Mandatory Properties of Component Concepts

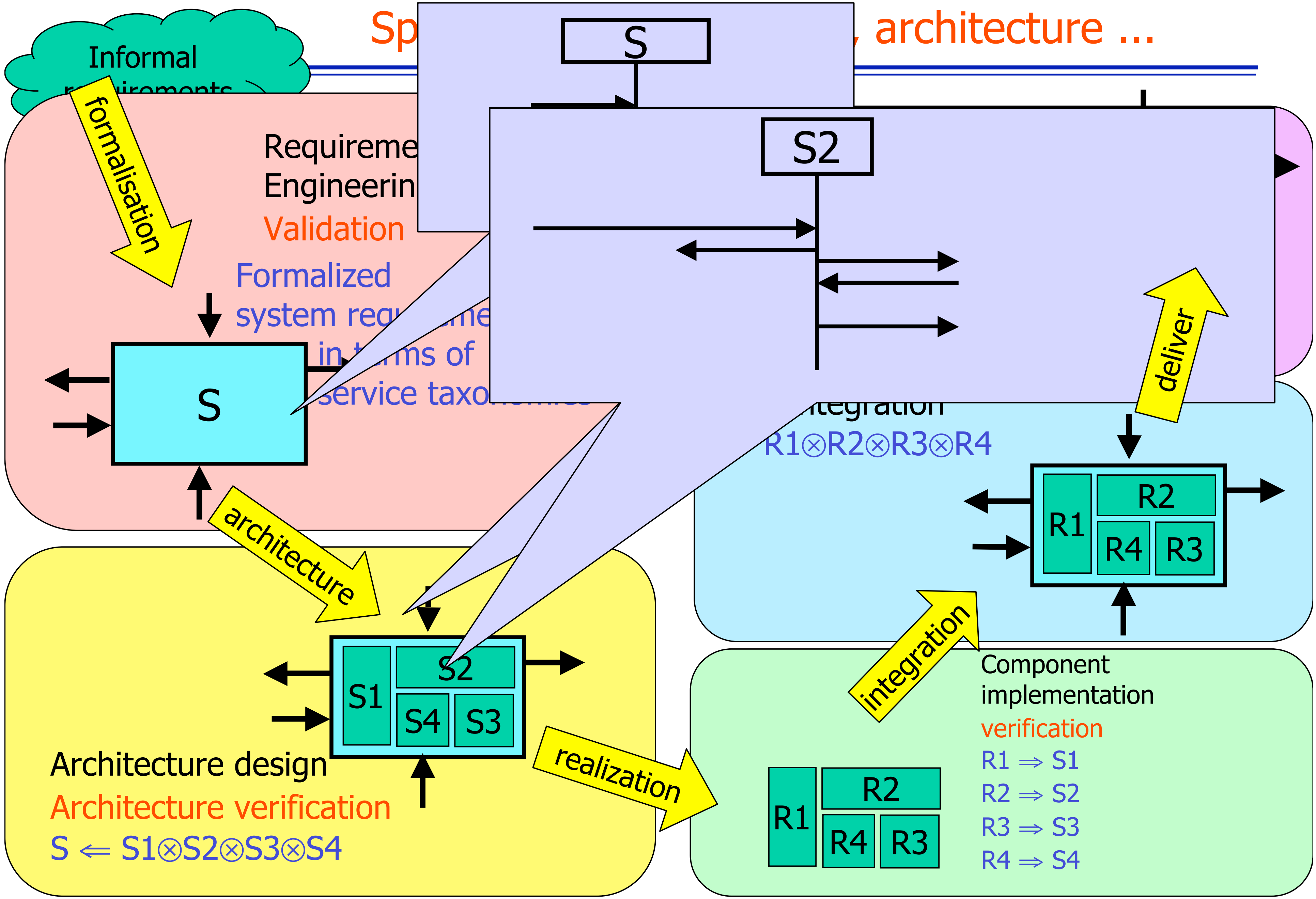
Manfred Broy

Technische Universität München
Institut für Informatik
D-85748 Garching, Germany

Content

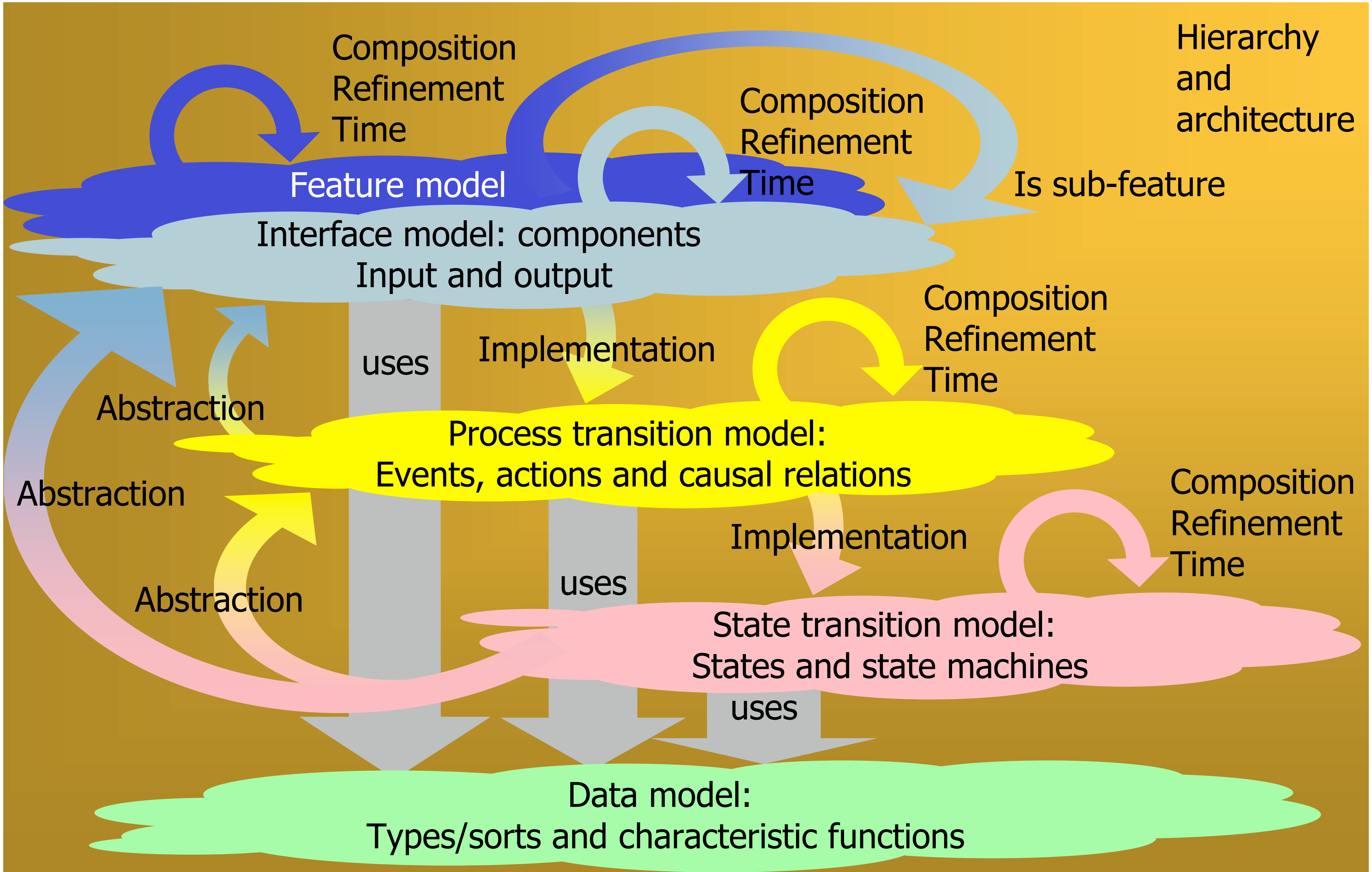
- What is a system component concept?

  A concept of a self-contained, independent unit carrying functionality that can be analysed, refined and composed to form larger systems:
  - deployable
  - executable
  - adaptable
- Mandatory ingredients: concept of component
  - specification - implementation independent
  - composition/decomposition - architecture
  - refinement - properties and levels of abstraction
  - implementation - executable system descriptions
  - abstraction
    - relating implementations to specifications
    - relating components at different levels of abstraction



Sp
, architecture ...
Informal
formalisation
Requireme
Engineerin
Validation
Formalized
system req
in t rms of
service taxo
S
S
S2
deliver
R1⊗R2⊗R3⊗R4
R1
R2
R4
R3
architecture
S1
S2
S4
S3
Architecture design
Architecture verification
S ⇐ S1⊗S2⊗S3⊗S4
realization
integration
Component
implementation
verification
R1 ⇒ S1
R2 ⇒ S2
R3 ⇒ S3
R4 ⇒ S4
R1
R2
R4
R3

Towards a comprehensive theory of system modelling: meta model



These notions ...

