are incomplete.
6. Without clear interfaces it is practically impossible to construct robust interaction-oriented, concurrent applications: "A protocol that is not validated is erroneous."
7. What is desperately needed: Interface repositories for roles.

Thank You!

Questions?

[1] *Leitbild 2030 für Industrie 4.0, Digitale Ökosysteme global gestalten*.
Bundesministerium für Wirtschaft und Energie (BMWI), 2019.

[2] Information Technology - Open Systems Interconnection – Basic Reference Model.
http://handle.itu.int/11.1002/1000/2820, 1994.
Früher "CCITT Recommendation", Identischer Standard: ISO/IEC 7498-1:1994 (Common).

[3] S. Kleene.
General recursive functions of natural numbers.
Mathematische Annalen, 112(5):727–742, 1936.

[4] L. Lamport.
Computer science and state machines.
In *Concurrency, Compositionality, and Correctness*, pages 60–65. Springer, 2010.

[5] J. Lewis and M. Fowler.
Microservices, a definition of this new architectural term.

https://martinfowler.com/articles/microservices.html, called 2017-03-10, March 2014.

[6] J. Reich.
Komposition und Interoperabilität von IT-Systeme.
Informatik Spektrum, 44:339–346, 2021.

[7] K. J. Schlager.
Systems Engineering - Key to Modern Development.
IRE Transactions on Engineering Management, pages 64–66, 1956.