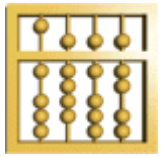
Rethinking Functional Requirements:
Novel Approaches to Categorizing Requirements

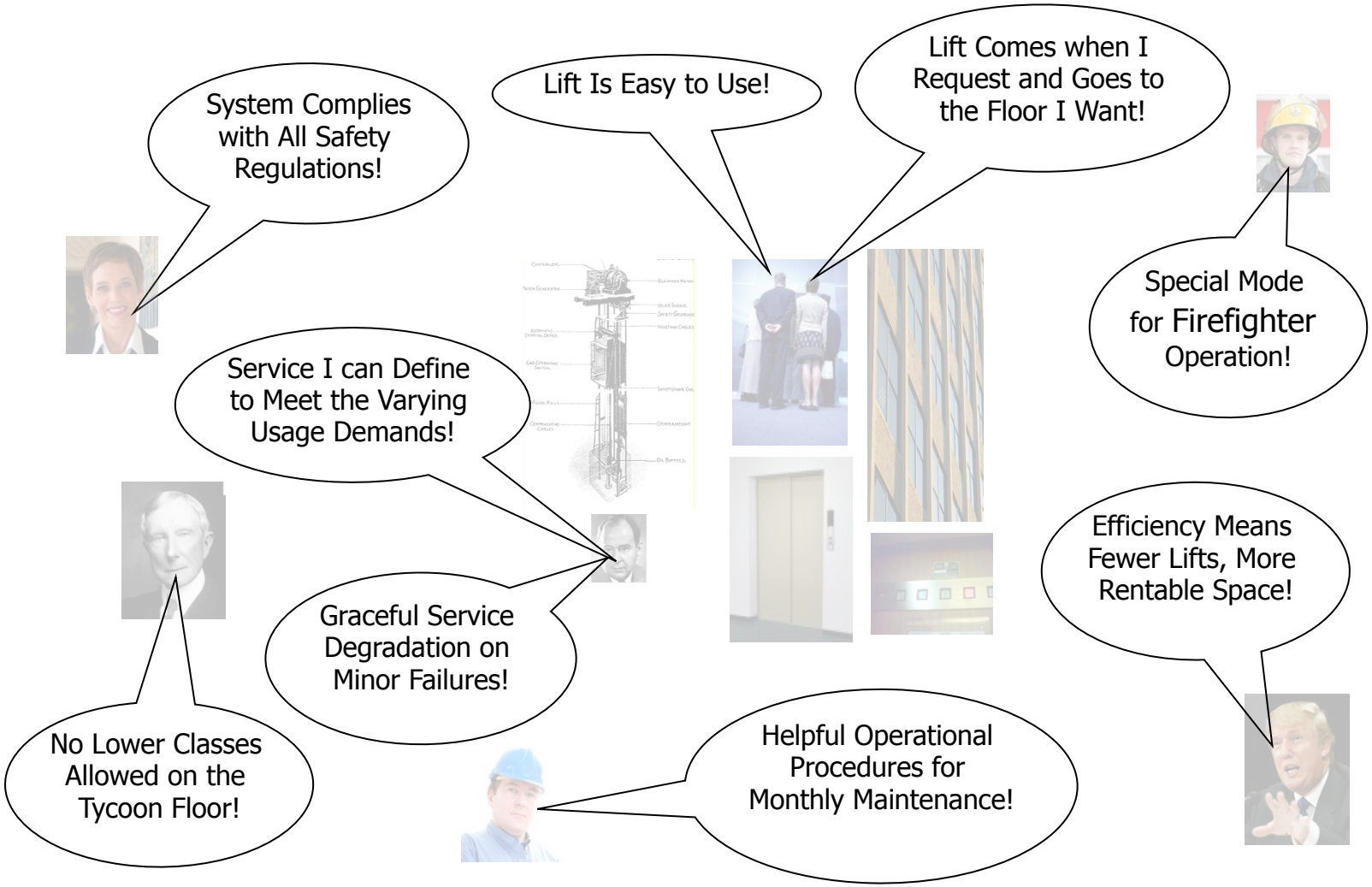
The conceptual power of mathematical modelling

Manfred Broy

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



Diversity of requirements: A slide due to Michael Jackson

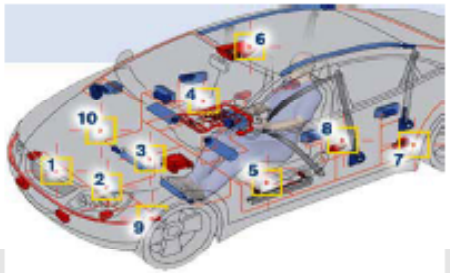


A slide due to Michael Jackson

Air Bag

Beispiel: **BMW X5**
5-Sterne im Euro-NCAP
(Maximalbewertung)

Crash Euro-NCAP
64km/h
Deformierbare Barriere
50% Offset



0 ms

Permanent werden alle Daten der Sensorik (1, 2, 4, 8) im Steuergerät (3) ausgewertet

40 ms

Bereits die ersten cm einer Deformation erzeugen eine Beschleunigung, die ausreicht einen Crash zu erkennen.
In der ca. 12ten Millisekunde (0,12 sec) ist die Entscheidung im Steuergerät getroffen die Airbags zu aktivieren.