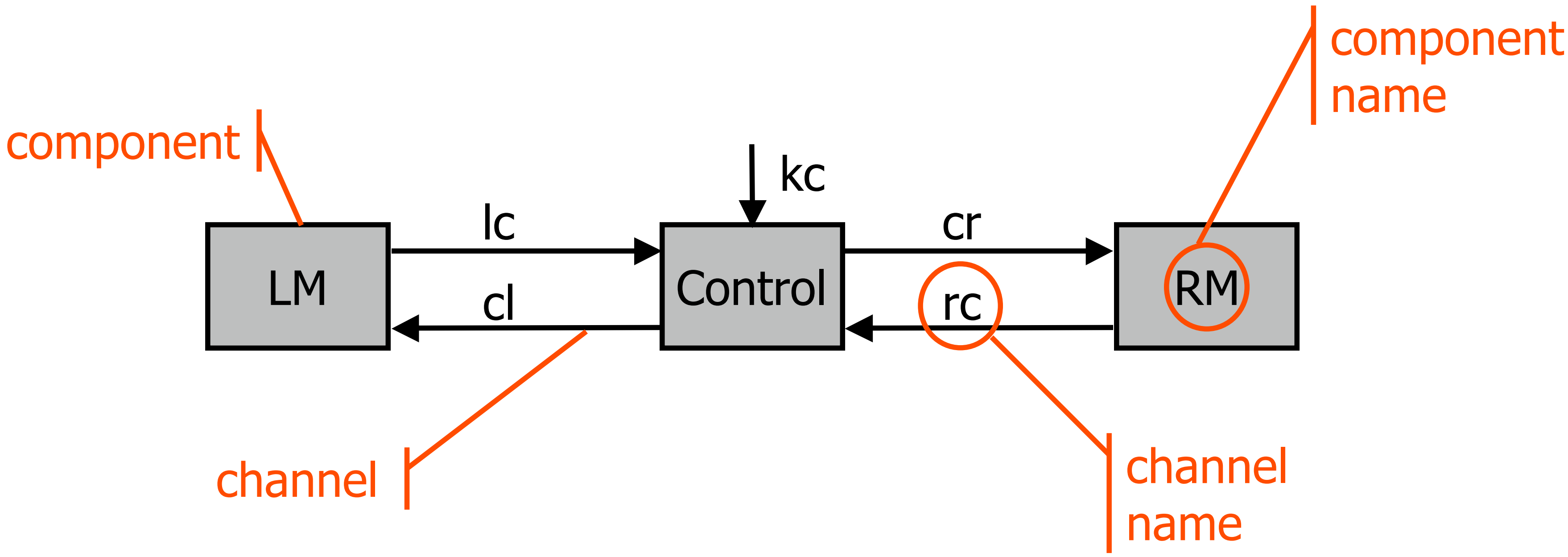
... form a taxonomy

But this is not enough!

We need a semantic modelling theory!

Towards a uniform model: Basic system model

System class: distributed, reactive systems



System consists of

- named components (with local state)
- named channels

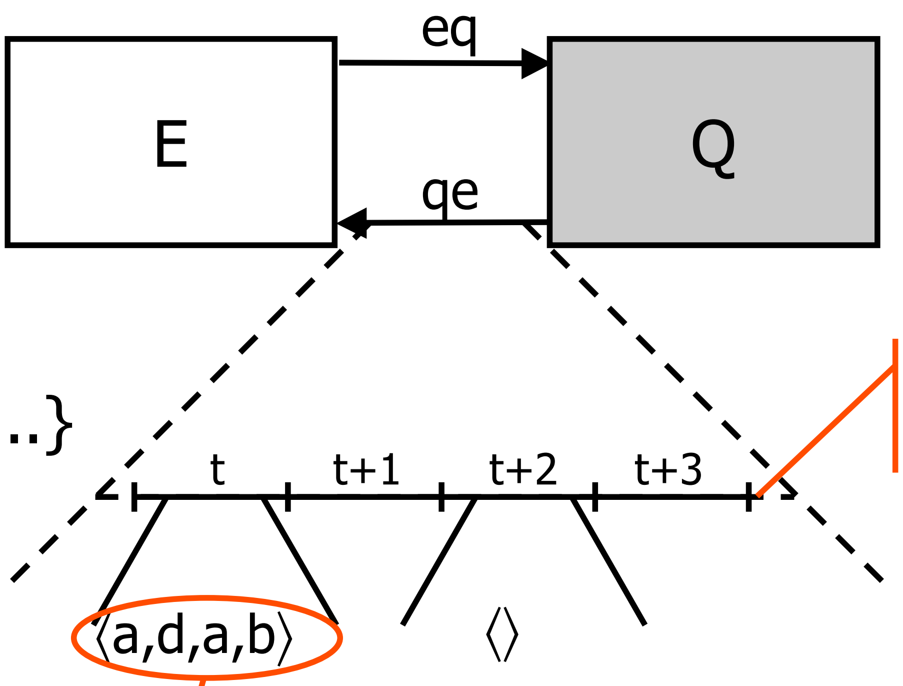
driven by a global, discrete clock

Basic system model

Timed Streams: Semantic Model for Black-Box-Behavior

Message set:

M = {a, b, c, ...}



infinite channel history

Messages
transmitted at time t

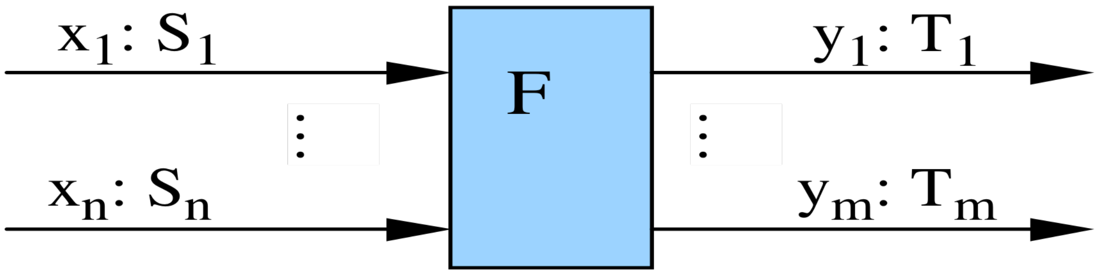
The Basic Behaviour Model: Timed Streams and Channels

C	set of channels
Type: C → TYPE	type assignment
$x : C \to (\mathbb{N}\backslash\{0\} \to M^*)$	channel history for messages of type M
$\vec{C}$ or IH[C]	set of channel histories for channels in C

