From programs to cyber-physical systems

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

• 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
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

Process – interaction between system and its operational context

Operational Context (CIB)

User Interface

Physical and technical context

SIB

System under Consideration (SuC)

A dependability view onto a system and its context

Context/process observations (OBS)

External (observable) failure

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

\((I \to O)\)

data stream of type \(T\)

\[ \text{STREAM}[T] = \{\mathbb{N}\setminus\{0\} \to T^*\} \]

valuation of channel set \(C\)

\[ \mathbb{I}[C] = \{C \to \text{STREAM}[T]\} \]

interface behaviour for syn. interface \((I \to O)\)

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

System interface behaviour - causality

\((I \to O)\) syntactic interface with set of input channels \(I\) and of output channels \(O\)

\[ F : \mathbb{I}[I] \to \wp(\mathbb{I}[O]) \] semantic interface for \((I \to O)\)

with timing property addressing strong causality

let \(x, z \in \mathbb{I}[I], y \in \mathbb{I}[O], t \in \mathbb{N}\):

\[ x_{\downarrow t} = z_{\downarrow t} \Rightarrow \{y_{\downarrow t+1} : y \in F(x)\} = \{y_{\downarrow t+1} : y \in F(z)\} \]

\(x_{\downarrow t}\) prefix of history \(x\) of length \(t\)

A system shows a total behavior

Component interface
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

\[
\text{TMC} \\
\text{in } x : T \\
\text{out } y : T \\
x \sim y
\]

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

Specifying interface assertion
Verification: Proving properties about specified systems

From the interface assertions we can prove

- Safety properties

\[ m\#y > 0 \land y \in \text{TMC}(x) \Rightarrow m\#x > 0 \]

- Liveness properties

\[ m\#x > 0 \land y \in \text{TMC}(x) \Rightarrow m\#y > 0 \]

Verification: adding causality

From the interface assertions we can derive properties!

**Specification:**
\[ y \in \text{TMC}(x) \Rightarrow (\forall m \in T: m\#x = m\#y) \]

**Strong causality:**
\[ x_{\downarrow t} = z_{\downarrow t} \Rightarrow \{y_{\downarrow t+1}: y \in \text{TMC}(x)\} = \{y_{\downarrow t+1}: y \in \text{TMC}(z)\} \]

From which by choosing \( z \) such that
\[ \forall m \in T: m\#(z_{\uparrow t}) = 0 \]
we can deduce (note then \( m\#x_{\downarrow t} = m\#z \))
\[ y \in \text{TMC}(x) \Rightarrow \forall t \in \text{Time}, m \in T: m\#(y_{\downarrow t+1}) \leq m\#(x_{\downarrow t}) \]
Specification of Timing Properties

Example: TMC with Timing Restrictions

TMC

\[
\begin{array}{c}
\text{in } x: T \\
\text{out } y: T \\
\forall t \in \mathbb{IN}: \forall m \in T:
\end{array}
\]

\[
\begin{array}{c}
m#(y \downarrow t + \text{delay}) \leq m#(x \downarrow t) \leq m#(y \downarrow t + \text{delay} + \text{deadline})
\end{array}
\]

Modularity: Rules of compositions for interface specs

F1

\[
\begin{array}{c}
\text{in } x_1, z_{21}: T \\
\text{out } y_1, z_{21}: T \\
S_1
\end{array}
\]

F2

\[
\begin{array}{c}
\text{in } x_2, z_{12}: T \\
\text{out } y_2, z_{21}: T \\
S_2
\end{array}
\]

F1 \otimes F2

\[
\begin{array}{c}
\text{in } x_1, x_2: T \\
\text{out } y_1, y_2: T \\
\exists z_{12}, z_{21}: S_1 \land S_2
\end{array}
\]
### Specification with Probabilities

**Example:**
**TMC with Probability Restrictions**

<table>
<thead>
<tr>
<th>in</th>
<th>x: T</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>y: T</td>
</tr>
</tbody>
</table>

\[
\forall t \in \mathbb{IN}: \forall m \in T: \\
\mathbb{P}(m#(x \downarrow t) \leq m#(y \downarrow t+\text{delay}+\text{deadline})) \geq 0.8
\]

---

**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

- \( I \triangleright O \)

Data stream of type \( T \)

- \( \text{STREAM}[T] = \{\mathbb{IN}\{0\} \rightarrow T^*\} \)

Valuation of channel set \( C \)

- \( \mathcal{H}[C] = \{C \rightarrow \text{STREAM}[T]\} \)

Interface behaviour for syn. interface \( I \triangleright O \)

- \( [I \triangleright O] = \{\mathcal{H}[I] \rightarrow \text{PD}(\varphi(\mathcal{H}[O])) \} \}

Set of all probability distributions over sets of output histories

---

Probabilistic Behavior Composition

Probabilistic behavior
\[ F: \wp(IH[I]) \to (\wp(IH[O]) \to [0:1]) \]

We write for \( X \subseteq IH[I], Y \subseteq IH[O] \)
\[ F(X)[Y] \quad \text{for the probability} \]
that the output is in \( Y \) provided the input is in \( X \)

