Modular Construction of Systems and Layered System Architectures by Rich Subsystem Specifications of Cyber-Physical Systems

Manfred Broy

Technische Universität München
Institut für Informatik
D-80290 Munich, Germany

From mechatronics to embedded systems …

Mechatronics (Wikipedia)

"*Mechatronics is a multidisciplinary field of engineering, that is to say, it rejects splitting engineering into separate disciplines. Originally, mechatronics just included the combination of mechanics and electronics, hence the word is a combination of mechanics and electronics* …"

Embedded System (Wikipedia)

"*An embedded system is a computer system designed for specific control functions within a larger system, often with real-time computing constraints*."

… and further to CPS

Cyber-Physical System (US NSF):
"*Cyber-physical systems (CPS) are engineered systems that are built from and depend upon the synergy of computational and physical components. Emerging CPS will be coordinated, distributed, and connected, and must be robust and responsive.*
The CPS of tomorrow will need to far exceed the systems of today in capability, adaptability, resiliency, safety, security, and usability."

The acatech position on CPS

"*Cyber-physical systems are systems with embedded software (as part of devices, buildings, means of transport, transport routes, production systems, medical processes, logistic processes, coordination processes and management processes), which:*

- *directly record physical data using sensors and affect physical processes using actuators;*
- *evaluate and save recorded data, and actively or reactively interact both with the physical and digital world;*
- *are connected with one another and in global networks via digital communication facilities (wireless and/or wired, local and/or global);*
- *use globally available data and services;*
- *have a series of dedicated, multimodal human-machine interfaces.*"

Cyber-physical systems: key properties and challenges

- Physicality
  - real time
  - probability
  - location awareness
  - context awareness
  - real world awareness
  - …
- Connectivity
  - connected to cloud services
- Systems of systems
  - Sub-system (de)composition
  - Service (de)composition
- Interoperability
  - Service platforms
- Openness
  - security
- HMI
  - Human Centric Engineering
- Dynamic systems
  - Dynamic interfaces
  - Dynamic architectures
  - Dynamic change of behavior (adaptivity)
- Mobile systems
  - space awareness
- Uncertainty

Cyber-physical systems: innovation

The two sides of CPS

The internet becomes real world aware

Embedded systems get connected to cloud services

From closed embedded systems to open systems connected to the cloud

Traditional embedded systems

- closed
- real time
- connected to the physical
- reliable
- high safety reqs
- low security reqs

Services in the cloud

- open
  - open interfaces
- restricted availability
- easy extendibility
- high interoperability
- low safety reqs
- high security reqs

"Smart" systems - Cyber-physical systems: focus points of innovation

- adaptive
- context aware
- autonomous
- big data
- open interfaces
- dynamic
- tactile internet

What is seamless model based development?

- Development by a chain of models
  - High expressive power
  - Clear structure of role of models
  - All aspects in the development captured by models
  - Tight integration in the artifact models
- Development steps by well-defined relationship between modes
  - Refinement
  - Decomposition
  - Change of scope
- Extended tool support
  - High automation
  - All artifacts in tools and comprehensive data base (development back bone)

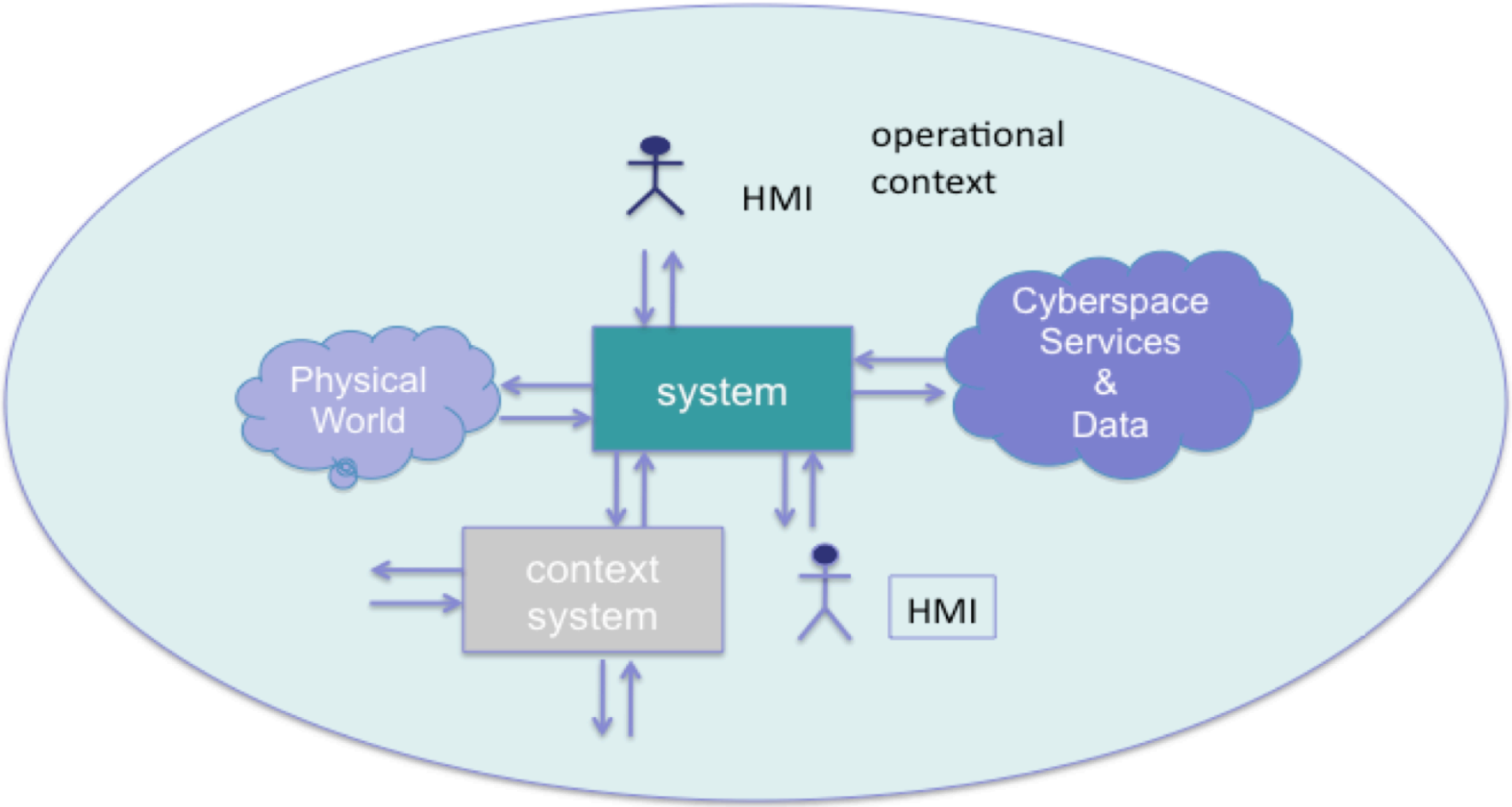
Essentials for Modeling Concepts for Seamless Modeling

- Expressive and tractable modeling concepts
- Clear understanding of relationship between modeling concepts
- Notion and understanding of integration of modeling concepts
- Tracing between the elements of modeling concepts
- Expressive power: relevant properties expressible
- Modularity and composition of modeling concepts
- Variety of forms of representation
  - Mathematical models
  - Logic
  - Intuitive and tractable representation: graphics, diagrams, tables, ...

Basic System Modeling Concepts

Technische Universität München
Institut für Informatik
D-80290 Munich, Germany

System and its context



Basic System Notion: What is a discrete system (model)

A system has

- a system boundary that determines
  - what is part of the systems and
  - what lies outside (called its context)
- an interface (determined by the system boundary), which determines,
  - what ways of interaction (actions) between the system und its context are possible (static or syntactic interface)
  - which behavior the system shows from view of the context (interface behavior, dynamic interface, interaction view)
- a structure and distribution addressing internal structure, given
  - by its structuring in sub-systems (sub-system architecture)
  - by its states und state transitions (state view, state machines)
- quality profile
- the views use a data model
- the views may be documented by adequate models

Discrete systems: the modeling theory – interface behaviour as key concept

Sets of typed channels

$I = \{x_1 : T_1, x_2 : T_2, \ldots\}$

$O = \{y_1 : T'_1, y_2 : T'_2, \ldots\}$

syntactic interface

$(I \blacktriangleright O)$

data stream of type T

$STREAM[T] = \{IN\backslash\{0\} \rightarrow T^*\}$

valuation of channel set C

$IH[C] = \{C \rightarrow STREAM[T]\}$

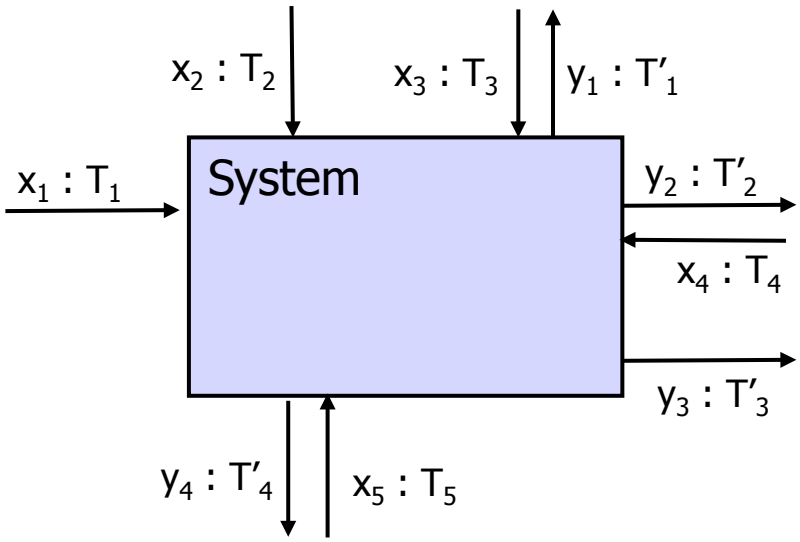
interface behaviour for syn. interface $(I \blacktriangleright O)$

