PROGRESS - Predictable Component-based Development of Embedded Systems

Paul Pettersson

PROGRESS

A national Swedish Strategic Research Centre

MÄLARDALEN UNIVERSITY

MRTC
MÄLARDALEN REAL-TIME RESEARCH CENTRE

Swedish Foundation for Strategic Research

Mälardalen University

- Campuses in Eskilstuna & Västerås
- 110 km west of Stockholm
- Top ranking in quality work evaluation by Swedish National Agency for Higher Education
- Six prioritized research areas
  - Embedded systems
  - Innovation and Product Development
  - Environment, Energy and Resource Optimization
  - Sustainable Development, Working Life, and Management
  - Welfare and Health
  - Didactics and Intercultural Communication

- In numbers:
  - Staff 1000
  - Professors 60
  - Students 13000
  - 60 programs and 750 courses
  - Turnover 75MEuro

PROGRESS

MRTC & Progress

- Mälardalen Real-Time Research Centre:
  - Hosted by Mälardalen univ.
  - Real-Time Embedded Systems
  - Industrial Software Engineering
- Research groups:

- Progress:
  - National Strategic Centre for Embedded Software Development hosted by MRTC
  - Lead by Hans Hansson
  - 2006-2011

Real-Time Modeling & Analysis	Program Analysis	Dependable Software Engineering	Industrial Software Engineering	Communication Performance & Analysis	Real-Time Systems Design	Embedded Systems SW Engineering
Paul Pettersson	Björn Lisper	Sasi P./Kristina L.	Ivica Crnkovic	Mats Björkman	Hans H./Mikael S.	Christer Norström

PROGRESS

Progress

Focus on component-based development of real-time embedded systems

Hypothesis:

building embedded software (and systems) from reusable components

- complexity,
- integration, and
- quality assurance

can be handled in a more cost efficient and scalable way

Overview

- MRTC, and Progress
- Procom – Progress component model
  - ProSave, ProSys,
  - attribute framework
- PrIDE – Progress IDE
  - editing
  - simulation
  - verification

- UPPAAL PORT
  - analysis model for real-time components
  - verification by partial-order reduction
- REMES
  - modelling of embedded resources
  - analysis: feasibility, optimality, trade-off
- Conclusion

Research groups

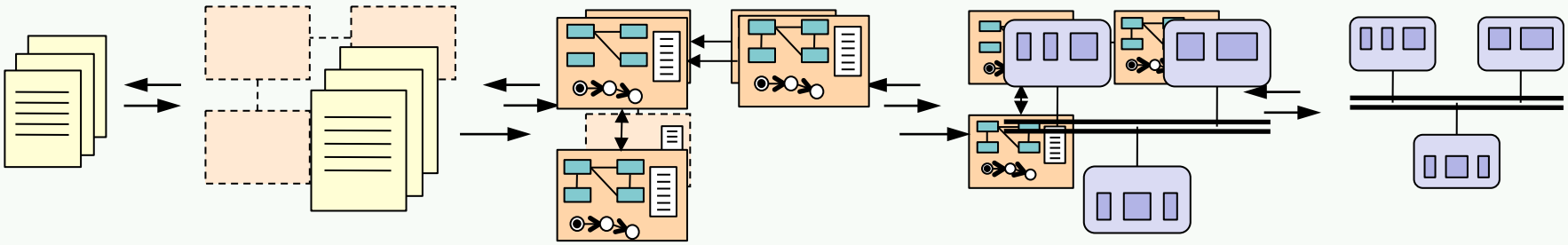
PROCOM

A component model for real-time embedded systems

PROGRESS

ProCom – Key aspects

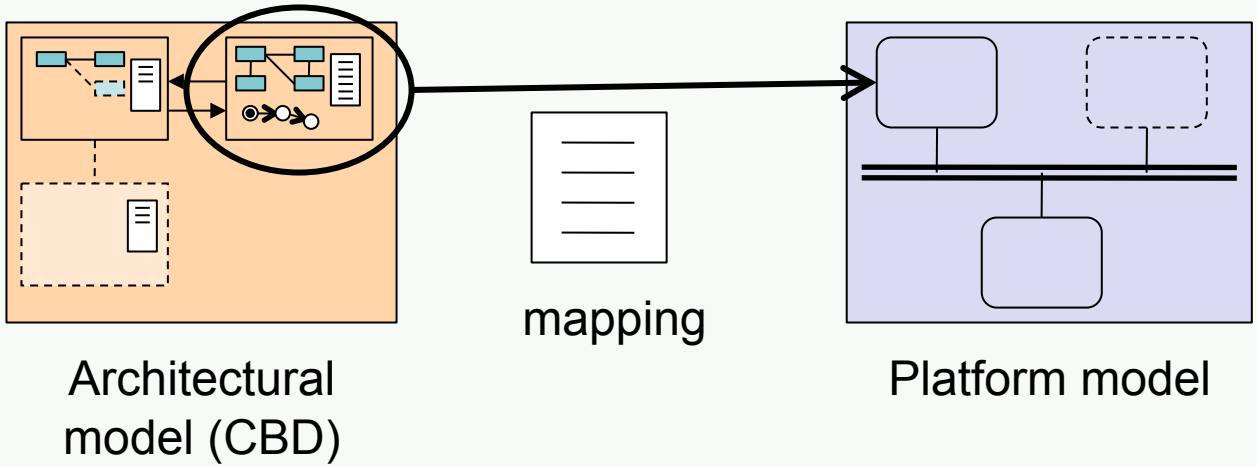
- Design-time components.