60 ms

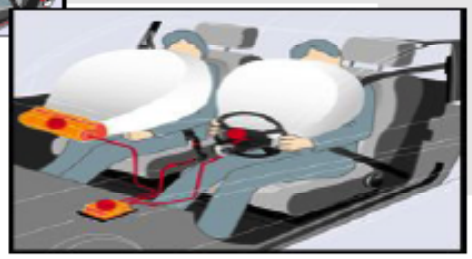
Die Airbags beginnen bereits in der Vorwärtsverlagerung der Passagiere sich zu entfalten.

80 ms

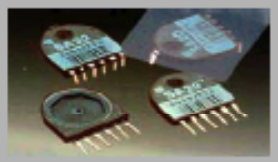
Nun ist die Schutzwirkung in voller Aktion.

400 ms

Die hohe Dynamik des Crashes zeigt sich in der weiteren Bewegung des Fahrzeugs.
Nun wird:
- Zentralverriegelung geöffnet
- Warnblinkanlage aktiviert
- Telefonnotruf abgesetzt.



Mikromechanische Sensorik



Example: A Short Example: airbag

1. The airbag has to be activated after the crash sensor indicates a crash within the interval of 160 to 180 millisec.
   - ◇ a problem domain requirement,

   functional
2. The probability that the airbag fails to fulfil specification (1) is less than 0,01 %.
   - ◇ a reliability/safety requirement,

   functional
3. The probability that the airbag is activated with 170 ± 5 millisec after the crash sensor indicates a crash is above 99.9 %.
   - ◇ a performance/safety requirement,

   functional
4. The probability that the airbag gets activated without a signal from the crash sensor is below 10^{-8} %.
   - ◇ a probabilistic safety property - a problem domain requirement.

   functional
5. The sidebag airbag is ordered as an extra by more than 80 % of the customers.
   - ◇ a business requirement

   non-functional

Motivation

- A clear understanding of the notion of "functional" (and "non-functional") requirements
- Categorizing requirements as a basis for structuring requirements
  - ◇ Based on system models
  - ◇ Based on quality models
  - ◇ Based on context structuring
- Understanding the relationship between different categories of requirements
- Having a formal basis that – at least in principle – supports the formulation of specifications about time and probability

Nonfunctional requirement

- Wikipaedia
  - ◇ a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors
- University of Toronto
  - ◇ Non-functional requirements are global constraints on a software system
- http://www.requirementsauthority.com
  - ◇ A Non-Functional Requirement is usually some form of constraint or restriction that must be considered when designing the solution.
- IEEE
  - ◇ non functional requirement: a software requirement that describes not what the software will do, but how the software will do

Some statements due to Martin Glinz

- Glinz writes "*Although the term non-functional requirements has been in use for more than twenty years, there is no consensus in the requirements engineering community what non-functional requirements are and how we should elicit, document and validate them*".
- However, in the end his particular definition is equally imprecise and fuzzy.
  - ◇ in his definition a non-functional requirement is an "attribute of" or a "constraint on" a system.
  - ◇ very unsatisfactory as a definition, because it reduces the fuzzy term "non-functional" requirement to the equally fuzzy and general terms such as "attribute" or "constraint".
  - ◇ not obvious why system functionality and functional requirements should not also be described by attributes or constraints.

