# Modeling Software and System Reliability

Manfred Broy
Lehrstuhl für Software & Systems Engineering

Technische Universität München
Institut für Informatik

## From programs to cyber-physical systems

```
/* $Id: GlobalOptions.java.in,v 4.3.2.2 2002/05/28 17:33:58
hoenicke Exp $
 */

package jode;
import java.io.PrintWriter;
import java.util.StringTokenizer;

public class GlobalOptions {
    public final static String version = "1.1.2-pre1";
    public final static String email = "jochen@gnu.org";
    public final static String copyright =
        "Jode (c) 1998-2001 Jochen Hoenicke <"+email+">";
    public final static String URL = "http://jode.sourceforge.net

    public static PrintWriter err = new PrintWriter(System.err,
true);
    public static int verboseLevel  = 0;
    public static int debuggingFlags = 0;

    public static final int DEBUG_BYTECODE = 0x001;
    public static final int DEBUG_VERIFIER = 0x002;
    public static final int DEBUG_TYPES    = 0x004;
    public static final int DEBUG_FLOW     = 0x008;
    public static final int DEBUG_INOUT    = 0x010;
    public static final int DEBUG_ANALYZE  = 0x020;
    public static final int DEBUG_LVT      = 0x040;
    public static final int DEBUG_CHECK    = 0x080;
```

- Programs:
  - mappings states to states or data to data,
  - supposed to terminate,
  - time and interaction not an issue,
  - concept of computation: Turing machines – algorithms
- Cyber-physical systems:
  - connected to the physical world,
  - needs a coherent model of context, interface, interaction, time, architecture, state, probability, data and event flow
  - concept of computation: interaction, generalized Mealy machines
  - extended requirement for dependability

## 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

- Classical correctness: Given: set $T \subseteq A^*$ of streams of correct output sequences

  output $t' \in A^*$ 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 $t' \in A^*$ 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 $t' \in A^*$ is roughly correct – formalized in fuzzy logic

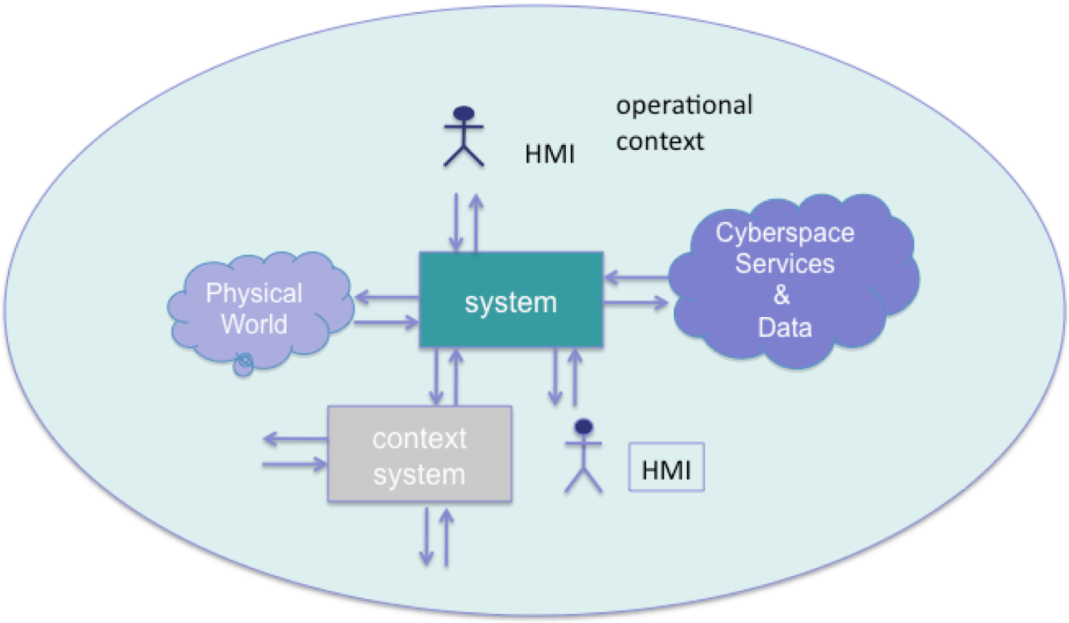


## **Reliability** as an element of Dependability

Comprehensive view **dependability**:

- **Availability** - readiness for 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

# System and its context






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

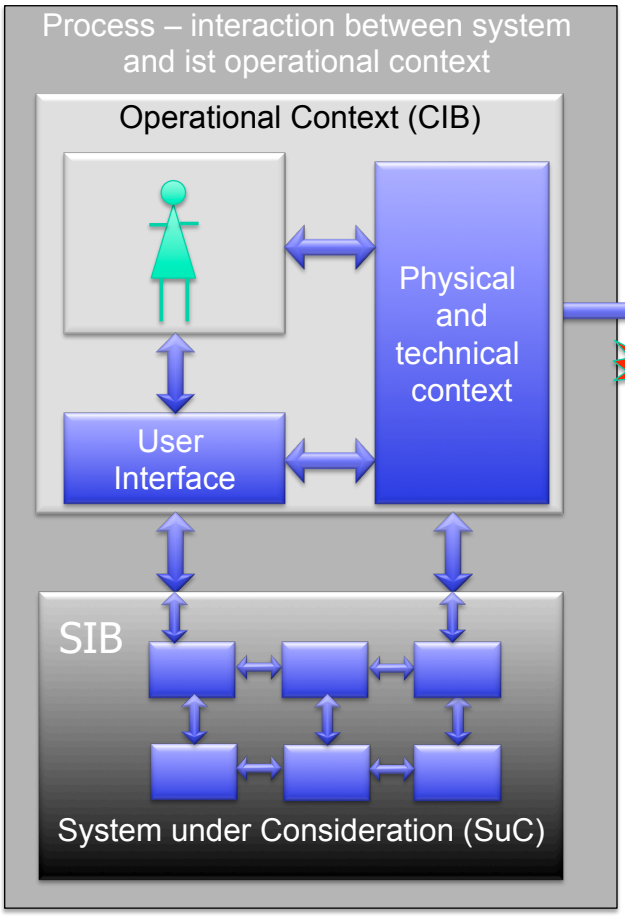
A system has

- 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

# System Views

- Operational Context View (CIB – context interface behavior)
  - Behavior of the operational context
- Interface View: System Interface Behavior (SIB)
  - Functional View: Interface Behavior
  - Functional features: hierarchy and feature interaction
- Interaction between CIB and SIB:
  - Observable behavior: process OBS
- Architectural View
  - Hierarchical decomposition in sub-systems
  - Sub-system behavior
- State View
  - State space
  - State transition






A depandability view onto a system and its context

No failure:

CIB ∧ SIB ⇒ No_failure(OBS)

# Discrete systems: the modeling theory

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

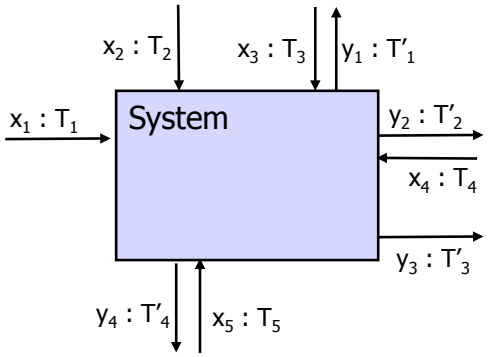
$STREAM[T] = \{\mathbb{N}\backslash\{0\} \to T^*\}$

valuation of channel set C

$\mathbb{H}[C] = \{C \to STREAM[T]\}$

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

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




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



# System interface behaviour - causality

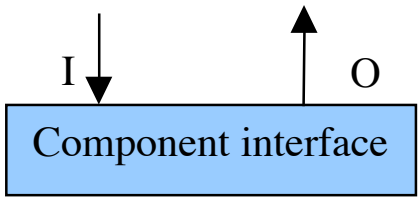
$(I \blacktriangleright O)$ *syntactic interface* with set of input channels I and of output channels O