System interface model

Channel: Identifier of Type stream

$I = \{ x_1 , x_2 , \ldots \}$ set of typed input channels
$O = \{ y_1 , y_2 , \ldots \}$ set of typed output channels



Interface behavior

$$F : \vec{I} \to \wp(\vec{O})$$

Set of interface behaviours with input channels I and output channels O:

IF[I ▶ O]

A component is a system

Set of all interface behaviours: IF

System interface behaviour - causality

(I ▶ O)

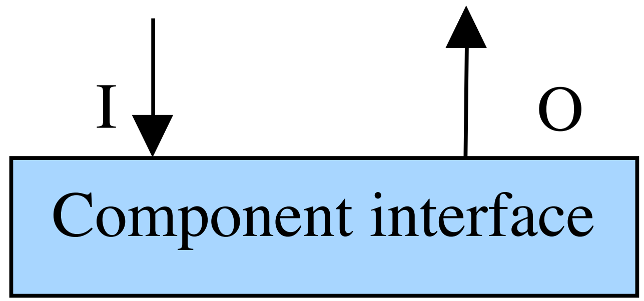
A system has a proper time flow

$F : \vec{I} \to \wp(\vec{O})$ *semantic interface* for (I ▶ O)
with *timing property addressing strong causality*
(let $x, z \in \vec{I}$, $y \in \vec{O}$, $t \in \mathbb{N}$):

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

$x \downarrow t$ prefix of history x of length t

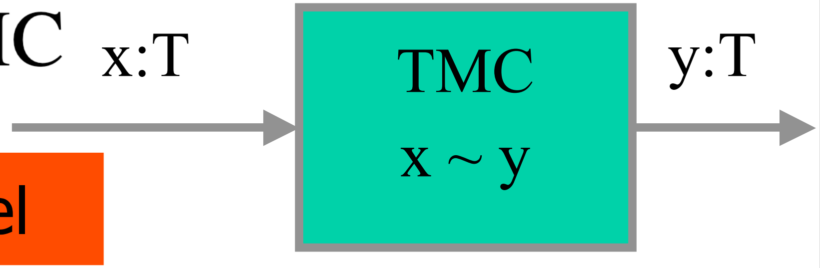
A system shows a total behavior



Example: Component interface specification

Spec name

A transmission component TMC



Input channel

TMC

in	x: T
out	y: T
$x \sim y$	

A system has a logical specification

Output channel

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

Specifying assertion

$\{m\}\#x$ denotes the number of copies of m in stream x

Verification: Proving properties about specified compondents

From the interface assertions we can prove

- Safety properties

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

- Liveness properties

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

A system specification
- can be structured by logical properties and
- can be used to prove properties

State Machines

A state machine (Δ, Λ) consists of

- a set Σ of states - the state space
- a set $\Lambda \subseteq \Sigma$ of initial states
- a state transition function Δ
  - in case of a state machine with input/output:

    events (inputs E) trigger the transitions and events (outputs A) are produced by them respectively:

    $$\Delta : \Sigma \times \mathrm{E} \rightarrow \Sigma \times \mathrm{A}$$

    in the case of nondeterministic machines:

    $$\Delta : \Sigma \times \mathrm{E} \rightarrow \wp(\Sigma \times \mathrm{A})$$

- Given a syntactic interface with sets I and O of input and output channels:

$$\mathrm{E} = \mathrm{I} \rightarrow \mathrm{M}^*$$

$$\mathrm{A} = \mathrm{O} \rightarrow \mathrm{M}^*$$

A system has an implementation

Computations of a State Machine with Input/Output

A state machine (Δ, Λ) defines for each initial state

$$\sigma_0 \in \Lambda$$

and each sequence of inputs

$$e_1, e_2, e_3, \ldots \in E$$

a sequence of states

$$\sigma_1, \sigma_2, \sigma_3, \ldots \in \Sigma$$

and a sequence of outputs

$$a_1, a_2, a_3, \ldots \in A$$

through

$$(\sigma_{i+1}, a_{i+1}) \in \Delta(\sigma_i, e_{i+1})$$

Implementations define computations

Computations of a State Machine with Input/Output

In this manner we obtain com

A system has an interface abstraction
- that correctly reflects computations

$$\sigma_0 \xrightarrow{a_1 / b_1} \sigma_1 \xrightarrow{a_2 / b_2} \sigma_2 \xrightarrow{a_3 / b_3} \sigma_3 \quad \ldots$$

For each initial state $\sigma 0 \in \Sigma$ we define a function

$$F_{\sigma 0} : \vec{I} \to \wp(\vec{O})$$

with

$$F_{\sigma 0}(x) = \{y: \exists\, \sigma_i: \sigma 0 = \sigma_0 \wedge \forall\, i \in IN: (\sigma_{i+1}, x_{i+1}) = \Delta(\sigma_i, y_{i+1})\}$$

$F_{\sigma 0}$ denotes the interface behavior of the transition function Δ for the initial state $\sigma 0$.