- Rich component concept.
  - Including models of timing and resources, analysis results, documentation, source code, etc.
  - Support for reuse components.
  - Components of different maturity should be allowed to co-exist.

ProCom – Key aspects

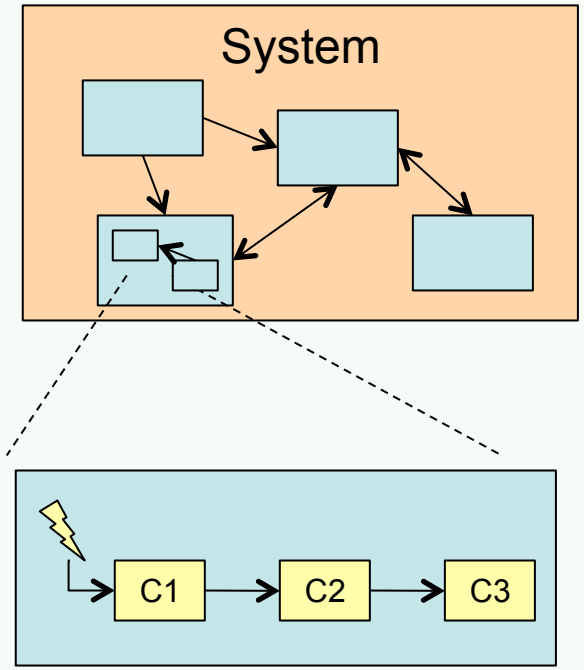
- Components abstracted from physical deployment.



- Different concerns depending on granularity.
  - Distribution, communication, analysis, etc.

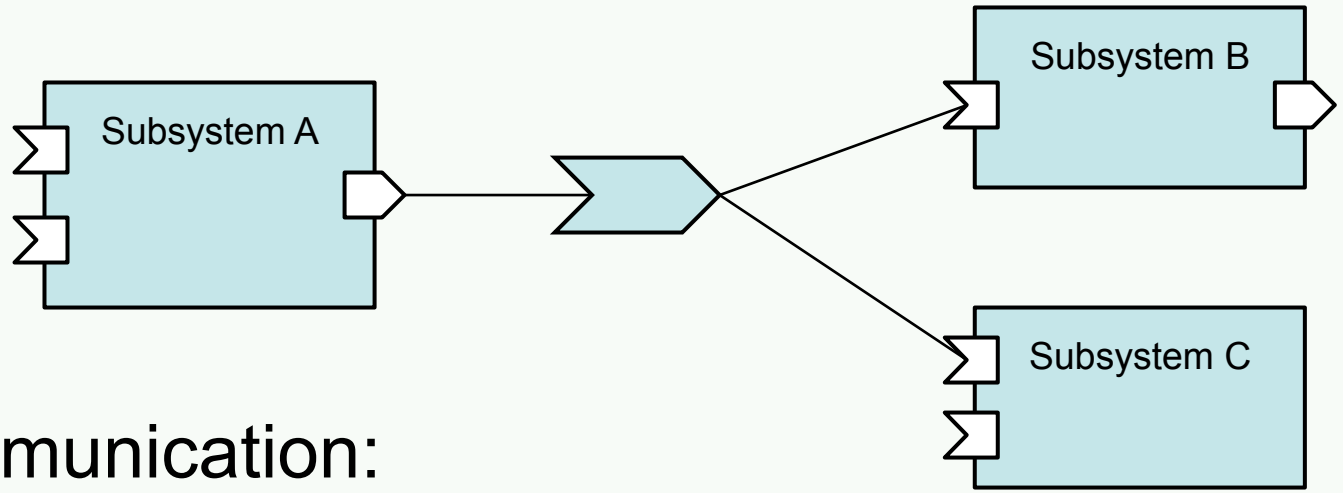
ProCom – a multi-layered component model

- ProSys model (upper layers)
  - Systems and subsystem components
  - Active, typically distributed
  - Asynchronous message passing
- ProSave model (lower layers)
  - Function block, primitive
  - Passive, non-distributed
  - Explicit transfer of data and control
- Connection between the layers
  - A subsystem can internally be modelled by ProSave.