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

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

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

A system shows a total behavior

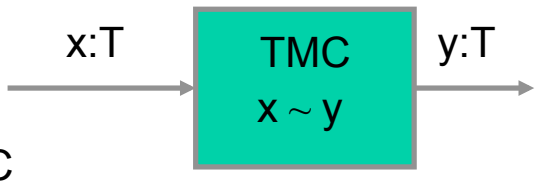


## Characteristics of the model

- Essential: interface behavior
- Time:
  - Causality – modeling time flow
  - System time vs. physical time
  - time requirements vs. execution time
- Interaction: sequence of steps
  - Context modeling – and the interaction between system and environment
- Non termination: Systems run without time limits
- Composition with the environment
  - Functional safety – no hazards
  - Security
  - Reliability
  - ...



## Example: System interface specification




A transmission component TMC

TMC

| |
|---|
| **in** x: T<br>**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

From the interface assertions we can derive properties!

Specification:

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

Strong causality:

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

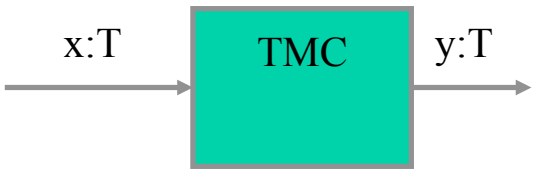
From which by choosing z such that

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

we can deduce (note then $m\#x{\downarrow}t = m\#z$ )

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

## Example: TMC with Timing Restrictions

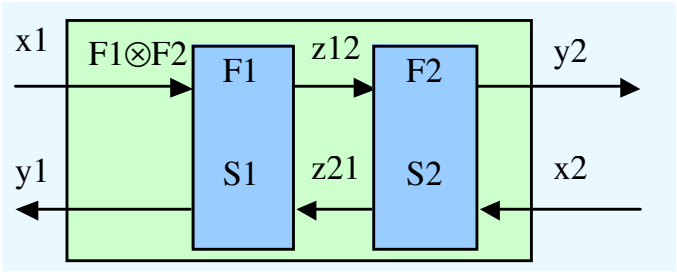



TMC

| |
|---|
| **in** x: T<br>**out** y: T |
| ∀ t ∈ IN: ∀ m ∈ T:<br>m#(y↓t+delay) ≤ m#(x↓t) ≤ m#(y↓t+delay+deadline) |



## Modularity: Rules of compositions for interface specs




F1

| |
|---|
| **in** x1, z21: T<br>**out** y1, z12: T |
| S1 |

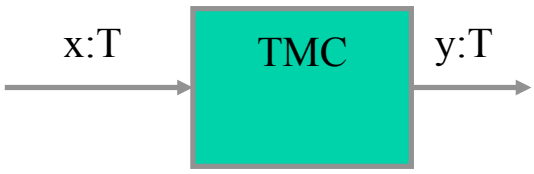
F2

| |
|---|
| **in** x2, z12: T<br>**out** y2, z21: T |
| S2 |

F1⊗F2

| |
|---|
| **in** x1, x2: T<br>**out** y1, y2: T |
| ∃ z12, z21: S1 ∧ S2 |

## Example: TMC with Probability Restrictions




TMC

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



## 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

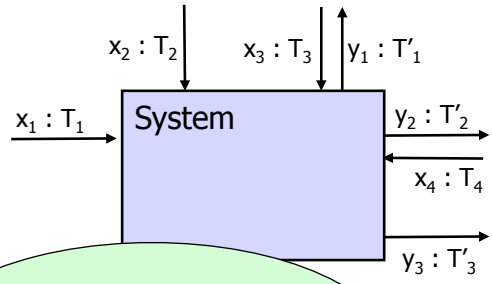
$STREAM[T] = \{\mathbb{N}\backslash\{0\} \to T^*\}$