A characterization due to Martin Glinz

M. Glinz:
On Non-Functional Requirements.
In: 15th IEEE International Requirements Engineering Conference (RE '07), 2007

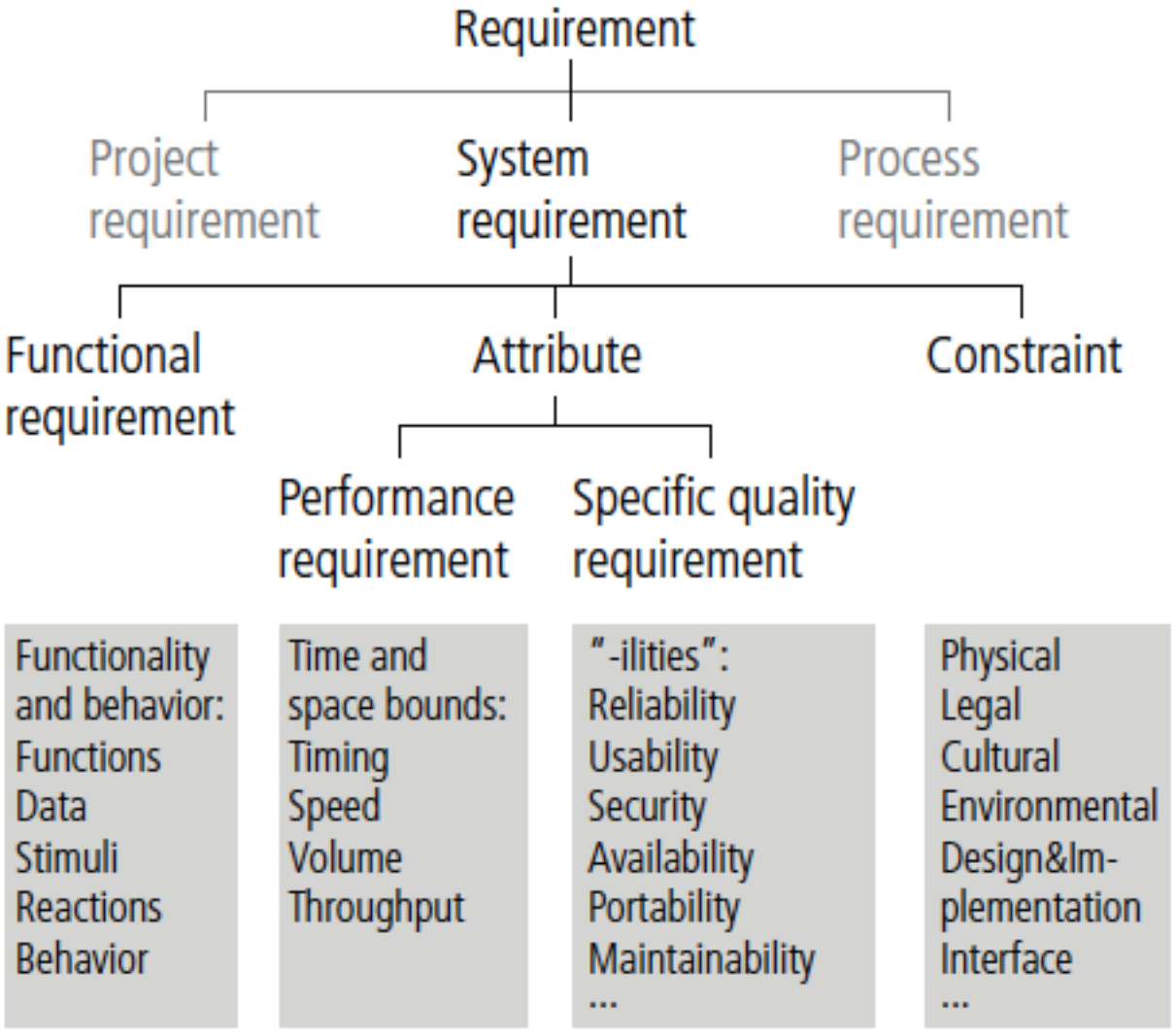


Figure 2. A concern-based taxonomy of requirements

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 the point of view of the context (interface behavior, dynamic interface, interaction view)
- a structure and distribution with an 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

Systems: the model

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

$STREAM[T] = \{IN \setminus \{0\} \to T^*\}$

valuation of channel set C

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

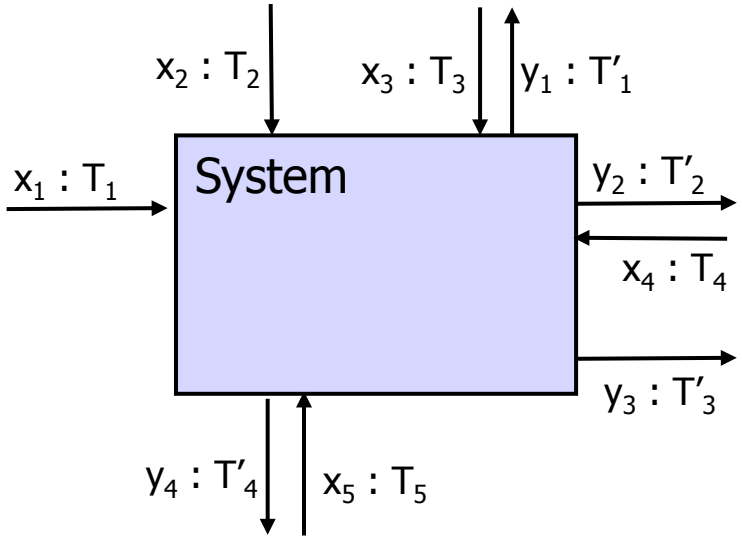
interface behaviour for syn. interface $(I \blacktriangleright O)$