$[I \blacktriangleright O] = \{IH[I] \rightarrow \wp(IH[O])\}$

interface specification

$p: IH[I] \times IH[O] \rightarrow IB$

represented as interface assertion S
logical formula with channel names as variables for streams



See: M. Broy: A Logical Basis for Component-Oriented Software and Systems Engineering. The Computer Journal: Vol. 53, No. 10, 2010, S. 1758-1782

Modeling Interface Behavior

Technische Universität München
Institut für Informatik
D-80290 Munich, Germany

System interface behaviour - causality

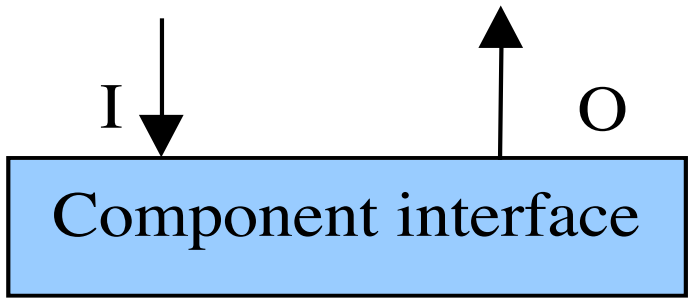
$(I \blacktriangleright O)$ — *syntactic interface* with set of input channels I and of output channels O

$F : IH[I] \rightarrow \wp(IH[O])$ — *semantic interface* for $(I \blacktriangleright O)$ with *timing property addressing strong causality* (let $x, z \in IH[I]$, $y \in IH[O]$), $t \in \mathbb{N}$):

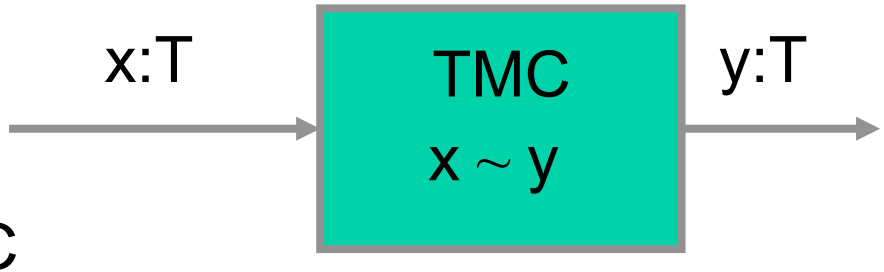
$$x{\downarrow}t = z{\downarrow}t \Rightarrow \{y{\downarrow}t+1 : y \in F(x)\} = \{y{\downarrow}t+1 : y \in F(z)\}$$

$x{\downarrow}t$ — prefix of history x of length t

A system shows a total behavior



Example: System interface specification



A transmission component TMC

TMC

in	x: T
out	y: T
x ~ y	

$x \sim y \equiv (\forall\ m \in T: m\#x = m\#y)$

Specifying interface assertion

Verification: Proving properties about specified systems

From the interface assertions we can prove

- Safety properties

$$m\#y > 0 \wedge y \in TMC(x) \Rightarrow m\#x > 0$$

- Liveness properties

$$m\#x > 0 \wedge y \in TMC(x) \Rightarrow m\#y > 0$$

Verification: adding causality – consistent time flow

From the interface assertions we can derive properties!

Specification:

$$y \in TMC(x) \Rightarrow (\forall\ m \in T\colon m\#x = m\#y)$$

Strong causality:

$$x{\downarrow}t = z{\downarrow}t \Rightarrow \{y{\downarrow}t{+}1\colon y \in TMC(x)\} = \{y{\downarrow}t{+}1\colon y \in TMC(z)\}$$

From which by choosing z such that

$$\forall\ m \in T\colon m\#(z{\uparrow}t) = 0$$

we conclude (note then $x{\downarrow}t = z{\downarrow}t \Rightarrow m\#(x{\downarrow}t) = m\#z$) and thus

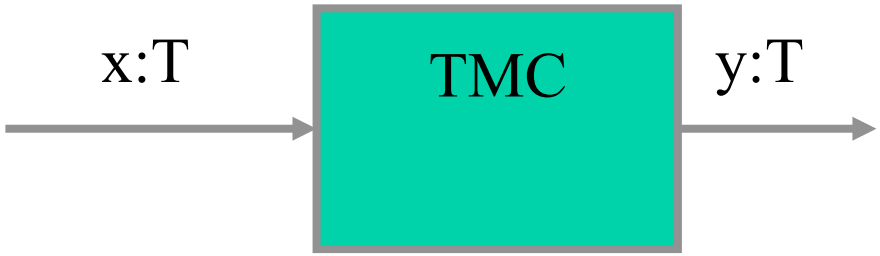
$$x{\downarrow}t = z{\downarrow}t \Rightarrow$$

$$\{y{\downarrow}t{+}1\colon y \in TMC(x)\} = \{y{\downarrow}t{+}1\colon y \in TMC(z)\} = \{y{\downarrow}t{+}1\colon y \sim z)\}$$

and thus

$$y \in TMC(x) \Rightarrow \forall\ t \in Time,\ m \in T\colon m\#(y{\downarrow}t{+}1) \leq m\#(x{\downarrow}t)$$

Example: TMC with Timing Restrictions



TMC

in x: T **out** y: T
$\forall\, t \in IN: \forall\, m \in T:$ $m\#(y{\downarrow}t+delay) \leq m\#(x{\downarrow}t) \leq m\#(y{\downarrow}t+delay+deadline)$

A flexible model of time

- Time is a key issue in embedded systems:
- Dealing with timing properties
  - Specification
  - Analysis
  - Verification
    - Analysis
    - Testing
    - Model checking
    - Deduction based verification
- Transforming time
- Dedicated models of time
  - Micro/Macro Step
  - Perfect synchrony
  - Scheduling
- Abstractions

See:
M. Broy: Relating Time and Causality in Interactive Distributed Systems.
In: M. Broy, W. Sitou, T. Hoare (eds): Engineering Methods and Tools for Software Safety and Security, NATO Science for Peace and Security Systems, D: Information and Communication Security, Vol. 22, IOS Press 2009, 75-130

Continuous systems: the model

Sets of typed channels

$I = \{x_1 : T_1, x_2 : T_2, \dots\}$

$O = \{y_1 : T'_1, y_2 : T'_2, \dots\}$

syntactic interface

$(I \blacktriangleright O)$

continuous data stream of continuous type M

$ConSTREAM[T] = \{IR_+ \rightarrow M\}$

valuation of channel set C

$CIH[C] = \{C \rightarrow ConSTREAM[T]\}$

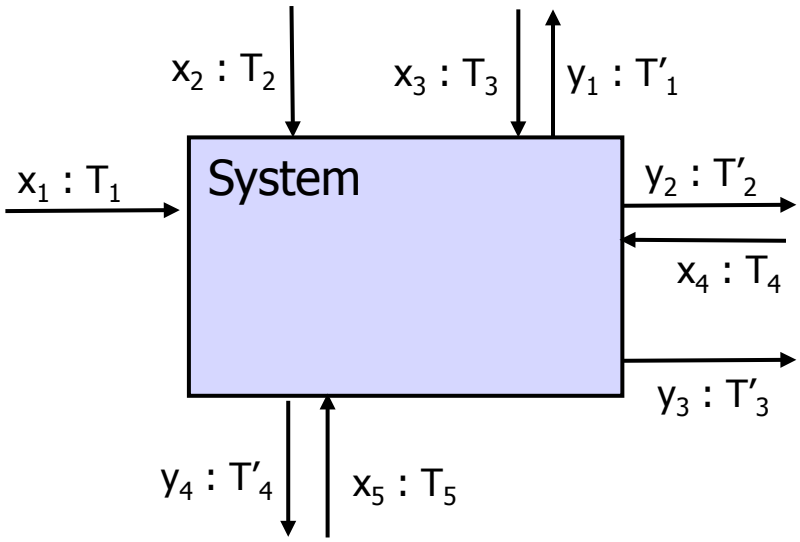
interface behaviour for syn. interface $(I \blacktriangleright O)$

