

# The Future of Software and Services

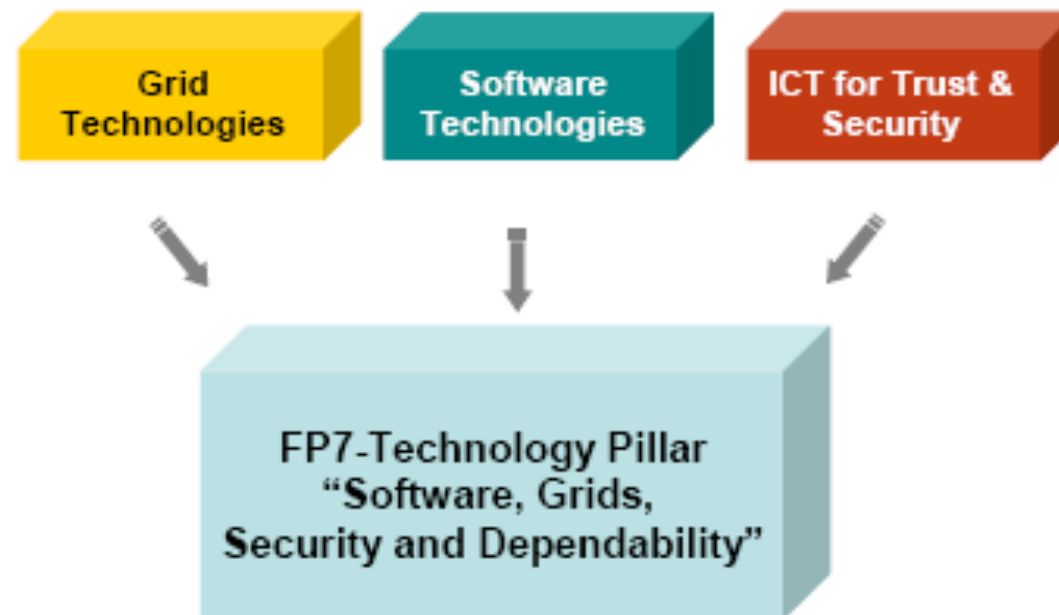
SGSD Technology Pillar

Günter Böckle, Siemens AG



Software &  
Engineering  
System and  
Software  
Processes

## Background

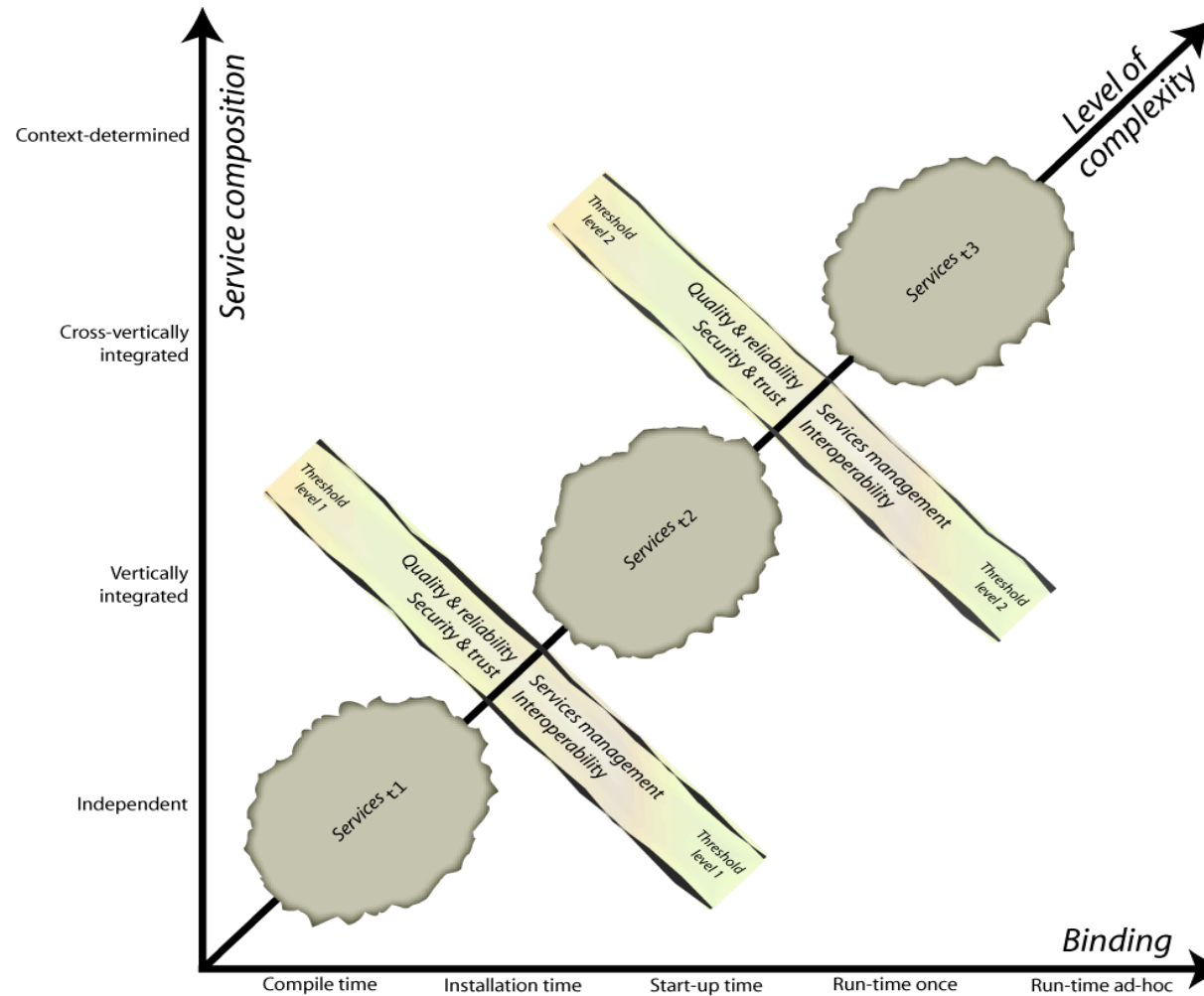


- **Plan:** Combining the three areas **Software, Grid, Security, and Dependability** into one technology pillar
- **Task:** Definition of scope and structure for research in this area
- **Result:** first version of background document prepared



Software &  
Engineering  
System and  
Software  
Processes

# Development of Services



Software & Engineering System and Software Processes

## Thesis (intentionally a bit provocative)

- **All major ideas for SW engineering that have been realized in the 90's and 00's come from the 60s and 70s:**
  - OO: Classes, objects, methods, parallel processing: all in Simula, 1962; information hiding: Parnas, 60's
  - Java: Concepts from Simula, 1962 (compiler prototype 1964, on Univac)
  - Product families: Parnas 1976
  - Commonality & variability: Dijkstra 1972, Parnas 1976
  - SW services: ISO OSI in the 70's: each layer uses only services from the level below and provides services for the layer above
  - DLLs: TOPS20 (OS on pdp10), 1981



## Conclusions, Goals

- **Which ideas – perhaps from the last 20 years – will (or need to) be realized in the next 10 – 20 years?**

### Goals:

- **Provide Qualities**
  - Adaptability - to user, to environment, to situation
  - Pervasiveness – getting a service wherever I am
  - Security
  - Dependability (reliability, availability)
  - Interoperability
  - Usability
- **Trust**
- **Virtualisation (of all kinds of resources), end-to-end**
- **Handling complexity**
- **Semantics**