Probabilistic Behavior Composition

Given probabilistic behaviors
\[ F_1: \wp(IH[I_1]) \to (\wp(IH[O_1]) \to [0:1]) \]
\[ F_2: \wp(IH[I_1]) \to (\wp(IH[O_2]) \to [0:1]) \]
Where \( O_1 \cap O_2 = \emptyset \); we define
\[ I = I_1 \setminus O_2 \cup I_2 \setminus O_1, \quad O = (O_1 \cup O_2) \setminus Z, \]
\[ Z = (I_1 \cap O_2) \cup (I_2 \cap O_1) \quad \text{shared channels} \]
Composition: define
\[ G: \wp(IH[I_1 \cup I_2]) \to (\wp(IH[O_1 \cup O_2]) \to [0:1]) \]
by
\[ G(X)[Y] = F_1(X|_{I_1})[\{y_1: \exists y \in Y: y|_{O_1} = y_1\}] \times \]
\[ F_2(X|_{I_2})[\{y_2: \exists y \in Y: y|_{O_2} = y_2\}] \]
Assuming probabilistic independence
Probabilistic Behavior Composition

Given $G$ we specify $F = F_1 \otimes F_2$

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

by

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

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.

\[
d: \mathbb{H}[C] \times \mathbb{H}[C] \rightarrow \mathbb{R} \cup \{-\infty, +\infty\}
d(x, x') = \operatorname{glb} \{1/2^t: x \downarrow t = x' \downarrow t\}
\]

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 \textit{field of sets} if

- $\emptyset \in F$,
- 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 \textit{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 \textit{measure} if the following properties holds:

- $\mu(A) \geq 0$ for all $A \in F$
- $\mu(\cup \{A_i : i \in \mathbb{N}\}) = \sum \{\mu(A_i) : i \in \mathbb{N}\}$
  for all pairwise disjoint sets $A_1, A_2, \ldots \in F$,
  i.e. with $A_i \cap A_j = \emptyset$ for all $i \neq j$

(the measure $\mu$ is then called \textit{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!

Reliability Metrics

**Example: Reliability**

From reliability theory:
Reliability distribution

\[ R(t) = P[\text{lifetime of system is at least as long as } t] \]

---

Reliability

Given: set \( Y \subseteq \mathcal{H}[O] \) histories

- output \( y' \in \mathcal{H}[O] \) is correct, iff \( y' \in Y \)
- system with output set \( Y' \subseteq \mathcal{H}[O] \) is correct, iff \( Y' \subseteq Y \)

- Probabilistic system behavior: \( P: \mathcal{P}(\mathcal{H}[O]) \rightarrow [0:1] \)
  Correctness w.r.t \( Y \): \( P[Y] \)

- **Reliability**: expected value \( E_R \) of \( t \) for distribution

\[
R(t) = P[\{y' \in \mathcal{H}[O]: \exists y \in Y: y' \downarrow t = y \downarrow t \}]
\]

is given by

\[
\sum \{ t \cdot R(t): t \in \mathbb{N} \}
\]
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**

- **Quality in End Use**
  - Reliability
  - Security
  - Safety
  - Maintainability
  - Reusability
  - Executability
  - Supportability

- **Quality in Development and Evolution**
  - Usability
  - Separability
  - Learnability
  - Usability
  - Applicability
  - Testability

- **Quality in Business**
  - Performance
  - Portability
  - Adaptability

---

**Auxiliary Quality Attributes**

- Adaptability
- Portability
- Adaptability

---

A novel characterization of system properties and requirements

<table>
<thead>
<tr>
<th>Syntactic Basis</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Logical</td>
</tr>
<tr>
<td>&quot;External&quot;</td>
<td>Functional view</td>
</tr>
<tr>
<td>&quot;Internal&quot;</td>
<td>Architecture view</td>
</tr>
<tr>
<td>&quot;Glass Box&quot;</td>
<td>State view</td>
</tr>
</tbody>
</table>

A novel characterization of functional requirements

<table>
<thead>
<tr>
<th>Interface</th>
<th>Architecture</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional properties</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Functional Suitability</th>
<th>Usability</th>
<th>Reliability</th>
<th>Security</th>
<th>Safety</th>
<th>Performance</th>
<th>Maintainability</th>
<th>Reusability</th>
<th>Releasability</th>
<th>Executability</th>
<th>Supportability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntactic</td>
<td>Logical</td>
<td>Probabilistic</td>
<td>Logical</td>
<td>Probabilistic</td>
<td>Logical</td>
<td>Probabilistic</td>
<td>Logical</td>
<td>Probabilistic</td>
<td>Logical</td>
<td>Probabilistic</td>
</tr>
<tr>
<td>Releasability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Executability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supportability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Rich Interface Specifications

• In a rich interface specification we speak about several views
• Example: Add probability

  Given:
  ◦ logical interface behavior for syn. interface \((I \rightarrow O)\)
    \(\{I[H[I] \rightarrow \varnothing(I[H[O]])}\}\)
  ◦ probabilistic interface behavior for syn. interface \((I \rightarrow O)\)
    \(\{I[H[I] \rightarrow PD[\varnothing(I[H[O]])]\}\}\)
  ◦ interface specification by an interface assertion \(q(x, y)\)
  ◦ specify for each input history \(x = a\) 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 be 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