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

interface specification

$p: I \cup O \to IB$

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



Continuous systems: the model

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

$(I \blacktriangleright O)$

continuous data stream of continuous type M

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

valuation of channel set C

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

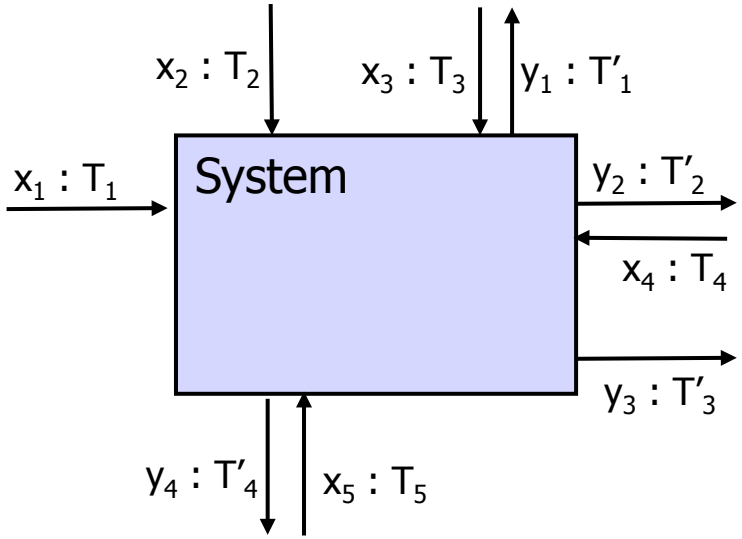
interface behaviour for syn. interface $(I \blacktriangleright O)$

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

interface specification

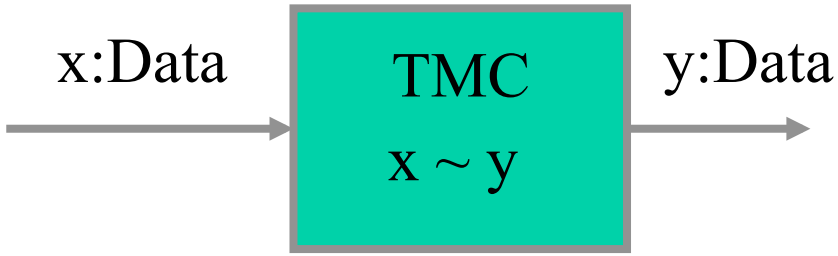
$p: I \cup O \to IB$

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. To appear in: Advances in Real-Time Systems. Springer

Example: System interface specification



A transmission component TMC

TMC
in x: Data **out** y: Data
x ~ y

See: M. Broy, K. Stølen: Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement. Springer 2001

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

Verification: Proving properties about specified components

From the interface assertions we can prove

- Safety properties

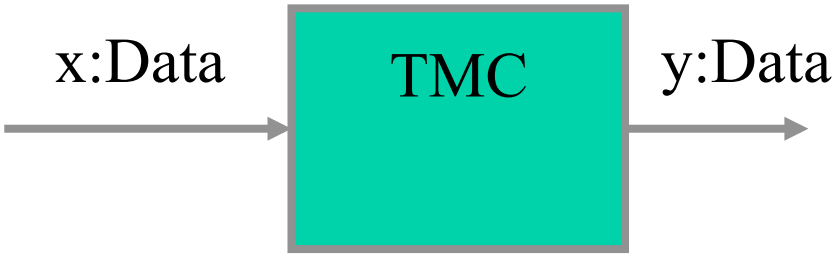
$$\{m\}\#y > 0 \wedge y \in TMC(x) \Rightarrow \{m\}\#x > 0$$

- Liveness/progress properties

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

Example: System interface specification - timing

A transmission component TMC



TMC(δ: Time)

in x: Data

out y: Data

$\forall\, t \in \text{Time}, m \in \text{Data}:$

$\{m\}\copyright x\downarrow t \leq$

$\{m\}\copyright y)\downarrow(t+\delta+1) \leq$

$\{m\}\copyright x)\downarrow(t+\delta)$

The principles of model based software and systems engineering

- The triangle
  - ◇ Mathematical modeling concepts (denotational semantics)
  - ◇ Logical system property calculus (deductive semantics)
  - ◇ System implementation concept (operational semantics)
- Concept of unit and of compositions – for all elements of the triangle
  - ◇ Modularity
- Concept of refinement
- Concept of semantic relationship and semantic coherence
- Concept of semantic dependencies
  - ◇ Beyond tracing

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

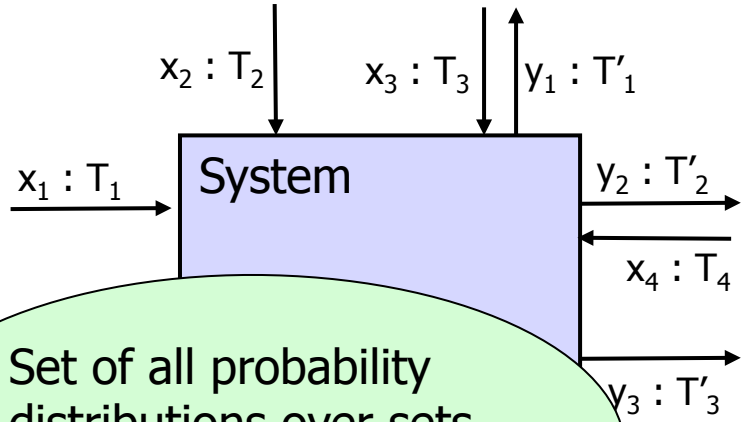
$STREAM[T] = \{IN \backslash \{0\} \to T^*\}$