valuation of channel set C

$\mathbb{H}[C] = \{C \to STREAM[T]\}$

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

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




Set of all probability distributions over sets of output histories

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

## Probabilistic Behavior Composition

Probabilistic behavior

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

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

$F(X)[Y]$ for the probability

that the output is in $Y$ provided the input is in $X$



## Probabilistic Behavior Composition

Given probabilistic behaviors

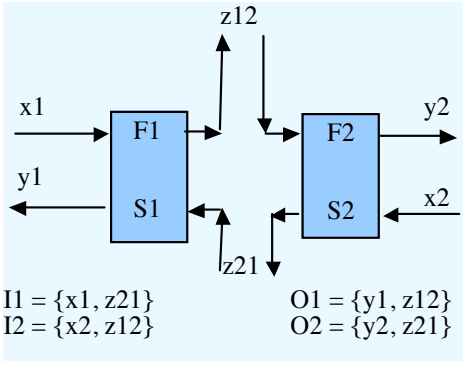
$F1: \wp(\mathbb{IH}[I1]) \to (\wp(\mathbb{IH}[O1]) \to [0{:}1])$

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

Where $O1 \cap O2 = \varnothing$; 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



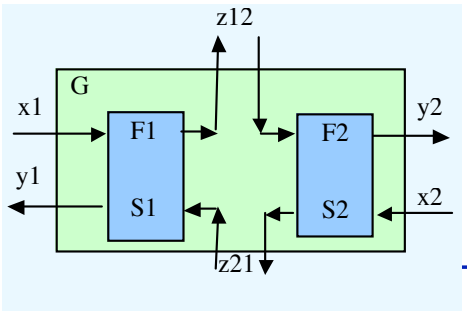

Composition: define

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

by

$G(X)[Y] = F1(X|_{I1})[\{y1: \exists\, y \in Y: y|_{O1} = y1\}] \times$

$F2(X|_{I2})[\{y2: \exists\, y \in Y: y|_{O2} = y2\}]$




Assuming probabilistic independence

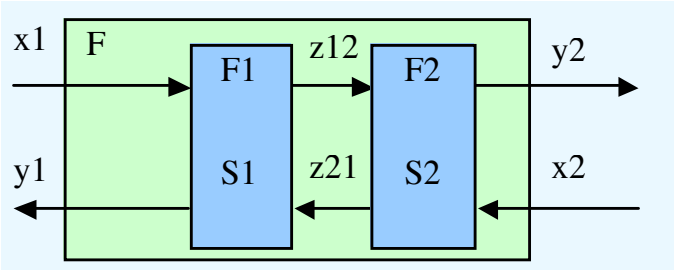
Given G we specify F = F1⊗F2

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

by

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

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






# Modelling Reliability & Availability

## Quantitative Properties

- Many interesting properties of systems have to be expressed **quantitatively,** using metrics or measures
- Examples
  - Resource Usage
  - System Operation Costs
  - **Dependability**
- Examples for **dependability metrics**
  - Uptime, Downtime
  - Reliability
  - Point-, Interval-, Steady State Availability



## Quantitative Specifications

Quantitative Specifications map observations about a system to a numeric value (i.e. the metric):

**Cantor Metric:** depends on the length of the longest common prefix of histories.

$$d: \mathbb{I}H[C] \times \mathbb{I}H[C] \to \mathbb{R} \cup \{-\infty, +\infty\}$$

$$d(x, x') = \text{glb}\ \{1/2^t:\ x \downarrow t = x' \downarrow t\ \}$$

Chatterjee, Henzinger, Jobastman, Singh: Measuring and Synthesizing Systems in Probabilistic Environments, CAV 2010

## Measurability

Let D be an arbitrary set and F be a set of subsets of D. We call F a *field of sets if*

- $\varnothing \in F$,
- if A is a set in F then its complement $D \setminus A$ is in F, and
- if A and B are in F then their union $A \cup B$ is also in F.

