Model-Based System Engineering for **Cyber-Physical Systems**

Frédéric Mallet

April 26th 2017
0. Systems Engineering Design Flow

requirement elicitation

Expectations: (Functional + Non-Functional)

functional specification refinement

Provisions: Concurrency Dependency

architectural specification refinement

Provisions: Intrinsic Non-Functional Properties (parallelism, time, consumption)

Adequation

Decision:
- Alloc/mapping/assignment/binding
- spatial + temporal (scheduling)

Multi-Domain Assemblies
0. Kairos (EPC Inria/I3S) : Multi-Domain Assembly

Domain 1

***

Domain n

Coordination

Requirements
(Functional + Non-Functional)

(formal) Modeling  Model Execution  (co-) simulation  Controller synthesis  Code vs. Model validation
0. Kairos (EPC Inria/I3S) : Modeling

Domain 1

... 

Domain n

Constraints = pre-conditions

Coordination

Allocation / Mapping

conflicts

System

Placement (spatial)

Expectations = post-conditions
time, power, energy

Requirements
(Functional + Non-Functional)

[GeMoC / BCoOL]
http://gemoc.org

The Clock Constraint Specification Language (CCSL)
0. Kairos (EPC Inria/I3S) : Optimization

Domain 1

***

Domain n

Coordination

Explore possible solution(s)

[timesquare.inria.fr]

System

Placement (spatial)

Schedule (temporal)

Requirements
(Functional + Non-Functional)

Requirements
(Funclional + Non-Functional)
1. Background: Cyber Physical Systems

- Network of Heterogeneous Embedded Systems
  - discrete logical time

- In a Physical (uncertain) Environment
  - stochastic continuous models

- With a feedback loop
  - control theory

Using "standard" notations:

- UML, SysML, MARTE, CCSL
2. Contribution:
MARTE / pCCSL

- The Clock Constraint Specification Language
  - Defined as Part of UML/MARTE
  - Stochastic extension (here) : pCCSL

- Verification
  - UML/MARTE/CCSL => Timed Automata  [SEFM’13]
  - UML/MARTE/pCCSL => Stochastic Hybrid Automata (UPPAAL-SMC)  [FACS’16]
  - Hybrid-AADL => Network of Priced Timed Automata  [TCAD’17]

- Case studies
  - Train Control System (CTCS-3)
  - Energy-aware building with two refinement strategies
3. Logical Time and clocks: what, why, how?

- Logical clocks
  - L. Lamport, synchronize distributed systems
    \textit{Partial order consistent with temporal observation}
  - G. Berry, multiform time in synchronous languages
    \textit{(Logical) Synchrony: two events cannot be told apart}

UML/MARTE: Introduce stereotypes « Clock » and « ClockConstraint »
CCSL: (Declarative) syntax to describe constraints on clocks
3. Logical Time and clocks: 
The Clock Constraint Specification Language

- Clocks are streams of ticks: \( c = (c_i)_{i \in \mathbb{N}} \)
- Two partial orders on Clocks
  - \((C, \preceq)\) **Causality**
    - \(\inf (\text{glb}), \sup (\text{lub})\)
  - \((C, \subseteq)\) **Subclock**
    - \(\text{inter (glb), union (lub)}\)
    - Filters (periodic, k-periodic, slices, delays)
- Synchronizations: **sampled**On
- Libraries: Periodic, Sporadic, Deadline, Jitter...
4. MARTE/pCCSL:

rate

- **Rate**: A ‘⊆’ B ‘rate:’ <nat> ‘/’<nat>
  - A ⊆ B: A can occur any time B occur
  - A every 10 B: A occurs precisely every 10\(^{th}\) occurrence of B
    - leaveHome every 1 day
    - leaveHome every 1 day + 8 hour
  - Rate gives more flexibility
    - goToWork ⊆ leaveHome rate: 5 / 7
In CCSL:

- The probability of a clock to tick depends on each state
- Ex:

\[
\begin{align*}
\{a, b\} & \\
\{a\} & \\
\emptyset & \\
\{a, c\} & \\
\{a\} & \\
\{a, c\} & \\
\end{align*}
\]

\[a \subseteq b\]

\[p_a = 2/3, \quad p_b = 1/3\]

\[a \preceq c\text{ size: 1}\]

\[p_a = ?, \quad p_c = ?\]
4. MARTE/pCCSL: probability

- In pCCSL:
  - The probability of a clock is specified
  - But it must be consistent with rates and relations
    - See papers for rules to check consistency
  - Ex: Clock haveMeeting probability 0.6
- In CCSL: precedence = unbounded FIFO
  - $A \preceq B$
  - $A \preceq B$ size: $n$

- In pCCSL:
  - A distribution to characterize the size of the FIFO
  - Ex: $\text{arrive} \preceq \text{leave}$ size: follows distribution $\rho$

![Diagram showing unbounded FIFO and distribution](attachment:image.png)
In CCSL:
- \( B := A \text{ delayFor } n \)
- \( B := A \text{ delayFor } n \text{ on } C \)
- \( B := A \text{ delayFor } n \text{ on } ms \)

In pCCSL:
- A distribution to characterize the delay
- Ex: \( B := A \text{ delayFor } \text{normal}(\mu, \sigma^2) \)
5. Hybrid-AADL:
AADL

- AADL from SAE

application
binding
architecture
5. Hybrid-AADL: AADL and CCSL

- AADL from SAE

5. Hybrid-AADL: CTCS-3

- Movement Authority (MA) Control in Chinese Train Control System Level 3 (CTCS-3)

Uncertain Communication delays

Radio Block Center

uncertain friction
5. Hybrid-AADL:
AADL + BLESS + Hybrid + Uncertainty

Hybrid Annex

BLESS Annex

AADL Design

Variation Information
(Acceleration, Delay, etc.)