$[I \blacktriangleright O] = \{CIH[I] \rightarrow \wp(CIH[O])\}$

interface specification

$p: CIH[I] \times CIH[O] \rightarrow IB$

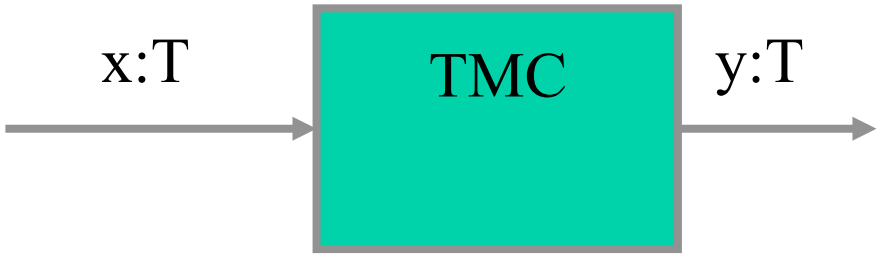
represented as interface assertion S
logical formula with channel names as variables for continuous streams



See: M. Broy: System Behavior Models with Discrete and Dense Time. In: S. Chakraborty, J. Eberspächer (Eds.): Advances in Real-Time Systems. Springer 2012

Specification with Probabilities

Example:
TMC with Probability Restrictions



TMC

in x: T **out** y: T
$\forall\, t \in IN: \forall\, m \in T:$ $\mathbf{P}(m\#(x{\downarrow}t) \leq m\#(y{\downarrow}t+delay+deadline)) \geq 0.8$

Discrete systems: the modeling theory - probability

Sets of typed channels

$I = \{x_1 : T_1, x_2 : T_2, \ldots \}$

$O = \{y_1 : T'_1, y_2 : T'_2, \ldots \}$

syntactic interface

$(I \blacktriangleright O)$

data stream of type T

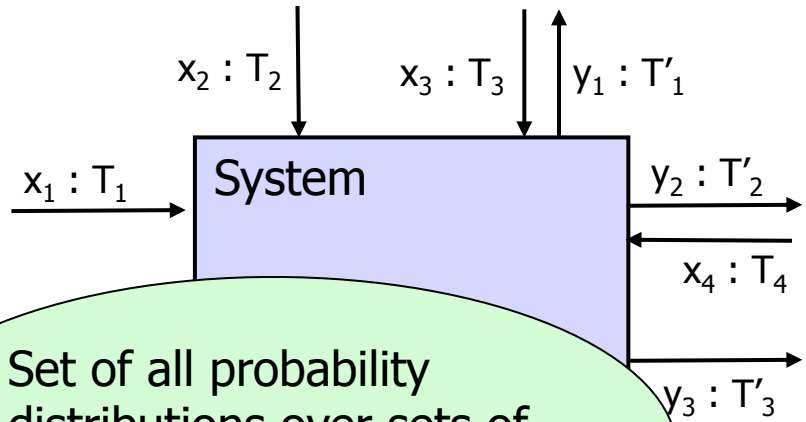
$STREAM[T] = \{IN\backslash\{0\} \rightarrow T^*\}$

valuation of channel set C

$IH[C] = \{C \rightarrow STREAM[T]\}$

interface behaviour for syn. interface $(I \blacktriangleright O)$

$[I \blacktriangleright O] = \{IH[I] \rightarrow PD[\wp(IH[O])]\}$



Set of all probability distributions over sets of output histories

interface specification represented as interface assertion S

that talk about the probability of the correctness of interface assertions

See: P. Neubeck: A Probabilitistic Theory of Interactive Systems. PH. D. Dissertation, Technische Universität München, Fakultät für Informatik, December 2012

Extensions of the model

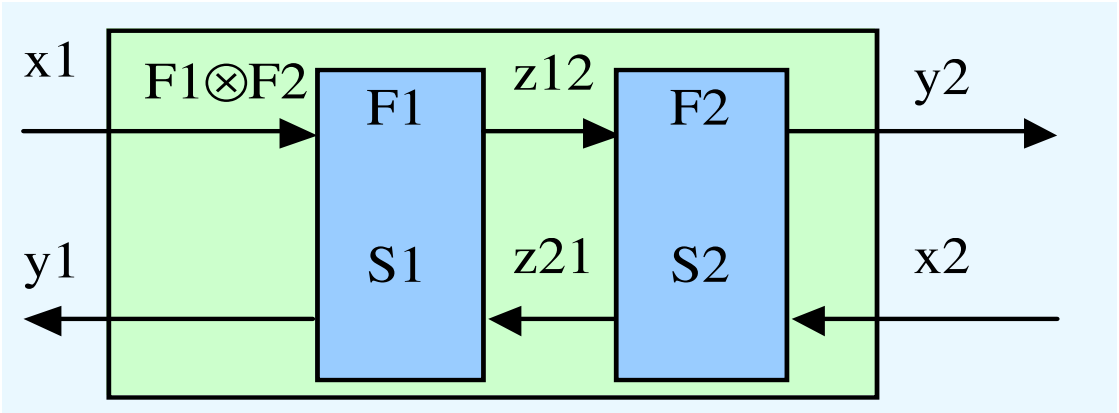
- Physical Aspects & Properties: Rich Models
  - Space
  - Geometry
  - Temperature
  - …

See: B. Hummel: Integrated Behavior Modeling of Space-Intensive Mechatronic Systems, Technische Universität München, Fakultät für Informatik, December 2010