F is called a *Borel field*, if it fulfills the additional property that for every countable enumeration of sets $A_1, A_2, \ldots \in F$ we get

- $\cup \{A_i : i \in \mathbb{N}\} \in F$



## Measurability

A function

$$\mu : F \to \mathbb{R} \cup \{-\infty, +\infty\}$$

from a Borel field F of sets to the extended real numbers $\mathbb{R} \cup \{-\infty, +\infty\}$ is called a *measure* if the following properties holds:

- $\mu(A) \geq 0$ for all $A \in F$
- $\mu(\cup \{A_i : i \in \mathbb{N}\}) = \Sigma \{\mu(A_i) : i \in \mathbb{N}\}$
  for all pairwise disjoint sets $A_1, A_2, \ldots \in F$,
  i.e. with $A_i \cap A_j = \varnothing$ for all $i \neq j$
  (the measure $\mu$ is then called *completely additive*)

## Measurability

The set D with a measure function $\mu$ defined on a field of sets F is called a *measure space* and the sets in F are called *measurable*.

Measure spaces are taken as the basis for probability theory.



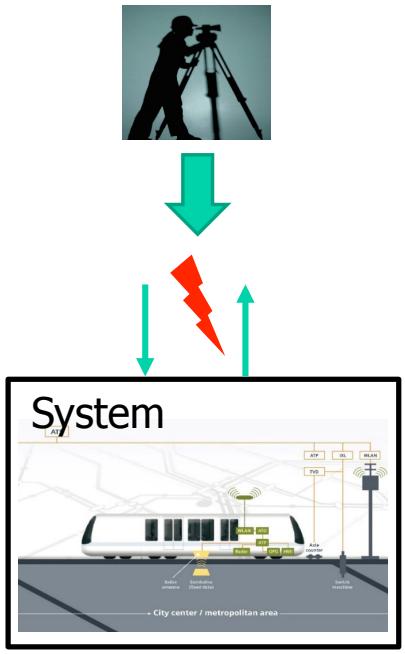
## Availability & Reliability

Our view on Availability & Reliability:

Availability & Reliability are properties of the (black-box) **interface behavior** of the system as observed by a user or external system.

Both quantify the amount of observable failures of the system.

What counts as failure needs to be explicitly defined!




See: M Junker, P Neubeck: A Rigorous Approach to Availability Modeling. Modeling in Software Engineering (MISE), 2012 ICSE Workshop

**Example: Reliability**

From reliability theory:

Reliability distribution

R(t) = **P**[lifetime of system is at least as long as t]





Given: set $Y \subseteq \mathbb{IH}[O]$ histories

- output $y` \in \mathbb{IH}[O]$ is correct, iff $y` \in Y$
- system with output set $Y` \subseteq \mathbb{IH}[O]$ is correct, iff $Y` \subseteq Y$

- Probabilistic system behavior: **P**: $\wp(\mathbb{IH}[O]) \to [0:1]$
  Correctness w.r.t Y: **P**[Y]

- Reliability: expected value $E_R$ of t for distribution
  $\mathbf{R}(t) = \mathbf{P}[\{y' \in \mathbb{IH}[O]: \exists\, y \in Y: y' \downarrow t = y \downarrow t\}]$
  is given by

$$\Sigma\, \{t \cdot \mathbf{R}(t): t \in \mathbb{N}\}$$

# Availability & Reliability Metrics

More sophisticated concepts of correctness and reliability:

- with which probability is the system output correct to which extent over which expected interval of time