Furthermore we define

$$Abs((\Delta, \Lambda)) = F_\Lambda$$

where:

$$F_\Lambda(x) = \{y \in F_\sigma(x) : y \in F_\sigma(x) \wedge \sigma \in \Lambda\}$$

F_Λ is called the interface behavior of the state machine (Δ, Λ)

Moore Machines

- A Mealy machine (Δ, Λ) with

  $$\Delta : \Sigma \times E \to \wp(\Sigma \times A)$$

  is called a Moore machine if for all states $\sigma \in \Sigma$ and all inputs $e \in E$ the set

  $$out(\sigma, e) = \{a \in A: (\sigma, a) = \Delta(\sigma, e) \}$$

  does not depend on the input e but only on the state σ.
- Formally: then for all $e, e' \in E$ we have

  $$out(\sigma, e) = out(\sigma, e')$$

Theorem: If is (Δ, Λ) a Moore machine the F_Λ is strongly causal.

An interface abstraction of an implementation
- has the required properties
- leads to correct assertions

Sub-Services, Functional Features

Technische Universität München
Institut für Informatik
D-85748 Garching, Germany

Syntactic sub-interfaces and projection

A typed channel set C' is called a *sub-type* of a typed channel set C if

- C' is a subset of C
- The message types of the channels in C' are subsets of the message sets of these channels in C

We write then

$$C' \textbf{ subtype } C$$

Then we denote for the channel history $x \in IH[C]$ by

$$x|C' \in IH[C']$$

the restriction of x to the channels and messages in C'

Sub-types between interfaces

For syntactic interfaces $(I \blacktriangleright O)$ and $(I' \blacktriangleright O')$ where

$$I' \textbf{ subtype } I \text{ and } O' \textbf{ subtype } O$$

we call $(I' \blacktriangleright$

An interface behaviour can be structured into

- independent sub-services
- a system offers services

For a behavior $F \in IF[I \blacktriangleright O]$ we define its *projection*

$$F \dagger (I' \blacktriangleright O') \in IF[I' \blacktriangleright O']$$

to the syntactic interface $(I' \blacktriangleright O')$ by (for all $x \in IH[I']$):

$$F \dagger (I' \blacktriangleright O')(x') = \{y|O' : \exists\, x \in IH[I]: x' = x|I' \wedge y \in F(x)\}$$

The projection is called *faithful*, if for all $x \in dom(F)$

$$F(x)|O' = (F \dagger (I' \blacktriangleright O'))(x|I')$$

Composing Systems

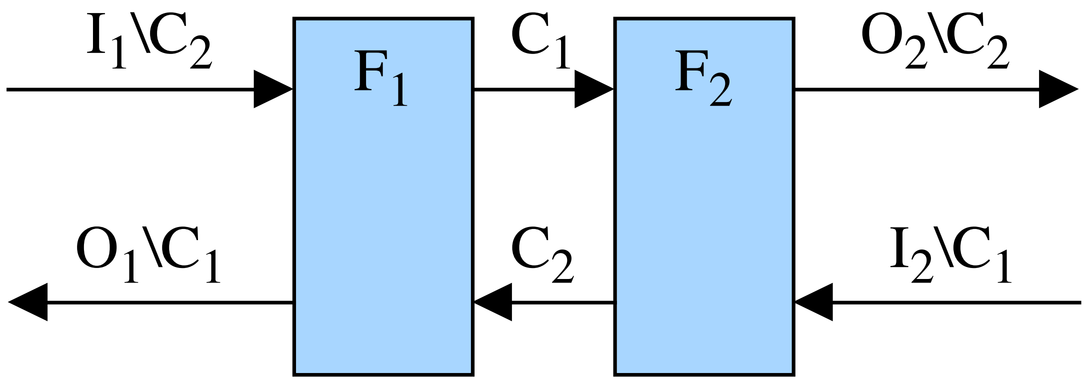
Technische Universität München
Institut für Informatik
D-85748 Garching, Germany

Composition

- of system models is compositional
- is hierarchical: a system is a component is a system

$F_1 \in IF[I_1 \blacktriangleright O_1]$

$F_2 \in IF[I_2 \blacktriangleright O_2]$



$C_1 = O_1 \cap I_2$

$C_2 = O_2 \cap I_1$

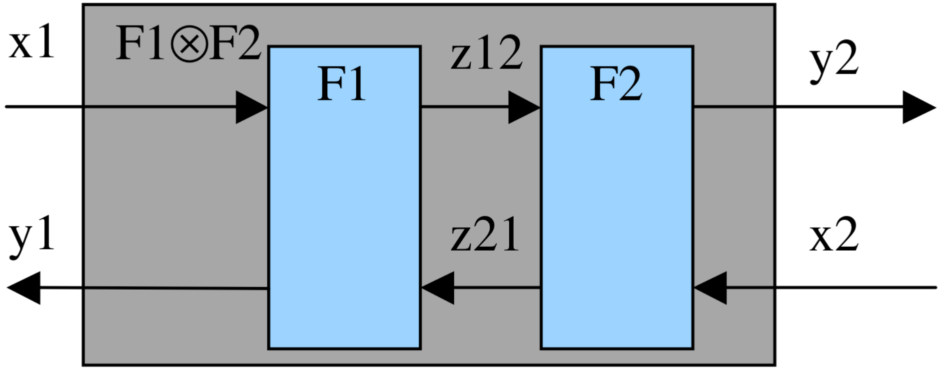
$I = I_1\backslash C_2 \cup I_2\backslash C_1$