Performance Requirement
(Time, Success Ratio, etc.)

Pr[<=T1](<>Train.tv<>0 && Train.ts<EOA )

5. Hybrid-AADL:  
AADL + BLESS + Hybrid + Uncertainty

• BLESS Annex
  - Behavioral Annex for AADL, rely on Priced Timed Automata


thread implementation Controller.impl
annex BLESS{**
variables
e : CTCS_Types::EOA;
iMA : CTCS_Types::MovementAuthority:=null;
states
READY : initial state;
GMA : complete state;
CMA : state;
RETRY : state;
MFR : complete state;
transitions
T0_go: READY -[]-> GMA {r!};
T1_MA_Check: GMA-[on dispatch]->CMA{m?(iMA);ea?(e)};
T2_MA_Ok: CMA-[not (iMA=null)]-> MFR{ };
T3_MA_NotOk: CMA-[iMA=null]-> RETRY { };
T4_MA_Retry: RETRY -[]-> GMA {r!};
**};

annex Uncertainty {**
variables
time t_delay applied to Controller.r
distributions
t_delay = Normal(0.2,0.07)
**}
5. Hybrid-AADL:
AADL + BLESS + Hybrid + Uncertainty

abstract Train
features
  ts: out data port CTCS_Types::Position;
  tv: out data port CTCS_Types::Velocity;
  ta: in data port CTCS_Types::Acceleration;
end Train;

abstract implementation Train.impl
annex Uncertainty {**
  variables
  time v_delay applied to Train.ts
  -- modeling connection delay
  static price v_fr applied to Train.fr
  -- modeling track friction
  distributions
  v_delay = Normal(0.15,0.04)
  v_fr = Normal(-0.1,0.05)
  queries
  p1 = Train.v<=0 && Train.s<EOA
    && Train.s>0 under <=300
  p2 = Train.s >= 4000 under <=200
**};

• Hybrid CSP
annex hybrid {**
  variables
  s : CTCS_Types::Position -- train position
  v : CTCS_Types::Velocity -- train velocity
  a : CTCS_Types::Acceleration -- train acceleration
  t : CTCS_Types::Time -- system time
  fr : CTCS_Types::Deceleration -- track friction
  behavior
  Train ::= 'DT 1 s=v' & 'DT 1 v=a+fr' & 'DT 1 t=1'
    [> ts!(s), tv!(v),ta?(a)]> Continue
    Continue ::= skip
    RunningTrain ::= s:=0 & v:=0 & a:=0 & REPEAT(Train)
**};
end Train.impl;
6. Analysis

• Query :
  - The train must stop safely before the EOA

\[ \Pr[\leq 300](\left< \text{Train}.v \leq 0 \right> \&\& \text{Train}.s < 6000 \&\& \text{Train}.s > 0) \]

<table>
<thead>
<tr>
<th>Causes</th>
<th>Constructs</th>
<th>Variations</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Delay</td>
<td><strong>Controller.r</strong></td>
<td>( N(0.1, 0.03^2) )</td>
<td>Seconds</td>
</tr>
<tr>
<td></td>
<td><strong>RBC.m</strong></td>
<td>( N(0.1, 0.03^2) )</td>
<td>Seconds</td>
</tr>
<tr>
<td></td>
<td><strong>RBC.ea</strong></td>
<td>( N(0.1, 0.03^2) )</td>
<td>Seconds</td>
</tr>
<tr>
<td></td>
<td><strong>Train.tv</strong></td>
<td>( N(0.15, 0.04^2) )</td>
<td>Seconds</td>
</tr>
<tr>
<td></td>
<td><strong>Train.ts</strong></td>
<td>( N(0.15, 0.04^2) )</td>
<td>Seconds</td>
</tr>
<tr>
<td></td>
<td><strong>Controller.ca</strong></td>
<td>( N(0.17, 0.04^2) )</td>
<td>Seconds</td>
</tr>
<tr>
<td>Parameter</td>
<td><strong>Train.fr</strong></td>
<td>( N(-0.1, 0.05^2) )</td>
<td>MPSS*</td>
</tr>
<tr>
<td>Execution Time</td>
<td><strong>RBC.T0</strong></td>
<td>( N(0.1, 0.03^2) )</td>
<td>Seconds</td>
</tr>
<tr>
<td></td>
<td><strong>Controller.T5</strong></td>
<td>( N(0.2, 0.07^2) )</td>
<td>Seconds</td>
</tr>
</tbody>
</table>

*MPSS indicates *Meter Per Second Squared.*
6. Analysis

• Query:
  - The train must stop safely before the EOA

Pr[<= 300](<> Train.v <= 0 && Train.s < 6000 && Train.s > 0)