valuation of channel set C

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

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

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



interface specification

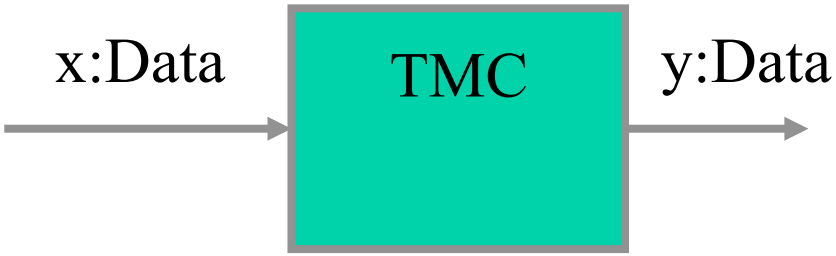
$p: I \cup O \to IB$

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

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

Example: System interface specification - timing

A transmission component TMC



TMC(δ: Time)

in x: Data

out y: Data

$\forall\, t \in \text{Time}, m \in \text{Data}:$

$\mathbf{P}(\{m\}\copyright x\downarrow t \leq$

$\{m\}\copyright y)\downarrow(t+\delta+1) \leq$

$\{m\}\copyright x)\downarrow(t+\delta)) > 0.99$

A refined model of behavior: probability

- A model of functional correctness - a qualitative model

$$F_{logic}: IH[I] \rightarrow \wp(IH[O])$$

- A model of probability - quantitative model

$$F_{probabilistic}: IH[I] \rightarrow \{(\mu, Y): Y \subseteq IH[O]\}$$

where (μ, Y) denotes a probability distribution μ over set Y

The overall system views

	Syntactic	Behavior	
		Logical	Probabilistic
Black Box Interface	Syntactic interface	Logical interface behavior	Probabilistic interface behavior
Glass Box			
Architecture	Data flow graph Syntactic component interface	Logical component interface behavior	Probabilistic component interface behavior
State	State space	Logical state machine	Probabilistic state machine