$O = O_1\backslash C_1 \cup O_2\backslash C_2$

$F_1 \otimes F_2 \in IF[I \blacktriangleright O],$

$(F_1 \otimes F_2).x = \{z|O: x = z|I \wedge z|O_1 \in F_1(z|I_1) \wedge z|O_2 \in F_2(z|I_2)\}$

Interface specification composition rule



F1

in x1, z21: T

out y1, z12: T

P1

Interface specification is modular

out y2, z21: T

P2

F1⊗F2

in x1, x2: T

out y1, y2: T

$\exists$ z12, z21: P1 $\wedge$ P2

Composition of the two state machines

Consider Moore machines $M_k = (\Delta_k, \Lambda_k)$ (k = 1, 2):

$\Delta_k: \Sigma_k \times (I_k \to M^*) \to (\Sigma_k \times (O_k \to M^*))$

We define the composed state machine

$\Delta: \Sigma \times (I \to M^*) \to (\Sigma \times (O \to M^*))$

as follows

$\Sigma = \Sigma_1 \times \Sigma_2$

for $x \in I$ and

Composition for the implementation concept is available

$\Delta((s_1, s_2), x) = \{((s_1', s_2'), z|O): x = z|I \wedge \forall k: (s_k', z|O_k) = \Delta_k(s_k, z|I_k)\}$

This definition is based on the fact that we consider Moore machines.

We write

$\Delta = \Delta_1 \parallel \Delta_2$

$M = M_1 \parallel M_2 = (\Delta_1 \parallel \Delta_2, \Lambda_1 \times \Lambda_2)$

An example of an essential property ...

$$\mathrm{Abs}((\Delta 1, \sigma 1) \,||\, (\Delta 2, \sigma 2)\,) =$$

$$\mathrm{Abs}((\Delta 1, \sigma 1)) \otimes \mathrm{Abs}((\Delta 2, \sigma 2))$$

Interface abstraction distributes for state machines over composition

Component Architectures

Technische Universität München
Institut für Informatik
D-85748 Garching, Germany

Composition of Specifications into Architectures

Input channels

Output channels

Internal channels

Composed component spec

in $x_1 : M_1, x_2 : M_2, \ldots$

out y_1:

$\exists\, c_1, c_2,$

Architectures

- have components with
  - interface specifications
  - implemented by
    - state machines
    - architectures
- are hierarchical forming systems of systems

System composition = logical and

Channel Hiding = existential quantification

Refining Systems

Technische Universität München
Institut für Informatik
D-85748 Garching, Germany

Horizontal Refinement

$$F: \vec{I} \rightarrow$$

is refined by a

$$\hat{F}: \vec{I} \rightarrow$$

if

$$\forall x \in \vec{I}: \hat{F}(x) \subseteq F(x)$$

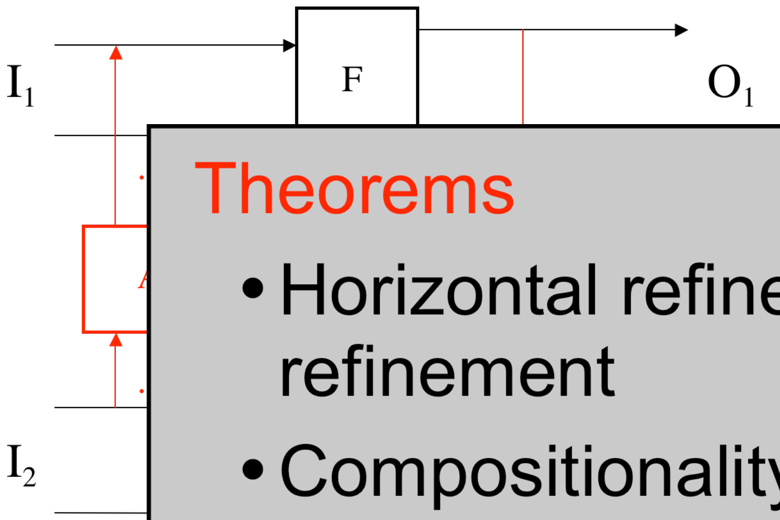
we write

$$F \approx\!>_{\mathbb{F}} \hat{F}$$

Compositionality of refinement

$$\frac{\forall k: F_k \approx\!>_{\mathbb{F}} \hat{F}_k}{\otimes\{F_k : k \in \mathbb{K}\} \approx\!>_{\mathbb{F}} \otimes\{\hat{F}_k : k \in \mathbb{K}\}}$$

Vertical Refinement: Changing Levels of Abstraction



Refiner
Given t
state tra

we call

such th

and for

Theorems

- Horizontal refinement implies vertical refinement
- Compositionality of vertical refinement
- Vertical refinement distributes over composition
- Abstractions of vertical refinements of implementations are vertical refinements of abstractions
- Vertical refinement is a Galois connection