Structure:
Composition and Decomposition

Technische Universität München
Institut für Informatik
D-80290 Munich, Germany

Modularity: Rules of compositions for interface specs



F1

in	x1, z21: T
out	y1, z12: T
S1	

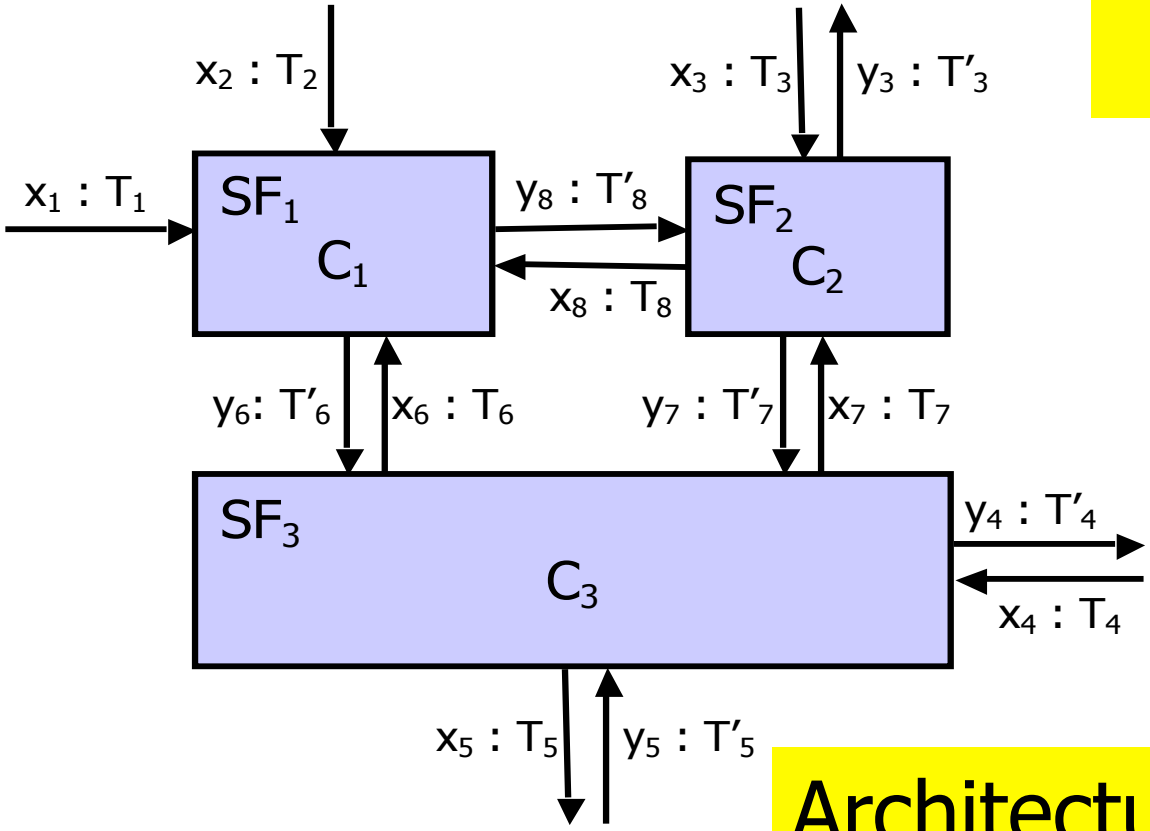
F2

in	x2, z12: T
out	y2, z21: T
S2	

F1⊗F2

in	x1, x2: T
out	y1, y2: T
∃ z12, z21: S1 ∧ S2	

Forming Architectures



Architecture Behaviour

$SF_1 \otimes SF_2 \otimes SF_3$

Architecture Spec

$C_1 \wedge C_2 \wedge C_3$

Architecture Correctness

$C_1 \wedge C_2 \wedge C_3 \Rightarrow SysSpec$

Probabilistic Behavior Composition

Probabilistic behavior

$$F: \wp(\mathbb{H}[I]) \rightarrow (\wp(\mathbb{H}[O]) \rightarrow [0:1])$$

We write for $X \subseteq \mathbb{H}[I]$, $Y \subseteq \mathbb{H}[O]$

$$F(X)[Y]$$

for the probability

that the output of F is in set Y provided the input is in set X

Probabilistic Behavior Composition

Given probabilistic behaviors

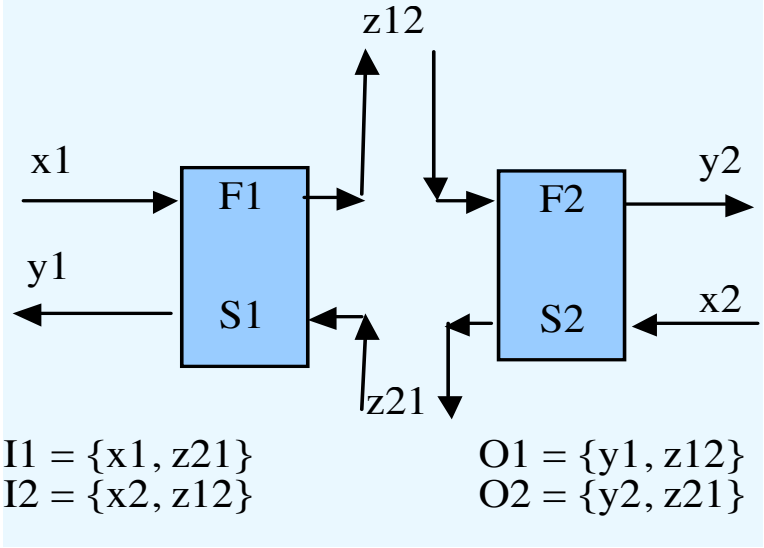
F1: $\wp(\mathbb{H}[I1]) \to (\wp(\mathbb{H}[O1]) \to [0:1])$

F2: $\wp(\mathbb{H}[I1]) \to (\wp(\mathbb{H}[O2]) \to [0:1])$

Where $O1 \cap O2 = \emptyset$; we define

$I = I1 \backslash O2 \cup I2 \backslash O1$, $O = (O1 \cup O2) \backslash Z$,

$Z = (I1 \cap O2) \cup (I2 \cap O1)$ shared channels

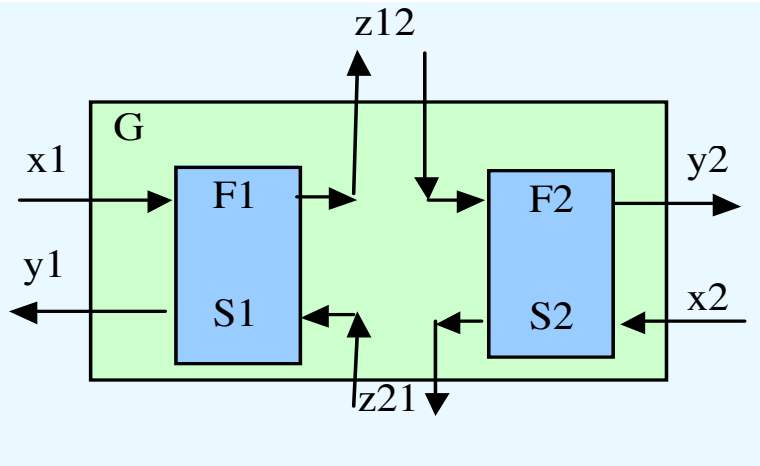


Independent composition: define

$$G: \wp(\mathbb{H}[I1 \cup I2]) \to (\wp(\mathbb{H}[O1 \cup O2]) \to [0:1])$$

by

$$G(X)[Y] = F1(X|_{I1})[\{y|_{O1}: y \in Y\}] \times F2(X|_{I2})[\{y|_{O2}: y \in Y\}]$$



Assuming probabilistic independence

Probabilistic Behavior Composition

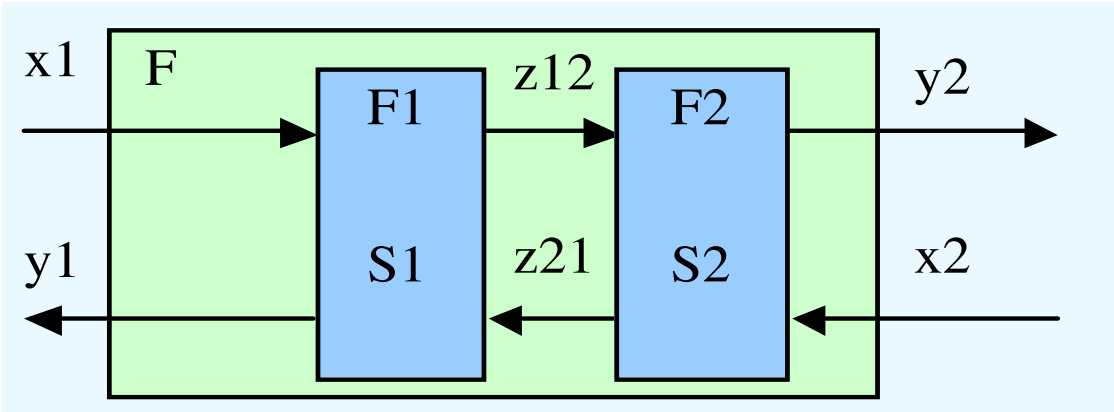
Given G we specify $F = F1 \otimes F2$

$$F: \wp(\mathbb{H}[I]) \rightarrow (\wp(\mathbb{H}[O]) \rightarrow [0:1])$$

by (here $y \in \mathbb{H}[O1 \cup O2]$)

$$F(X)[Y] \;=\; G(X')[\{y'|_O : \exists\, x' \in X' : y'|_O \in Y \wedge x'|_Z = y'|_Z\}]$$

where $X' = \{x : x|_I \in X\}$.