ProSys – the upper layers

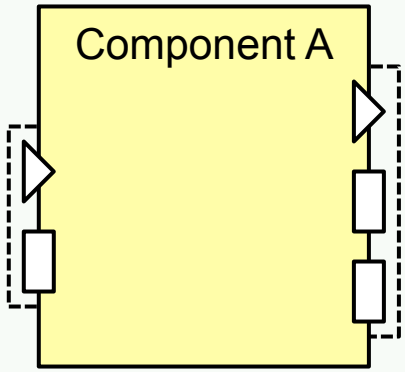
- Components (subsystems):
  - Active, possibly distributed.
  - Interact through message ports.



- Communication:
  - Asynchronous messages.
  - Explicit message channels.

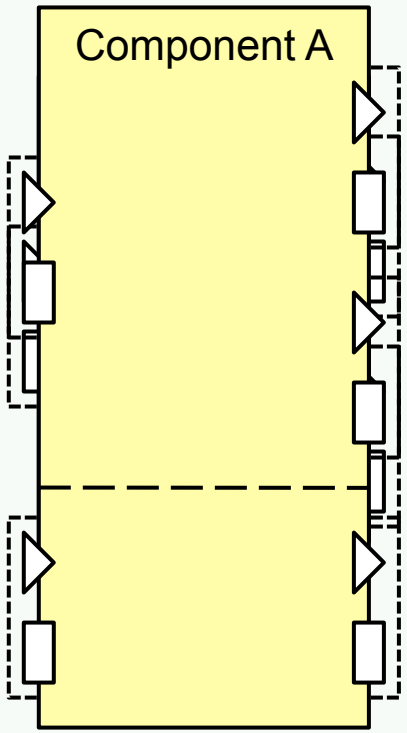
ProSave – the lower levels

- Passive components
  - Similar to task or function block
- Interact through input- and output ports.
  - Data ports
  - Trigger ports
- read-execute-write semantics:
  1. Initially passive, receiving input data.
  2. When triggered, read input data and turn active
  3. Executing internal component behavour
  4. Write output; goto 1.



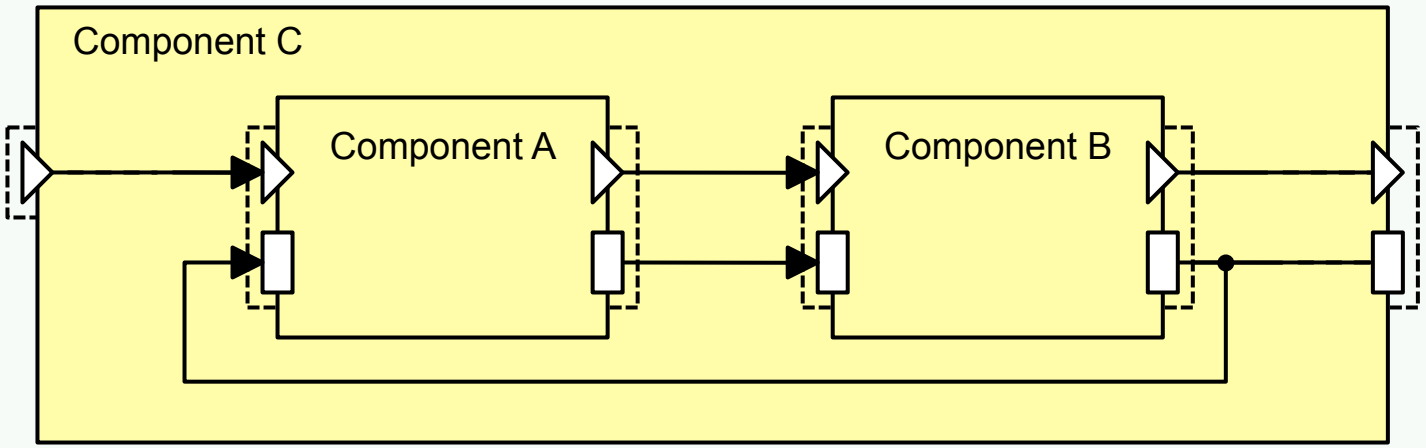
ProSave – the lower level

- More complex components can have:
  - Multiple output groups:
    - Output can be produced at different points in time.
    - Each group written once per activation.
  - Multiple input groups (services):
    - Services can share state.
    - Individual control flows