Software &  
Engineering  
System and  
Software  
Processes

## Adaptability, Availability, and Reliability

- **Whenever a SW tries adapting to me I get confused – sometimes angry**
    - Adaptation must be comprehensible and traceable
    - User must be able to enforce or prevent it
  - **What do reliability and availability mean for changing SW and services?**
    - Reliability = constantly performing according to requirements – but the requirements did not specify the changes for adaptation!
    - Availability: what does it mean when the originally defined service is no longer available, but different adapted ones?
- ➔ **New definitions of availability and reliability**
  - ➔ **New approaches and techniques for requirements engineering**
  - ➔ **New approaches for ensuring availability and reliability**



Software &  
Engineering  
System and  
Software  
Processes

## Pervasiveness

- **SW & Services can run anywhere – users just want the end-service**
  - **Virtualisation of all kinds of resources**
    - Architectures for virtualisation, e.g. scalable service architectures
    - Grid access everywhere
    - Middleware
    - Interoperability
    - Mapping-services: national, regional specificities (legal, procedural, ...)
    - Adequate business models
    - Open standards
    - Open source



## Trust

- **Trust encompasses**
  - Dependability (reliability, availability), security, safety
  - Honesty of service providers and creators
  - Comprehensibility
  - User participation
- **Open standards**
- **Open source**
- **Transaction behaviour**
- **Authenticity – a service must do what it claims to do**
- **Fail-safe grid architectures**
- **User-constructed services**
- **Certification**



## Research Fields

- **Services science**
  - What services do users want?
  - What service behaviour annoys users?
  - Cultural, regional, sociological patterns and peculiarities
  - Adequate business models
- **Business models**
  - Adaptive business models
  - User interaction support
- **Managing complexity**



Software &  
Engineering  
System and  
Software  
Processes

## Research Fields

- **SW and Systems engineering**
  - Development processes: esp. for collaboration, cyber enterprises
  - Development approaches that support abstraction, like MDD, AOP, ...
  - Dealing with evolution and degradation
  - Variability management
  - Requirements engineering: methods considering design-time and run-time requirements and emergent behaviour
  - Mapping requirements to architecture

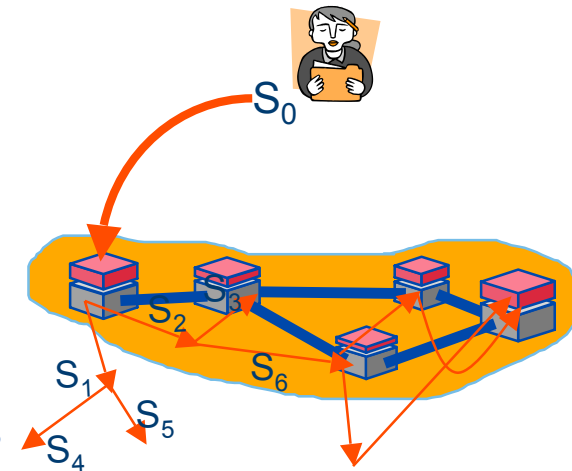


Software &  
Engineering  
System and  
Software  
Processes

## Research Fields

### • Scenario:

- User calls Service  $S_0$
- $S_0$  calls  $S_1$  and  $S_2$  - A cascade of services is invoked
- Traces of these services occur in many nodes of the network(s)
- If  $S_0$  completes correctly, all traces except the intended ones in the network(s) have to be eliminated
- If  $S_0$  does not complete correctly, all changes made at any place in the network must be undone to the state before  $S_0$  was called
- Some of the cancelled transactions will create state changes (e.g. billing for performed DBMS accesses – have to be paid even if transaction aborts)