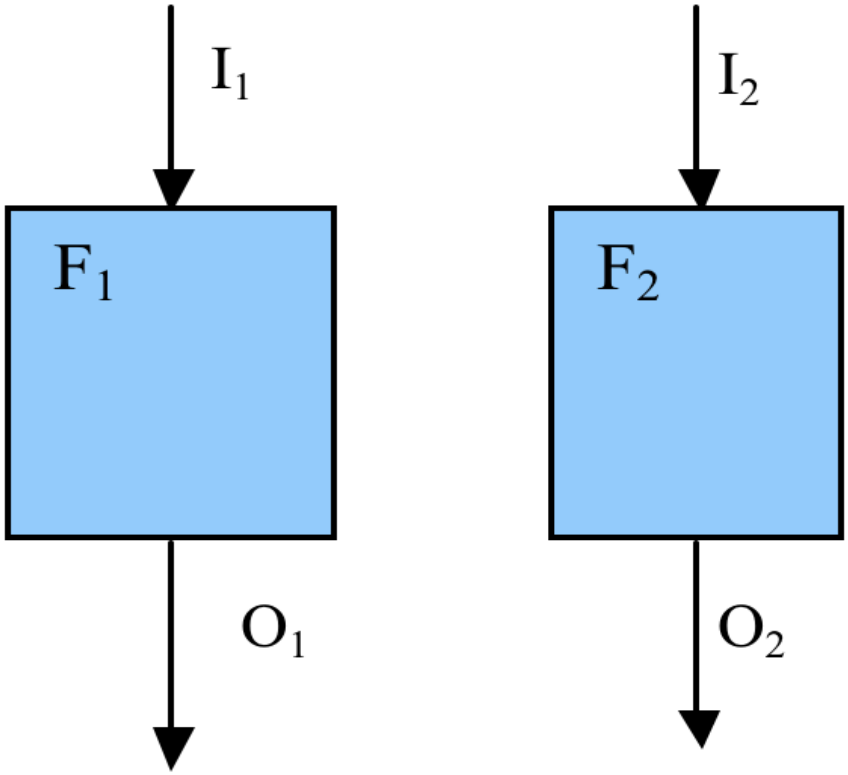


Functional View: Functional Decomposition

Technische Universität München
Institut für Informatik
D-80290 Munich, Germany

Combining Functions

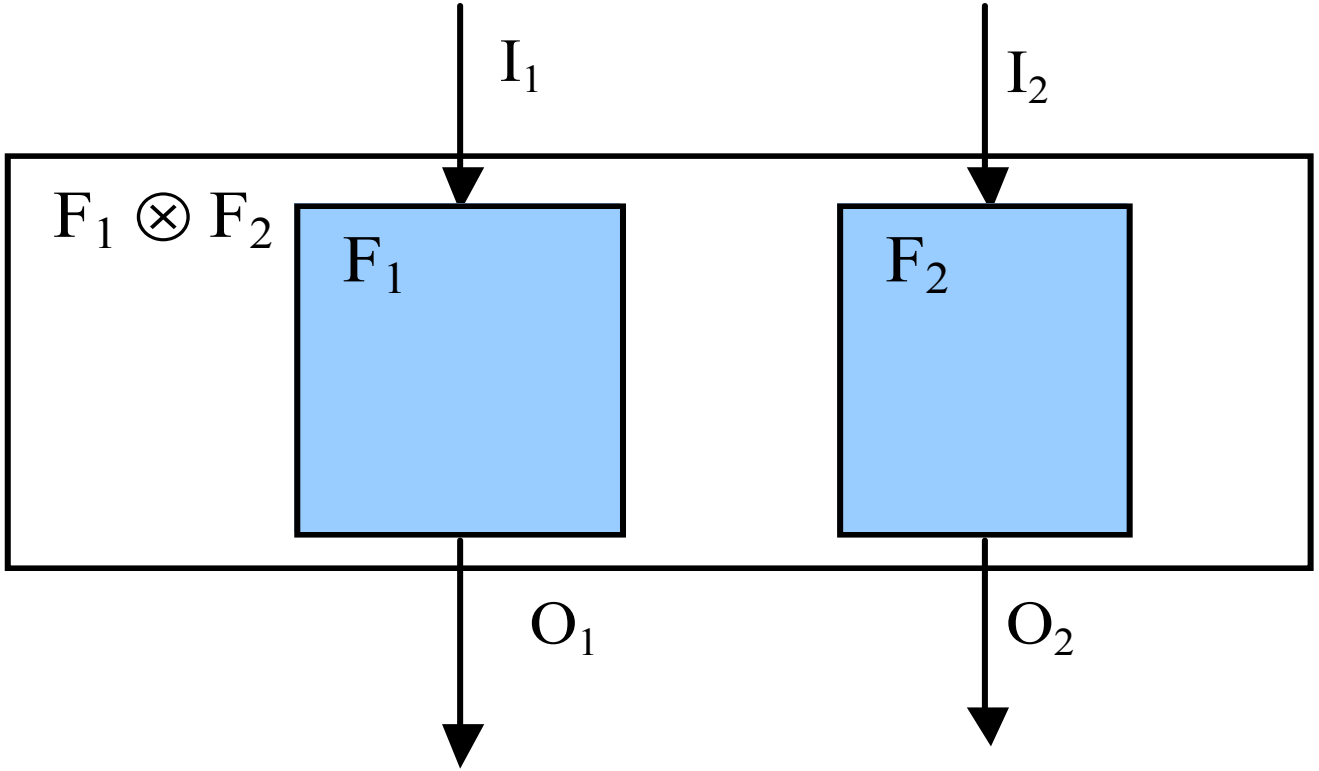
Given two functions F_1 and F_2 in isolation



We want to combine them into a function $F_1 \otimes F_2$

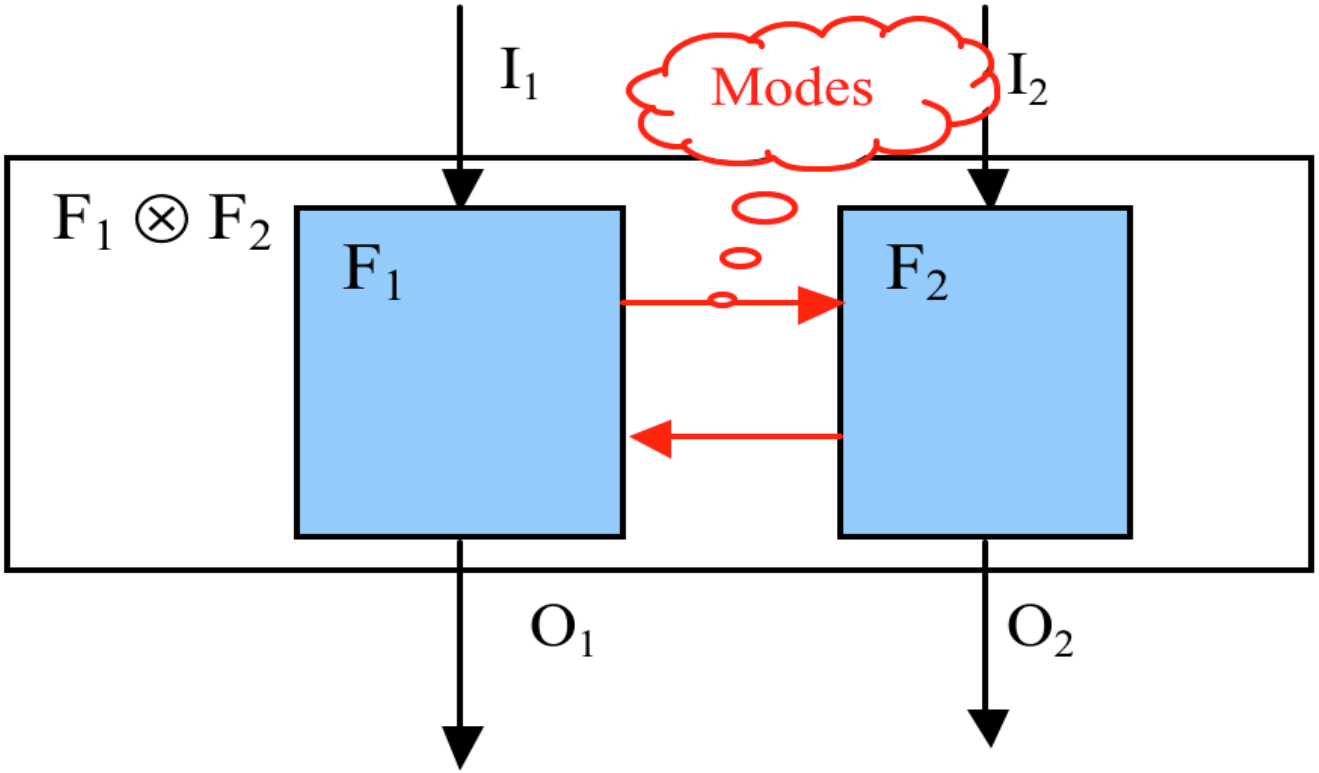
Combining Functions

Their isolated combination



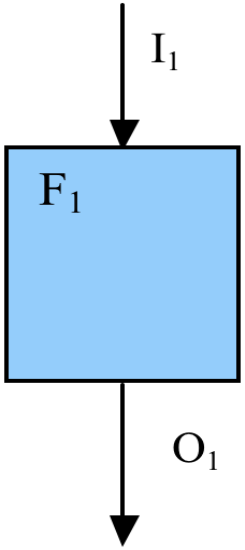
Combining Functions

If services F_1 and F_2 have feature interaction we get:

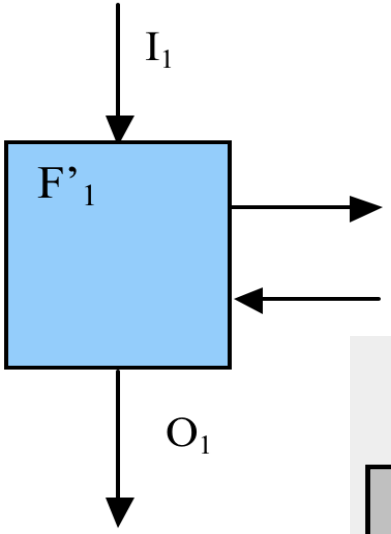


We explain the functional combination $F_1 \otimes F_2$ as a refinement step

The steps of function combination

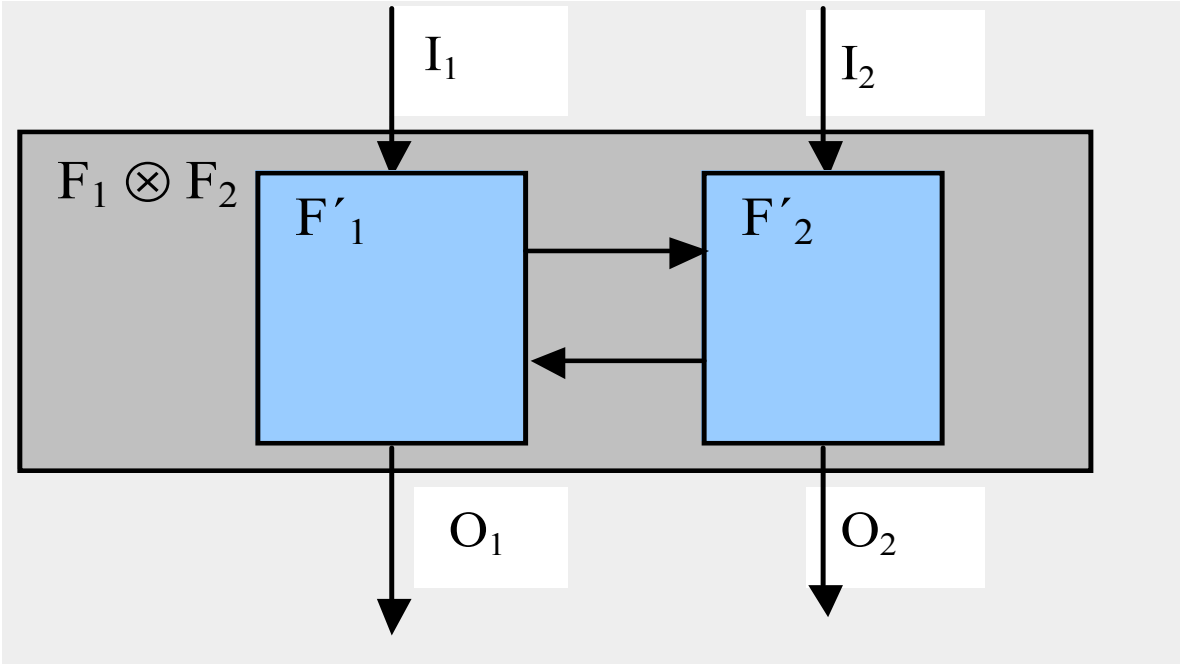


Given the isolated function F_1



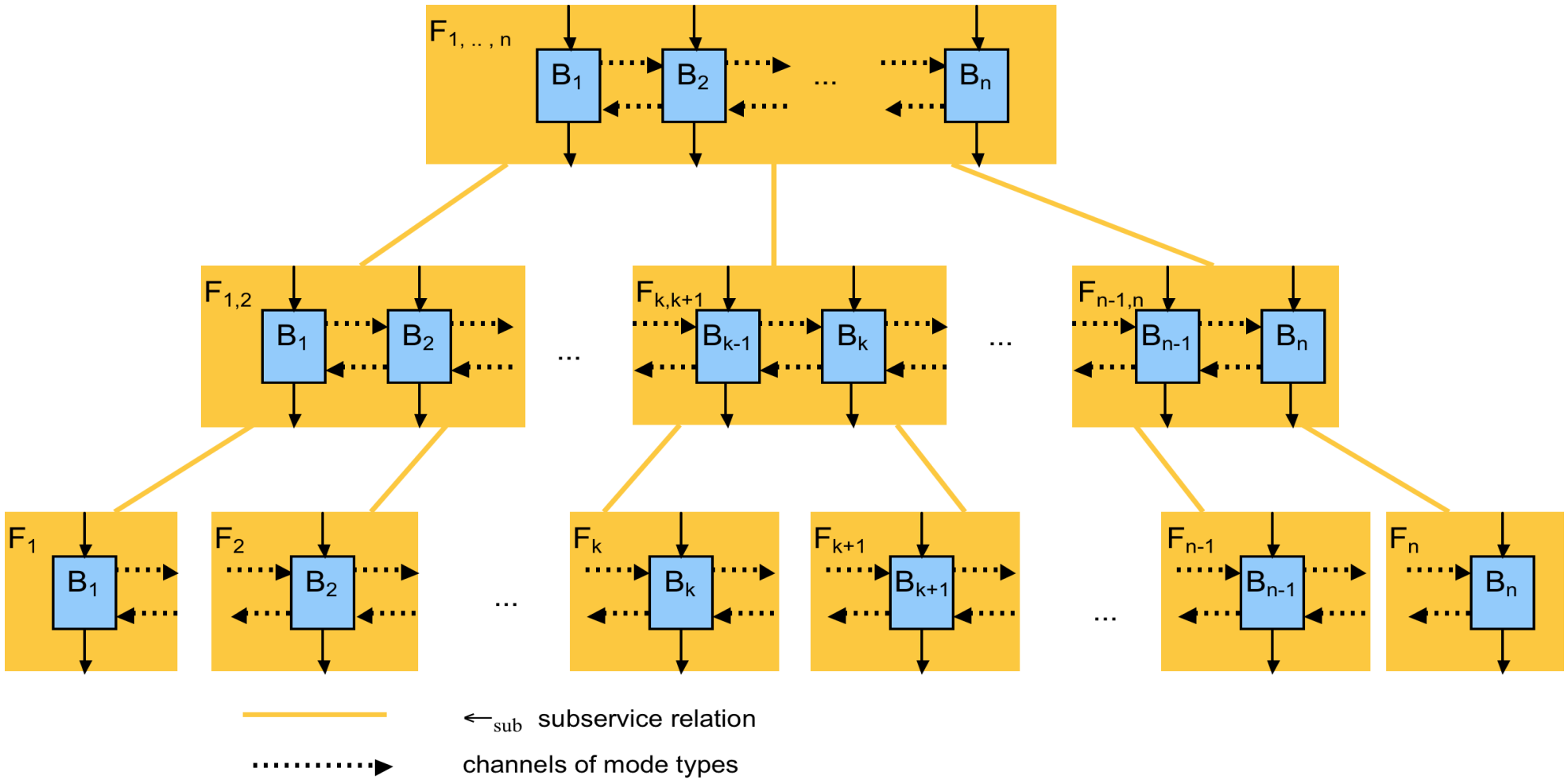
We construct a refinement F'_1

And combine F'_1 with a refinement F'_2 of F_2



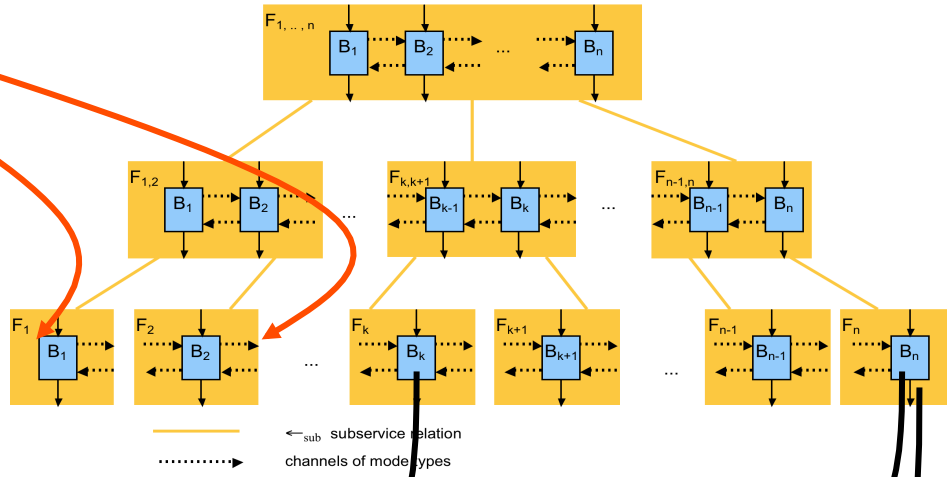
See: M. Broy: Multifunctional Software Systems: Structured Modeling and Specification of Functional Requirements. Science of Computer Programming 75 (2010), S. 1193–1214

Function Hierarchy

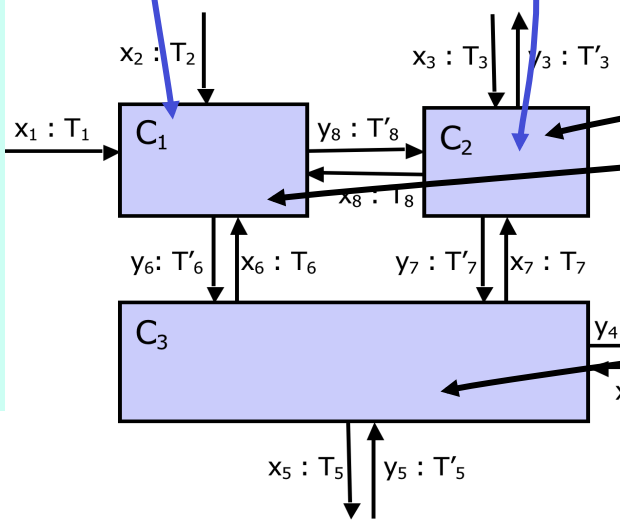


Relational view: Tracing – The power of logics

	Functiona	Safety	Priority	Component	Function
A_1	…	Yes	high		
A_2	…	No	medium		
A_n	…	no	low		