Expressive power: time

- The system model can express timing properties

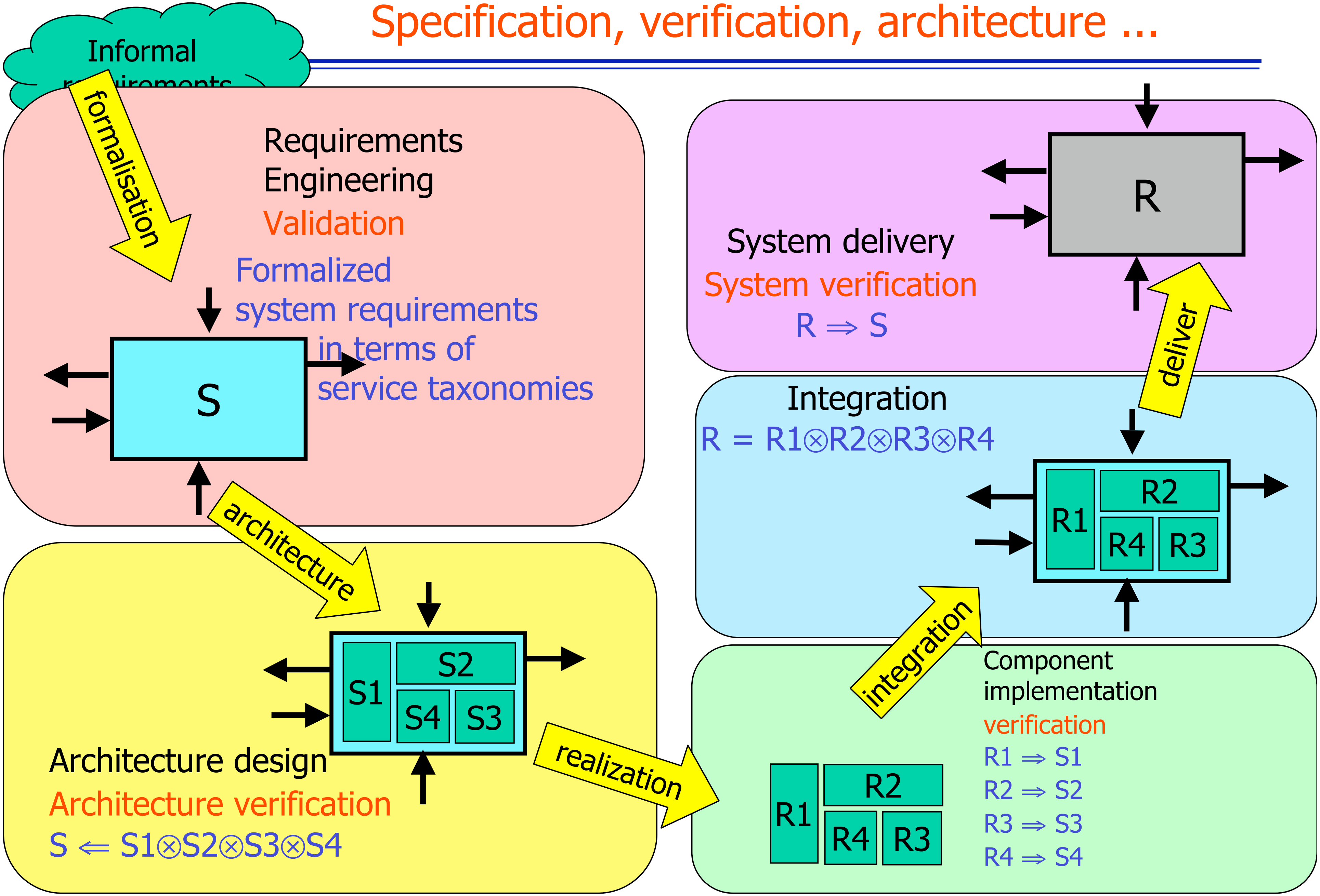
$$y \in TMC(x) \Rightarrow \{m\}\#x{\downarrow}t \leq \{m\}\#y{\downarrow}t+\delta$$

- The time granularity can be refined
  - a special case of vertical refinement

Mandatory properties

- Concept of interface - interface abstraction
  - syntactic interface
  - interface specification - behaviour
  - verification of interface properties
  - relating components (compatibility, refinement)
  - behavioural abstraction
- Implementation
  - interface abstraction - correctness
  - verification of properties - testing, model checking and deductive proofs
- Composition
  - architectures
  - compositional
    - behavioural specification
    - implementations
    - architectures
  - hierarchical (system of systems)
  - modular for refinement
- Further aspects
  - sub-functions (function hierarchy)
  - time
  - probability
  - performance
  - ...

Specification, verification, architecture ...