Example: Availability

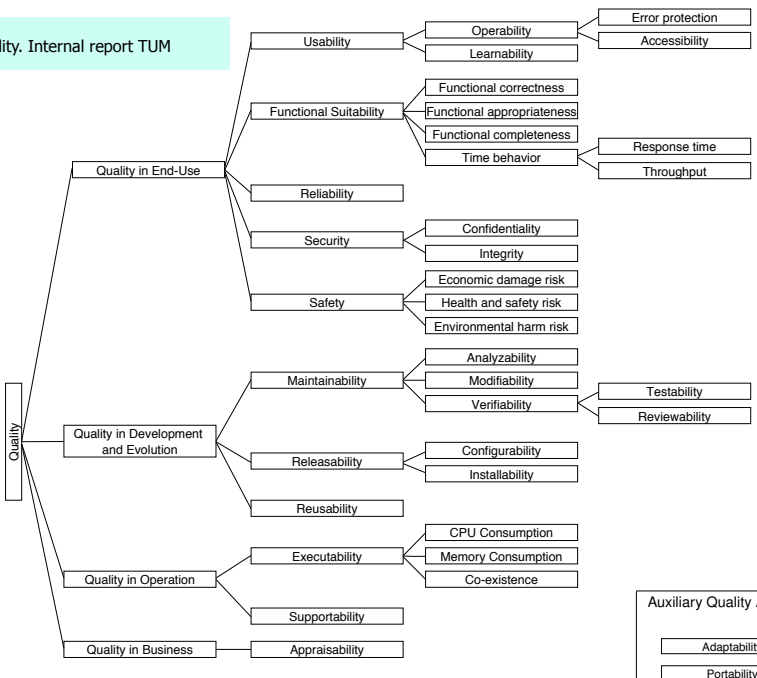
- An important metric for availability is the percentage of uptime.



# System Quality Models: Quality Concerns

**Main Quality Attributes**

K. Lochmann, S. Wagner:
A Quality Model for Software Quality. Internal report TUM



# A novel characterization of system properties and requirements

| | | Syntactic Basis | Behavior | |
|---|---|---|---|---|
| | | | Logical | Probabilistic |
| "External" Black Box View | Functional view | Syntactic Interface | Logical interface behavior | Probabilistic interface behavior |
| "Internal" Glass Box View | Architecture view | Hierarchical data flow graph of subsystems | Subsystems and their logical Interface behavior | Subsystems and their probabilistic Interface behavior |
| | State view | State space structure (Attributes) I/O messages | Logical state machine logical state transitions | Probabilistic state machine probabilistic state transitions |



# A novel characterization of functional requirements

| | Interface | | | Architecture | | | State | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Functional properties | | | | | | | | | |
| | Syntactic | Logical | Probabilsitc | Syntactic | Logical | Probabilsitc | Syntactic | Logical | Probabilsitc | Representation |
| Functional Suitability | ■ | ■ | ▒ | | | | | | | |
| Usability | ■ | ▒ | | | | | | | | ■ |
| Reliability | ■ | ▒ | ■ | | | ▒ | | | | |
| Security | ▒ | ▒ | ▒ | ▒ | | ▒ | | | | ▒ |
| Safety | ■ | ▒ | ■ | | | ▒ | | | | |
| Performance | ■ | ▒ | ■ | | | ▒ | | | | |
| Maintainability | ▒ | ▒ | ▒ | ▒ | ▒ | | ▒ | ▒ | | ■ |
| Reusability | ▒ | ▒ | | ■ | ■ | | ■ | ■ | | ▒ |
| Releasability | | | | | | | | | | ■ |
| Executability | | | | | | | | | | |
| Supportability | | | | | | | | | | |

## Rich Interface Specifications

- In a rich interface specification we speak about several views
- Example: Add probability
  Given:
  - logical interface behavior for syn. interface (I ▶ O)
    $$\{\mathbb{H}[I] \to \wp(\mathbb{H}[O])\}$$
  - probabilistic interface behavior for syn. interface (I ▶ O)
    $$\{\mathbb{H}[I] \to PD\,[\,\wp(\mathbb{H}[O])\,]\,\}$$
  - interface specification by an interface assertion q(x, y)
  - specify for each input history x = a probability distributions **P**(y|a) on the set of output histories
    $$\{y: q(a, y)\}$$



## Rich Specifications

In a rich specification we specify functional and "non-functional" properties of system functions

- logical interface behavior
- probabilistic interface behavior
- quality concerns
  - Usability
  - Time behavior
  - Reliability
  - Security
  - Safety
  - quality of service
  - ...

# Conclusion

- To model, specify and verify cyber-physical systems we need quantitative notions of behavior and correctness
- These models have to to coherent extensions of existing theories
- Such models support a variety of key notions
  - classical functional correctness
  - probabilistic correctness
  - quality attributes
- We want to express specifications:
  The system produces an output that is correct to a certain degree over a certain time span with a certain probability