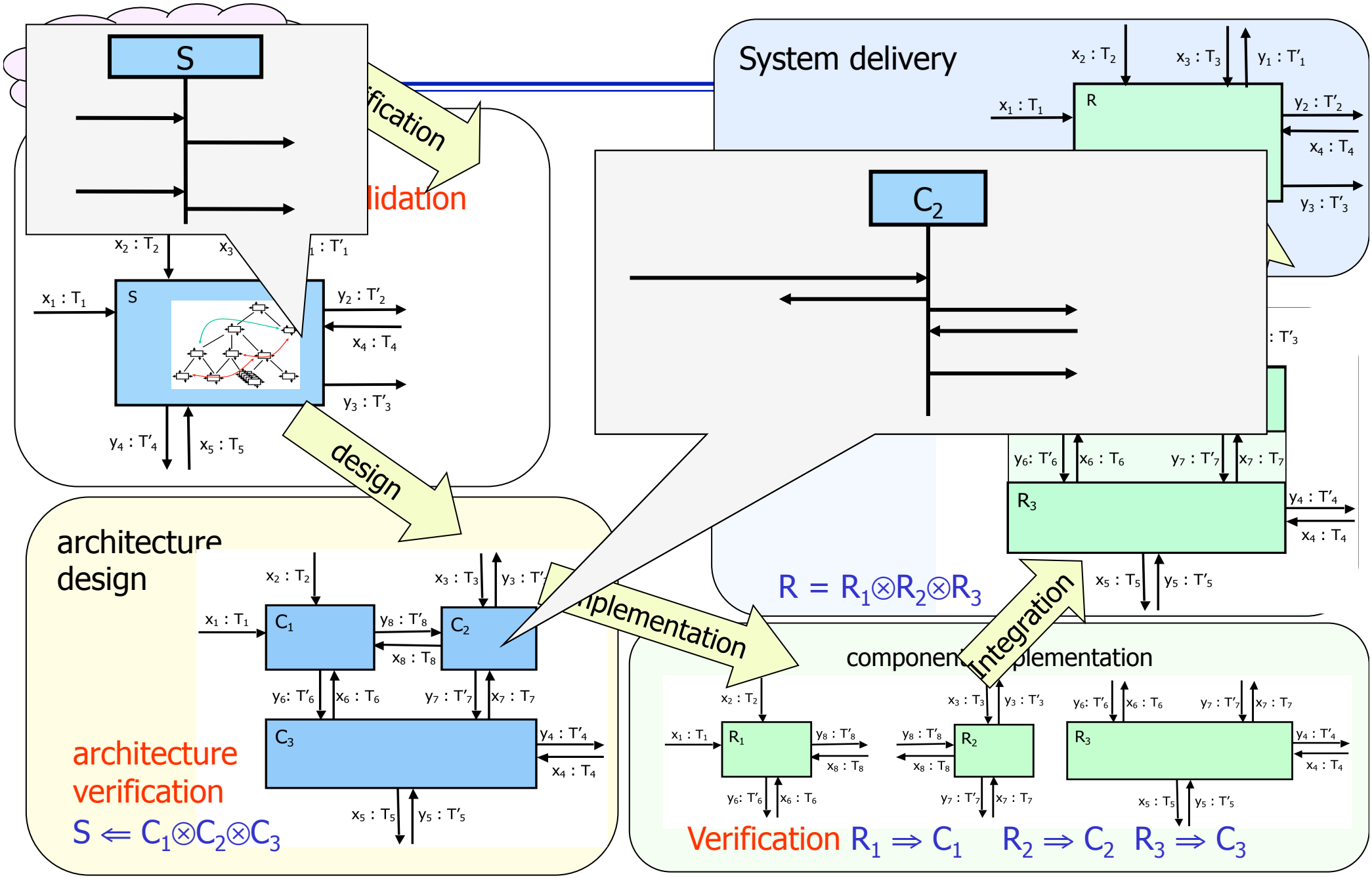


See: M. Broy: The Logic of Requirements – Formalizing Tracing, In: Forms/Format 2012, Technische Universität Braunschweig, edited by Eckehard Schnieder und Géza Tarnai, Beyrich Digital Servide GmbH & Co. KG, S. 2-4



Modular Model Based System Development

Technische Universität München
Institut für Informatik
D-80290 Munich, Germany


S
ification
lidation
design
System delivery
R
C2
R = R1⊗R2⊗R3
Integration
architecture
design
C1
C2
C3
nplementation
component
plementation
R1
R2
R3
architecture
verification
S ⇐ C1⊗C2⊗C3
Verification R1 ⇒ C1 R2 ⇒ C2 R3 ⇒ C3

Extended Notions of Correctness: Reliability

Technische Universität München
Institut für Informatik
D-80290 Munich, Germany

Correctness and reliability …

And what about requirements specification?

- Correctness does not make sense without specifications!
- Reliability needs also notions of correctness!

However for cyber-physical system specification and correctness is a bit more tricky …

- Time
- Probability
- Precision
- Uncertainty of the physical world
- …

The challenge: uncertainty and correctness of software/systems

- Classical: „sharp" correctness – black or white
  - a system/program is correct or not
- Unsharp correctness:
  - Correct to a certain degree
  - Correct with a certain probability
  - Correct over a certain time
  - Correct in some fuzzy way

Challenge

- specification
- verification

in der presence von unsharpness/uncertainty

Formalizations of unsharp correctness

Given: set $T \subseteq \mathbb{H}[I]$ of correct output histories

- Classical correctness:

  output $t' \in \mathbb{H}[I]$ is correct, iff $t' \in T$

- Extension: output t' more correct than output t''

  Define distance $d(t, t')$ between output streams:
  t' is more correct as t'' iff

$$\min \{ d(t, t'): t \in T \} < \min \{ d(t, t''): t \in T \}$$

- result $t' \in \mathbb{H}[I]$ is correct with a certain probability :

$$\mathbf{P}[t' \in T] > 0.9$$

$$\mathbf{P}[\min \{ d(t, t'): t \in T \} < 0.1] > 0.9$$

- Fuzzy: result $t' \in \mathbb{H}[I]$ is roughly correct – formalized in fuzzy logic

Dependability

Comprehensive view **dependability**:

- **Availability** - readiness for sufficiently correct service
- **Reliability** - continuity of correct service
- **Safety** - absence of catastrophic consequences on the user(s) and the environment
- **Security** - Integrity - absence of improper system alteration/ degree of resistance to or protection from vulnerability
- **Maintainability** - ability for a process to undergo modifications and repairs

Rich Interface Specifications

In a rich interface specification we speak about several views

Example: Add probability

Given:

- logical interface behavior for syn. interface $(I \blacktriangleright O)$

$$\{\mathbb{H}[I] \rightarrow \wp(\mathbb{H}[O])\}$$

- probabilistic interface behavior for syn. interface $(I \blacktriangleright O)$

$$\{\mathbb{H}[I] \rightarrow PD\,[\ \wp(\mathbb{H}[O])\ ]\ \}$$

- interface specification by an interface assertion $q(x, y)$
- specify for each input history $x = a$ probability distributions $\mathbf{P}(y|a)$ on the set of output histories

$$\{y: q(a, y)\}$$

Rich Specifications

Specifying system behavior and functionality:

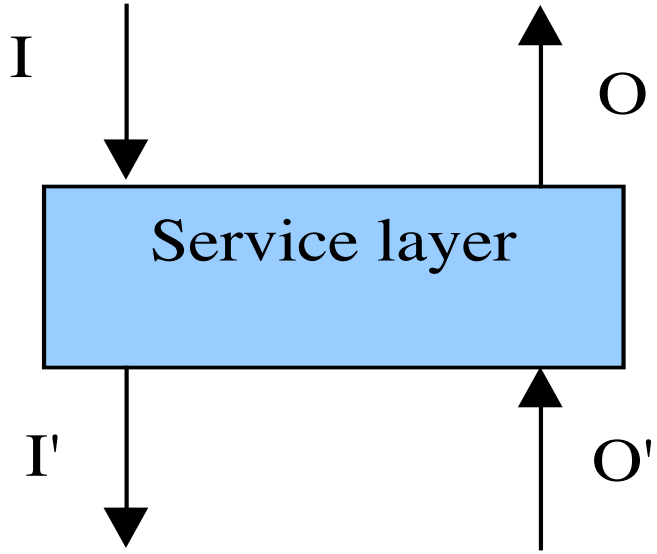
- Syntactic interface
- Logical system behavior
  - Data flow
  - Timing
  - Discrete and/or continuous
- Probabilistic system behavior
- Quality aspects
  - Quality of service: reliability – availability/correctness
  - Performance
  - Safety
  - Security
  - …
- Physical behavioral aspects

Service Layers and Service Stacks

Technische Universität München
Institut für Informatik
D-80290 Munich, Germany

Service Layers

A service layer is a service
with syntactic interface $(I \cup O' \blacktriangleright I' \cup O)$
structured into an
provided ("exported") service $(I \blacktriangleright O)$
requested ("imported") service $(I' \blacktriangleright O')$.



We assume $I \cap O' = \emptyset$ and $O \cap I' = \emptyset$.

A service layer is a service with the interface behavior

$$L: \mathbb{H}(I \cup O') \rightarrow \wp(\mathbb{H}(I' \cup O))$$

where both input and output actions are disjoint sets.

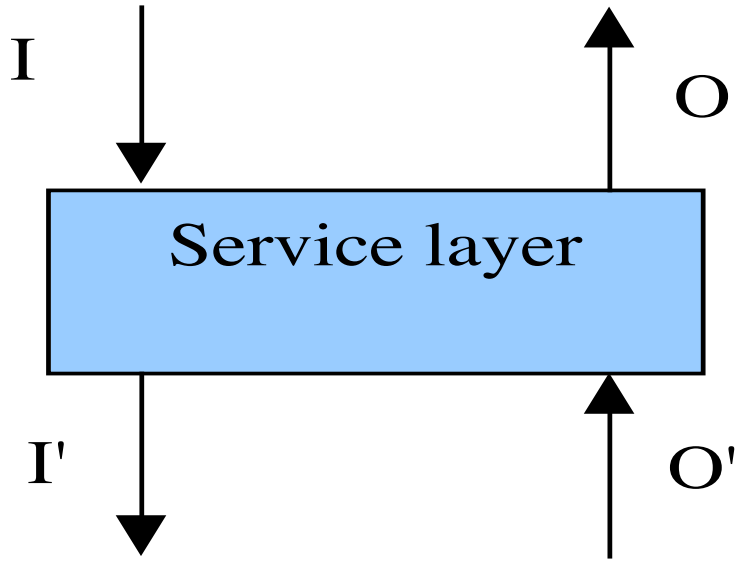
Service Layers

We denote the syntactic interface of a service layer by

$(I \blacktriangleright O/O' \blacktriangleright I')$ syntactic service layer interface

The service layer is a service

$L \in [I \cup O' \blacktriangleright O \cup I']$

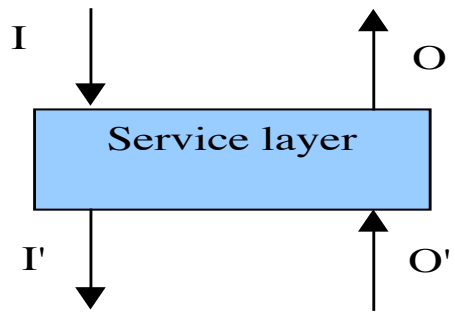


Specifying Service Layers

To specify a service layer we specify two services

The provided service $F \in [I \blacktriangleright O]$

The required service $R \in [I' \blacktriangleright O']$



A layer provides the service F under the condition that it gets the service R from "below".