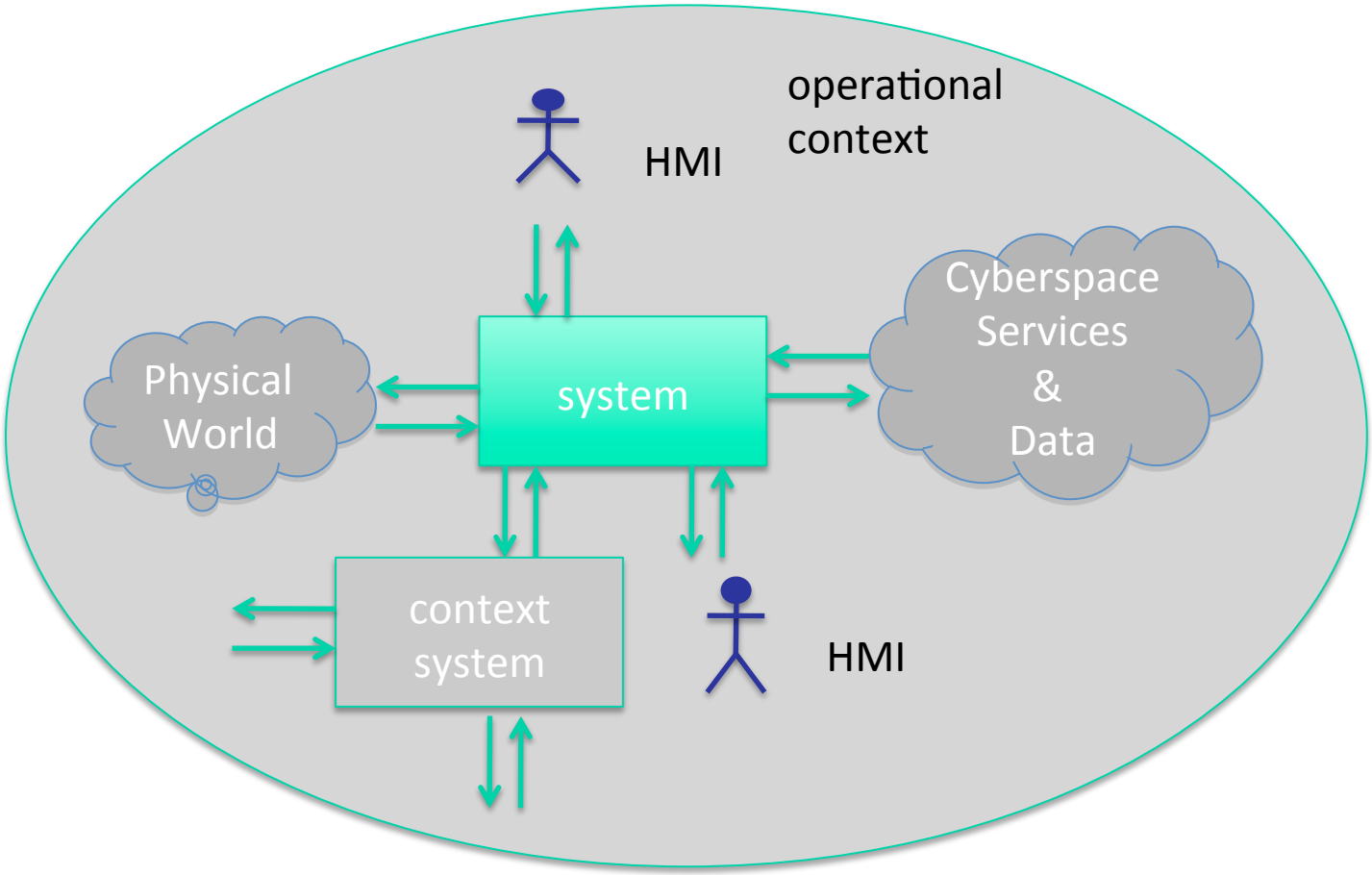
Functional requirements

	Syntactic	Behavior	
		Logical	Probabilistic
Black Box Interface	Syntactic interface	Logical interface behavior	Probabilistic interface behavior
Glass Box			
Architecture	Data flow graph Syntactic component interface	Logical component interface behavior	Probabilistic component interface behavior
State	State space	Logical state machine	Probabilistic state machine

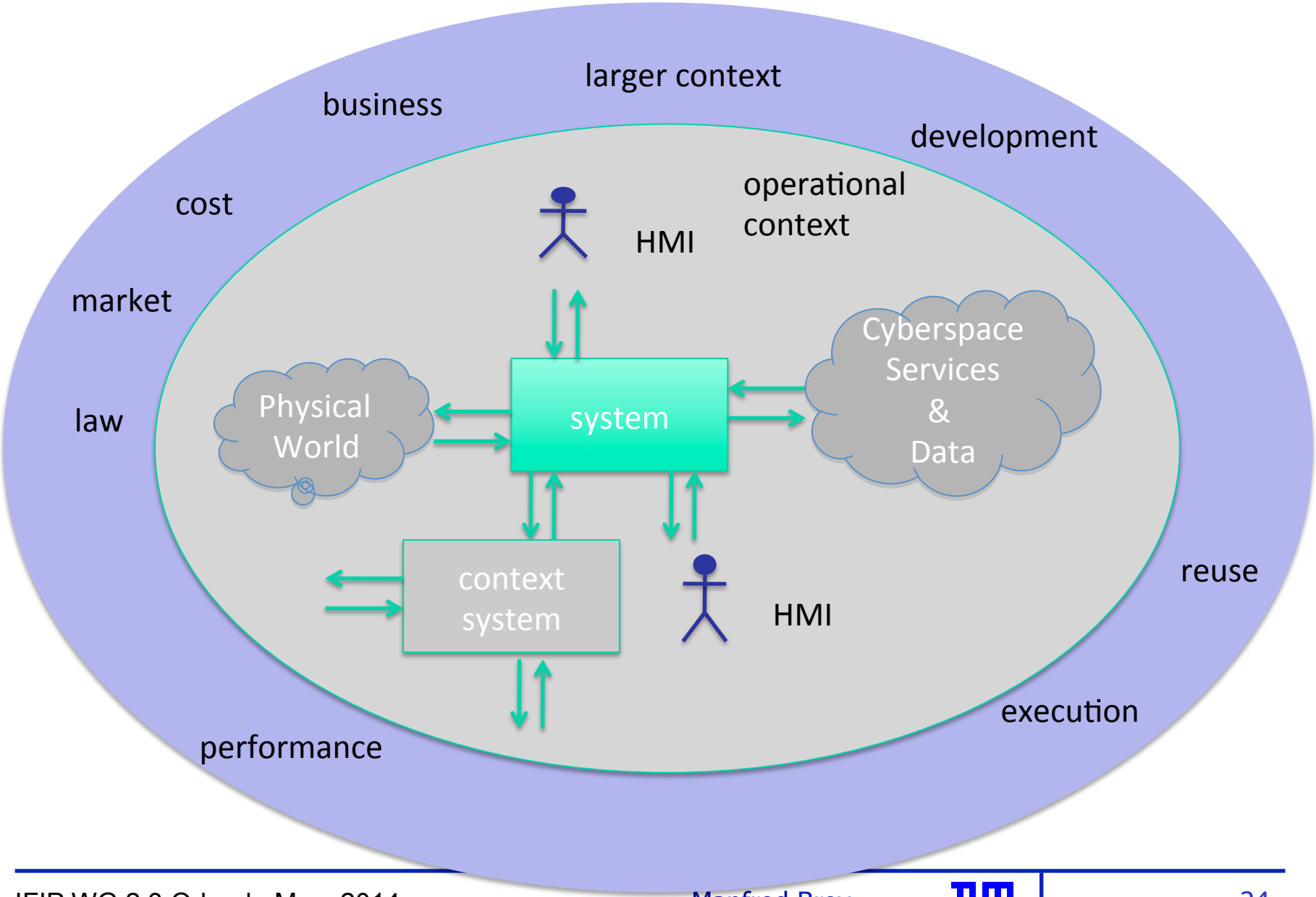
The overall system views

	Syntactic	Behavior	
		Logical	Probabilistic
Black Box Interface	Syntactic interface	Logical interface behavior	Probabilistic interface behavior
Glass Box			
Architecture	Data flow graph Syntactic component interface	Logical component interface behavior	Probabilistic component interface behavior
State	State space	Logical state machine	Probabilistic state machine