Open Issues

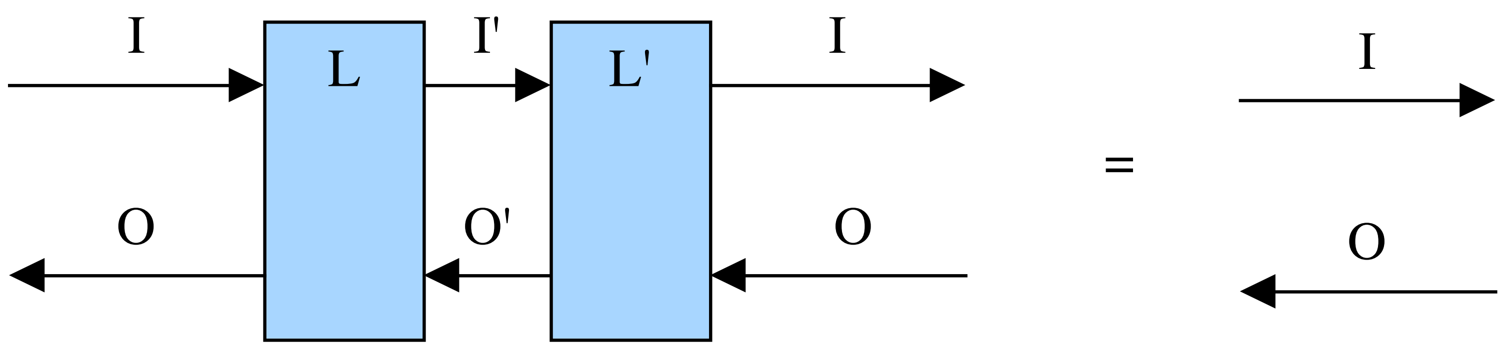
- Probability
- Non-functional properties
- Modelling of
  - hardware issues
  - mechanical aspects

Refinement layers

- A layer refinement pair are two layers that form the time independent identity

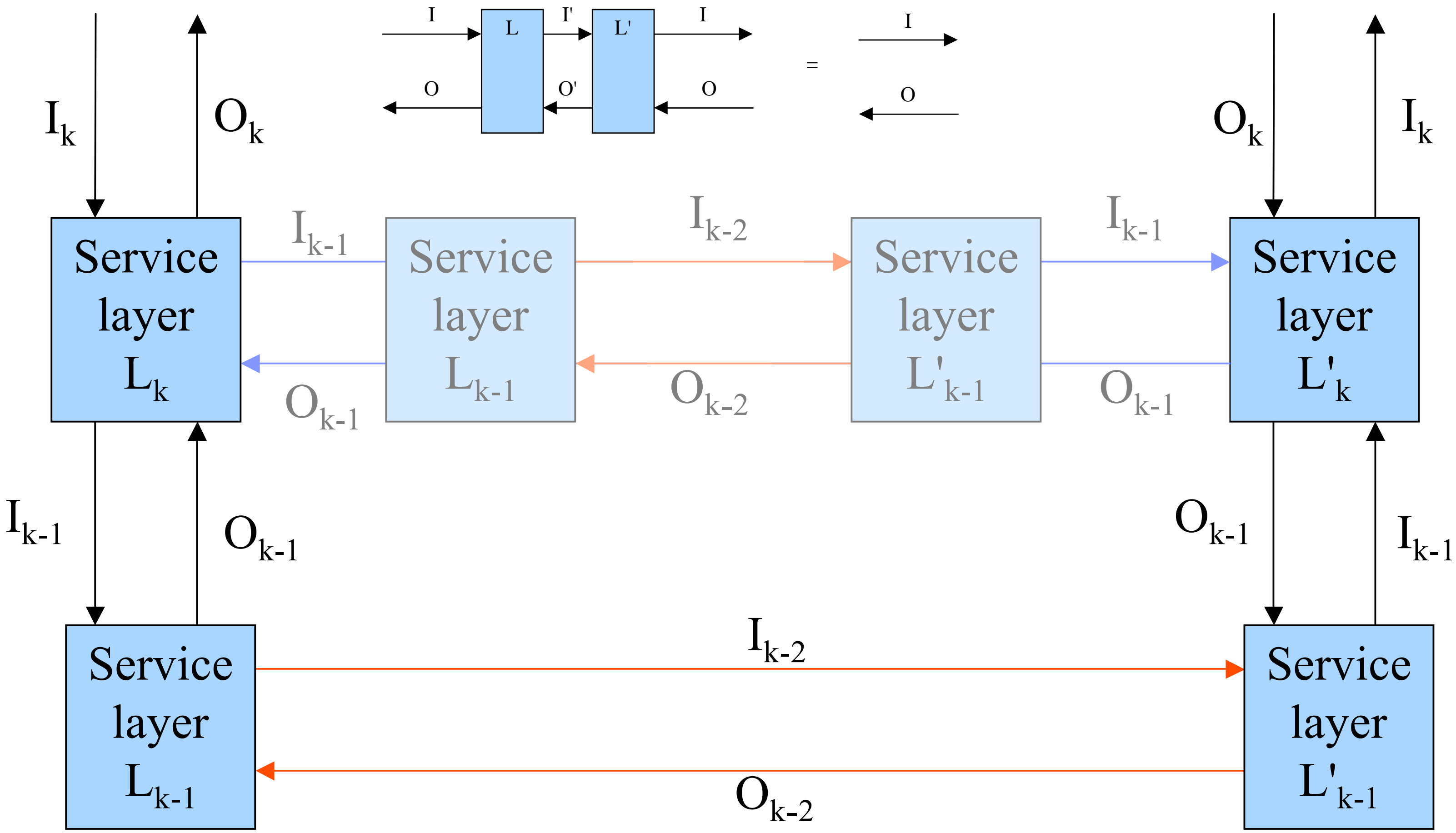
Two layers L and L' are called a *refinement pair* for if

$$L \otimes L' = Id(I \blacktriangleright O)$$



Layered protocols

Remember



The comprehensive model