Note that

- R does not specify the service of the layer as provided by L but the required service.

Given F and R we denote the layer L that provides service F provided service R is offered as an auxiliary service by

$$F//R$$

Specifying Service Layers

Layer

$$L = F//R$$

is specified as follows (for $x \in \mathbb{H}(I \cup O')$)

$$L(x) = \{y \in \mathbb{H}(I' \cup O): x|O' \in R(y|I') \Rightarrow y|O \in F(x|I)\}$$

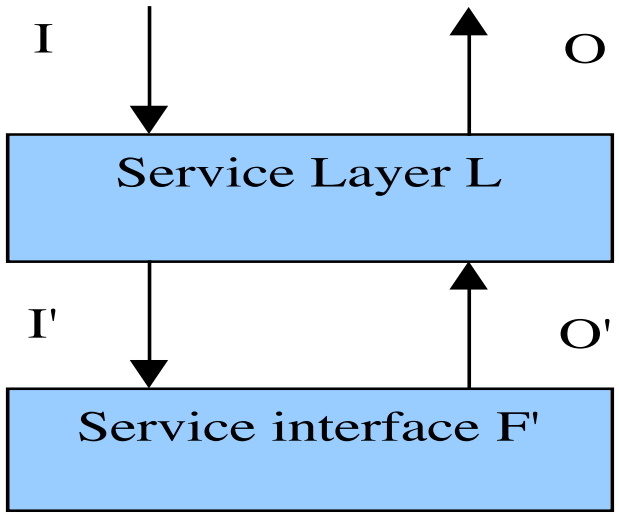
This expresses that

- if the service provided from "below" is correct as required and specified by R than the offered service is as specified by F.
- Note that this is written in the pattern of an assumption/ commitment specification.

Composing Layers with Services

Given a service R' as requested from below

$R' \in [I' \blacktriangleright O']$ imported service



and a service layer

$L \in [I \blacktriangleright O / O' \blacktriangleright I']$

where $L = F//R$

with given $F \in [I \blacktriangleright O]$ and $R \in [I' \blacktriangleright O']$

We get the composition of layer L with service R'

$L \otimes R' \in [I \blacktriangleright O]$ composition of layer L with service R'

Service Refinement

Given services

$$F, F' \in [I \blacktriangleright O]$$

F' is called a refinement of F iff

$$\forall\, x \in \mathbb{I}H(I): F'(x) \subseteq F(x)$$

then we write

$$F \rightarrowtail F'$$

Composing Layers with Services

$L \otimes R' \in [I \blacktriangleright O]$ is a refinement of the provided service $F \in [I \blacktriangleright O]$

$$F \rightarrowtail L \otimes R'$$

provided service R' is a refinement of the requested service R

$$R \rightarrowtail R'$$

Given layer $L = F//R$ we get the following proof rule for layered architectures

$$L = F//R \wedge R \rightarrowtail R'$$
$$\Rightarrow$$
$$F \rightarrowtail L \otimes R'$$

Composing Layers

Given two layers

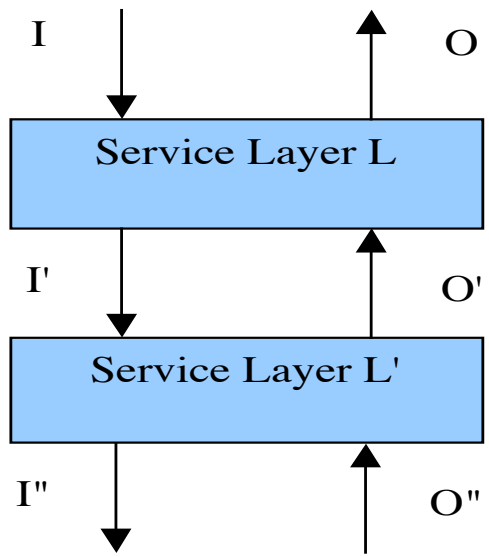
$$L \in [I \blacktriangleright O/O' \blacktriangleright I'],\ L' \in [I' \blacktriangleright O'/O'' \blacktriangleright I'']$$

where we assume that $I \cap I'' = \emptyset$ and $O \cap O'' = \emptyset$;

we define the layer composition

$$L \otimes L' \in [I \blacktriangleright O/O'' \blacktriangleright I'']$$

yielding a layer in $[I \blacktriangleright O/O'' \blacktriangleright I'']$.



Assume service L and L′ are described as follows

$$L = F//R$$

$$L' = F'//R'$$

We call these two layers fitting if

$$R \rightarrowtail F'$$

then we conclude $F//R' \rightarrowtail L \otimes L'$

proof rule for layered architectures

$$L = F//R \wedge L' = F'//R' \wedge R \rightarrowtail F'$$

$$\Rightarrow$$

$$F//R' \rightarrowtail L \otimes L'$$

Layered Architecture forming a Service Stack

The service stack is specified by a set of services.

$$F_j \in [I_j \blacktriangleright O_j]$$

the family of export services for $0 \leq j \leq n$.

The layers

$$L_{j+1} \in [I_{j+1} \blacktriangleright O_{j+1} / O_j \blacktriangleright I_j]$$ family of layers

form the stack. Given requested services

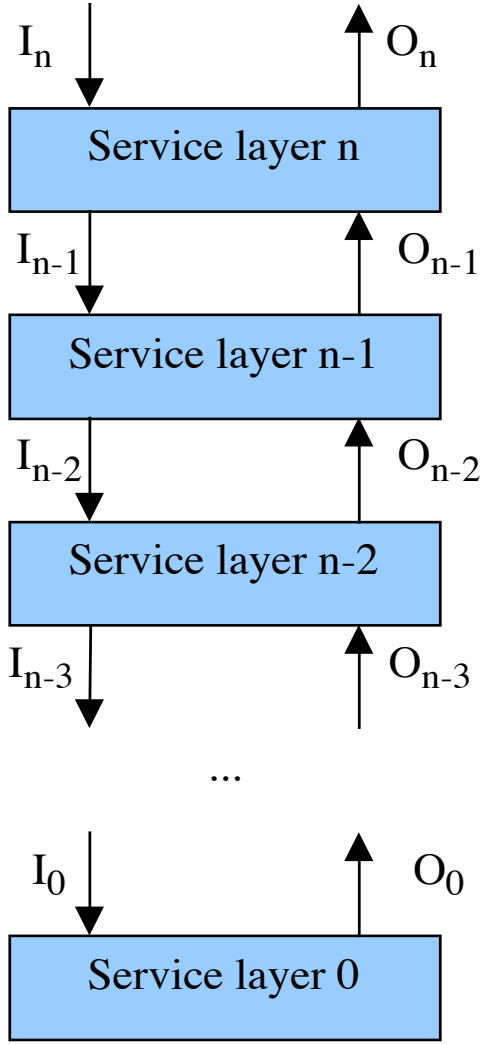
$$R_{j+1} \in [I_j \blacktriangleright O_j]$$

and layer specifications

$$L_j = F_j \,//\, R_j$$

Proof condition for the stack:

$$R_{j+1} \rightarrowtail F_j$$



Service layers that fit together can be composed to service stacks.

Rich Service and Service Layer Specification

Adding probabilistic behavior models, we can extend this approach to services writing

- Rich service specifications
  - Quality of service specification: reliability, availability, service quality
- Rich specifications of service layers
- Rich specifications of service stacks
- Rich specifications of service plaforms
- Service composition

The Triad of Modeling

- Denotational:
  - ◊ mathematical system models (modeling behavior)
  - ◊ relating models: composition, refinement, abstraction

Mathematical representations of models for understanding and getting the modeling theory right

- Logical – system properties:
  - ◊ specification
  - ◊ deduction
  - ◊ verification
  - ◊ transformation

Representations of all properties of systems (requirements, interface, functionality, architecture) in logic for specification and verification

- Notational:
  - ◊ Graphical: diagrams
  - ◊ Tables
  - ◊ Formulas

Representations of models of systems (requirements, interface, functionality, architecture) in structured easier to grasp forms

Concluding Remarks

- The modelling framework FOCUS
  - originally worked out for model based development
  - specification
  - verification
  - tool support
- Tool: Autofocus 3
- Also useful for semantic foundation following the same approach
  - SDL
  - Bus Systems: CAN, FLEXRAY
  - UML/SysML
  - SOA

See: https://af3.fortiss.org/projects/autofocus3

See: K. Pohl, H. Hönninger, R. Achatz, M. Broy: The SPES 2020 Methodology. Springer Verlag 2012

Concluding Remarks

- Today software & systems engineering is too much orientated towards the technical architecture and solutions/ implementation in the beginning
- We need a comprehensive "architectural" model-based view onto systems including requirements for dealing with complex multi-functional systems
- The models allow for
  - Separation of concerns
  - Separation of technical aspects from application aspects
- Technical architectures are modelled along the same theory
- Code and test cases can be generated from the models

The power of generalizing ideas, of drawing comprehensive conclusions from individual observations, is the only acquirement, for an immortal being, that really deserves the name of knowledge.

"Mary Wollstonecraft (1759–1797), British feminist. A Vindication of the Rights of Woman, ch. 4 (1792)