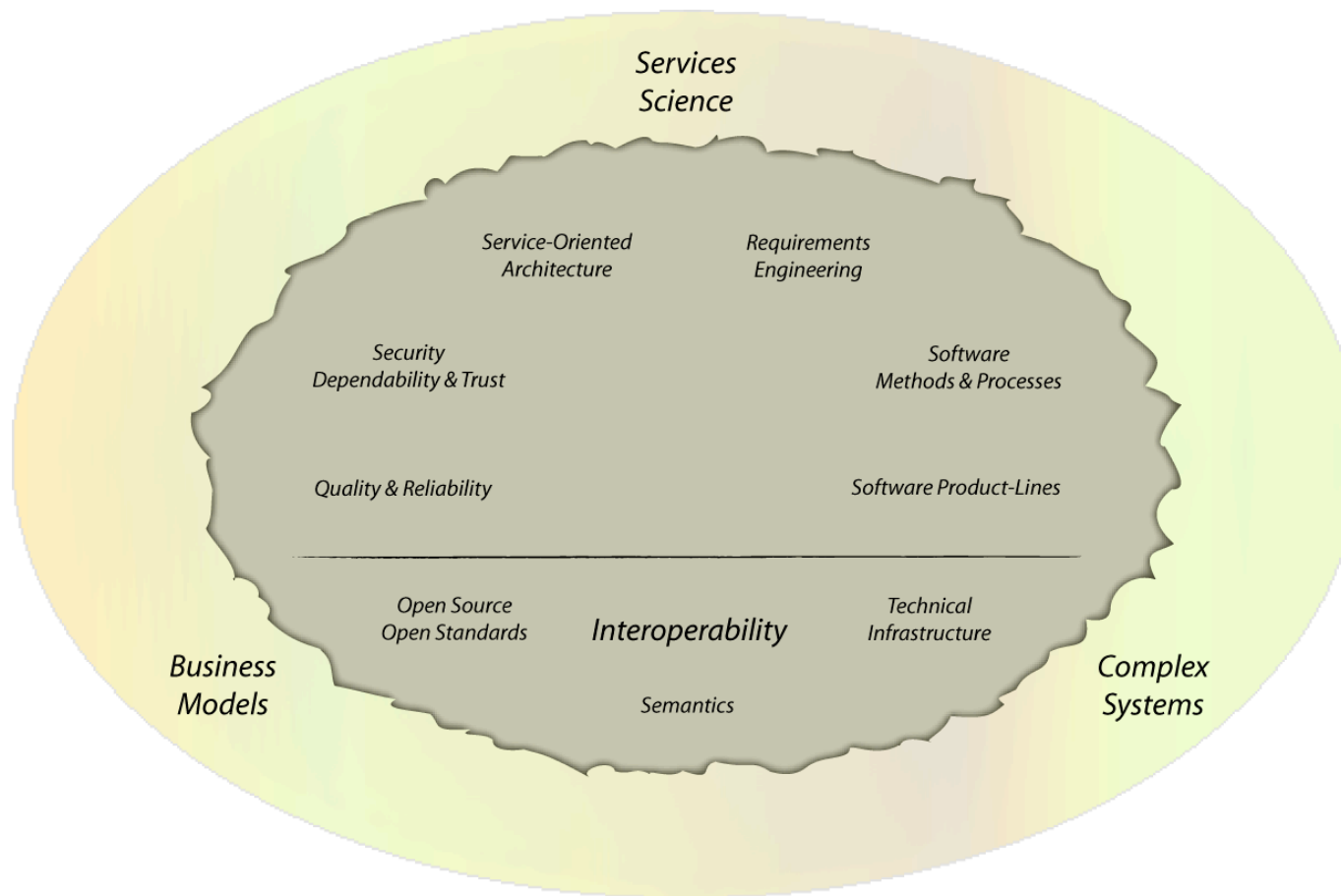
## Research Fields

- **Transaction theory and technique**

- The transaction mechanisms known from DBMS and banking must be extended for services
- We need a new theory of transactions – covering networks, middleware, protocols, applications, etc.
- New logging and roll-back approaches needed
- The BPEL scope (context for activities where compensation and fault handlers are defined) must be put into practice



## Conclusion: Research Areas



Software & Engineering System and Software Processes

See report at [ftp://ftp.cordis.lu/pub/ist/docs/directorate\\_d/st-ds/fp7-report\\_en.pdf](ftp://ftp.cordis.lu/pub/ist/docs/directorate_d/st-ds/fp7-report_en.pdf)