System and its operational context

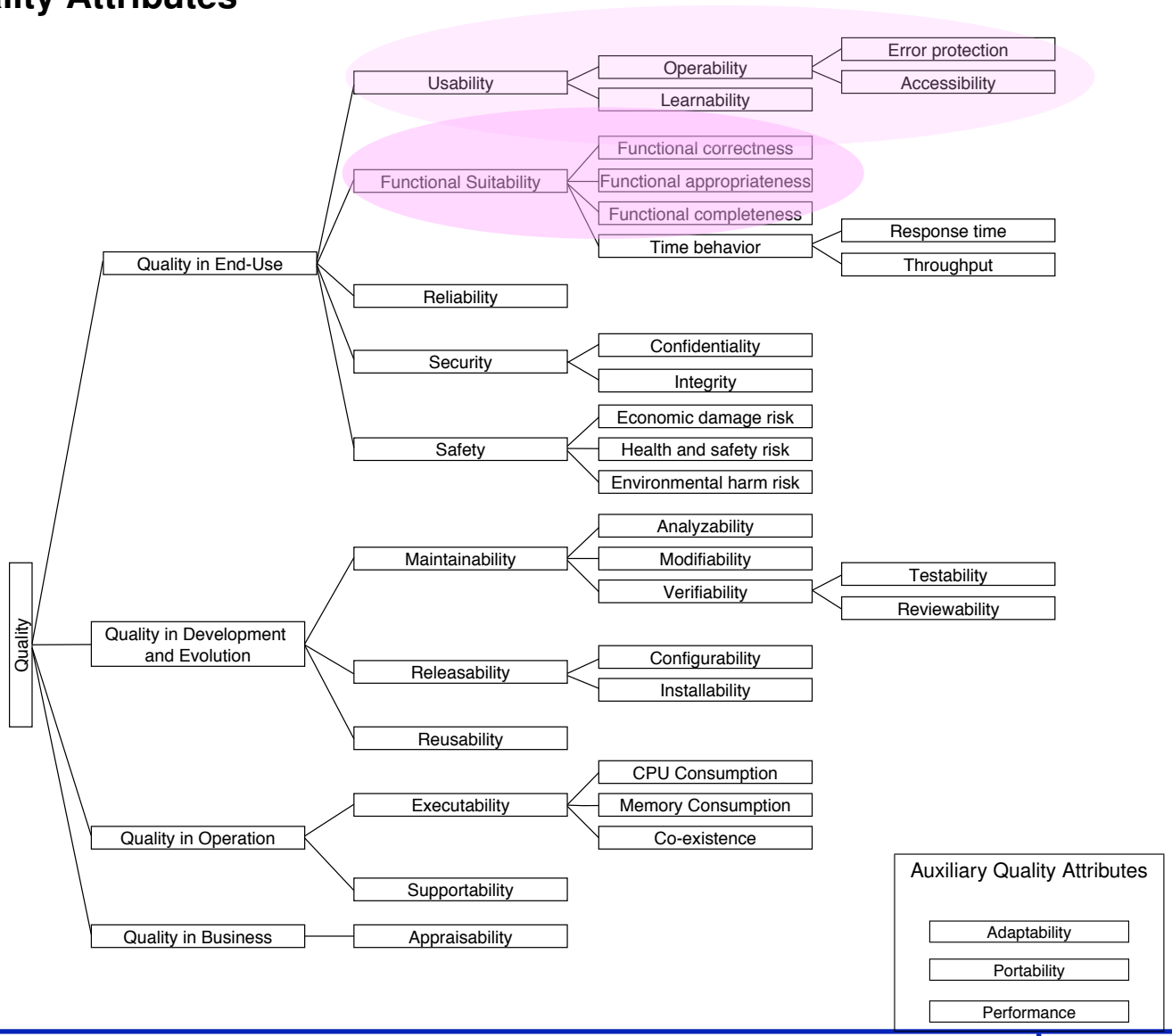


System and its wider context



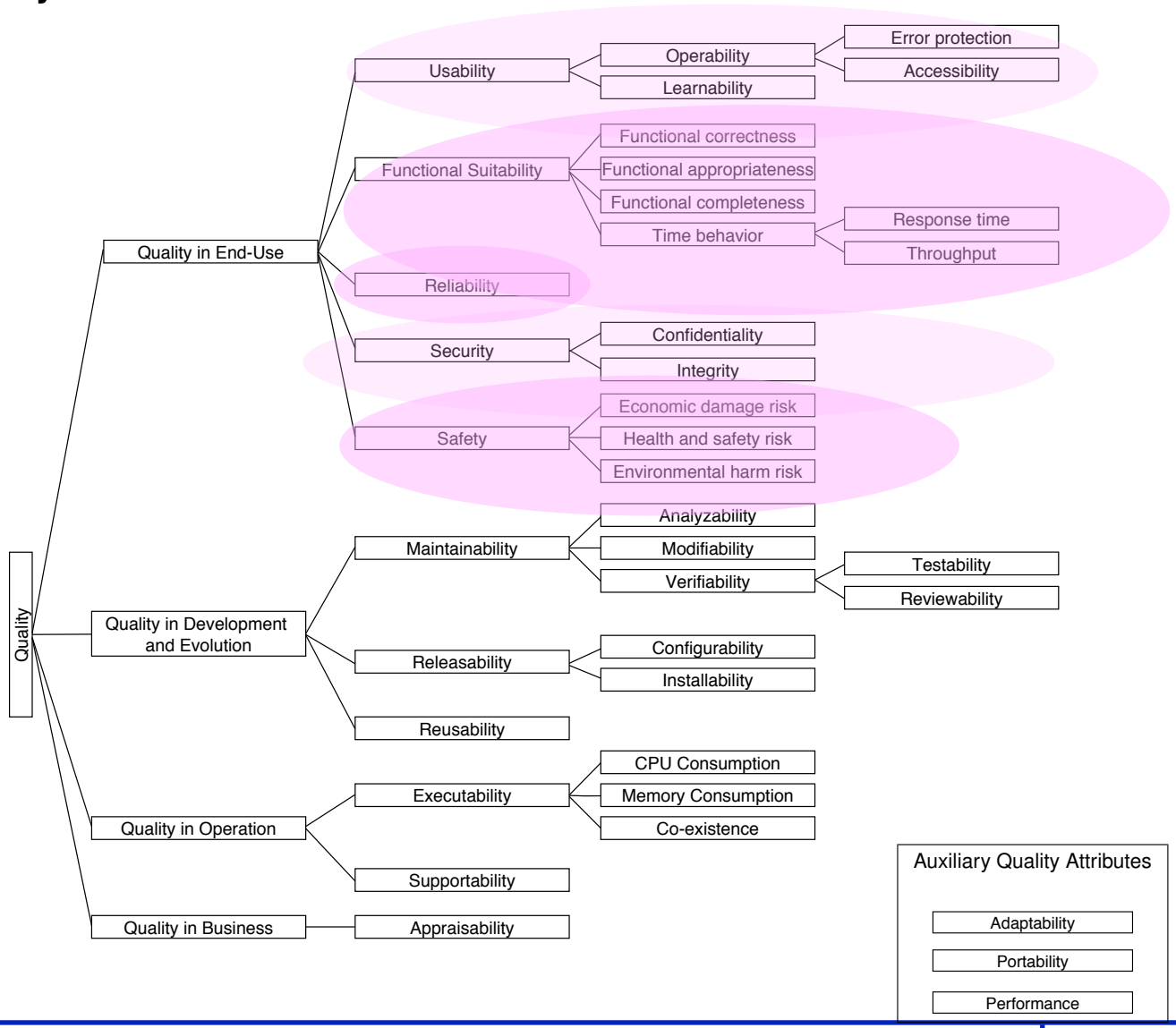
Quality model: functional requirements – conventional view

Main Quality Attributes



Quality model: functional requirements – novel view

Main Quality Attributes



Classification

- Functional requirements: logical and probabilistic interface behavior (including faults):
  - ◇ functional features
  - ◇ safety
  - ◇ reliability
  - ◇ …
- Architectural requirements: logical and probabilistic sub-system interface behavior (including faults)
  Quality requirements such as:
  - ◇ Performance
  - ◇ Security
- Requirements related to system context
  - ◇ Usability
  - ◇ Business - Return on investment

Conclusion

Probability

- Probabilistic system models and specifications are refinements of nondeterministic system models and specifications
- A rich set of so-called "non-functional" properties is captures by probabilistic interface specifications and thus become functional

Time

- Time-aware system models and specifications are refinements of non-time-aware system models and specifications
- For time critical systems so-called "non-functional" timing properties can be captured by by time-aware interface specifications and thus become functional

Performance

- There are two concepts of performance:
  - ◇ response time (in the case of non-time critical functionality)
  - ◇ efficient utilization of resources
- Response time is a functional property captured by time-aware probabilistic interface specifications

Conclusion

Interesting questions

- Systems show probabilistic behavior!
- Is software probabilistic?
  - ◇ If no random concepts are part of the program?
  - ◇ If the operational context is probabilistic?
  - ◇ If the execution platform is understood to be "probabilistic"?
- Remark on refinement of probabilistic specifications, architecture models and implementations
  - ◇ A system behavior specification with a specified probabilistic distribution cannot be further refined!
  - ◇ General probabilistic specifications define sets of propabilistic behaviors with specific probability distributions.