ProSave – the lower level

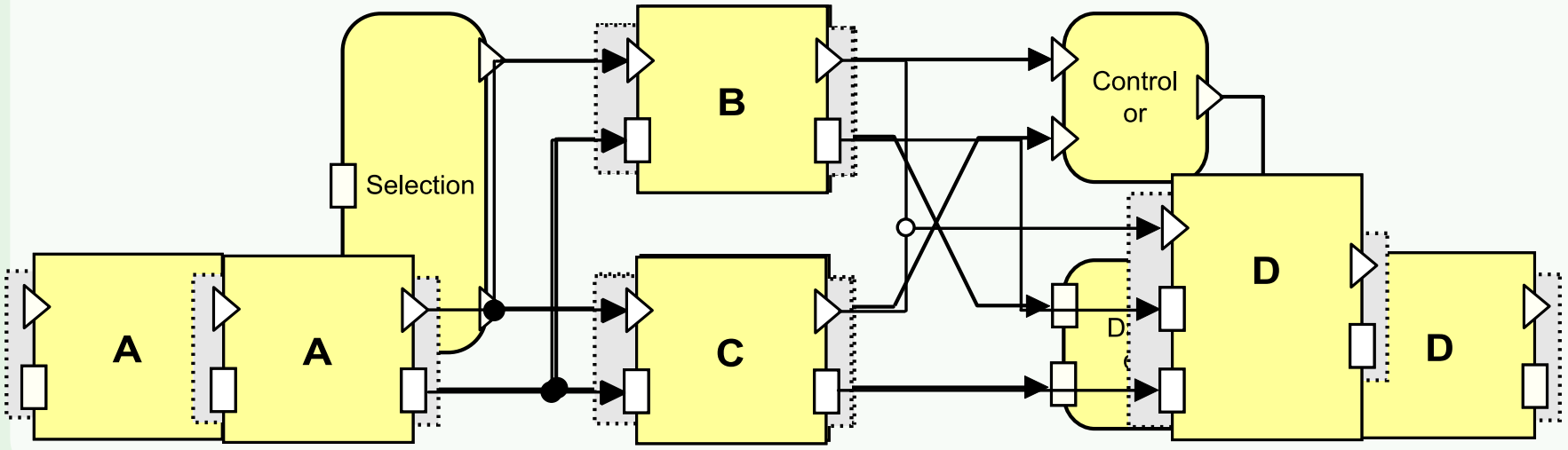
- Separated data- and control flow



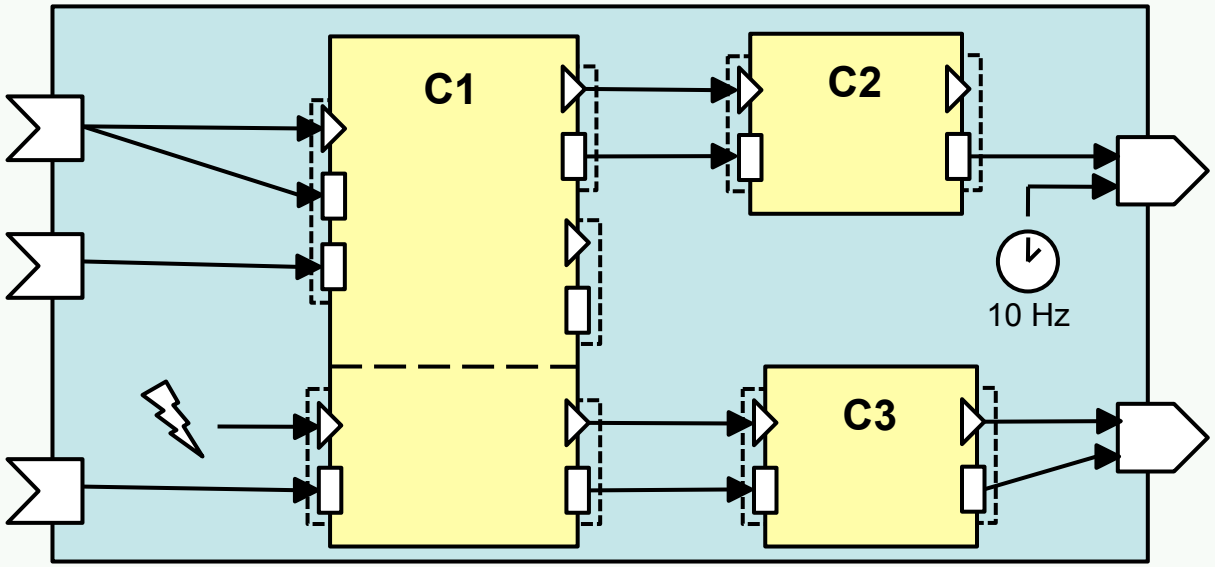
- Hierarchical nesting
  - Primitive components
  - Composite components

ProSave – the lower level

- Connectors for more elaborate control:
  - Control fork
  - Control join
  - Control selection
    - Control or
  - Data fork
  - Data or

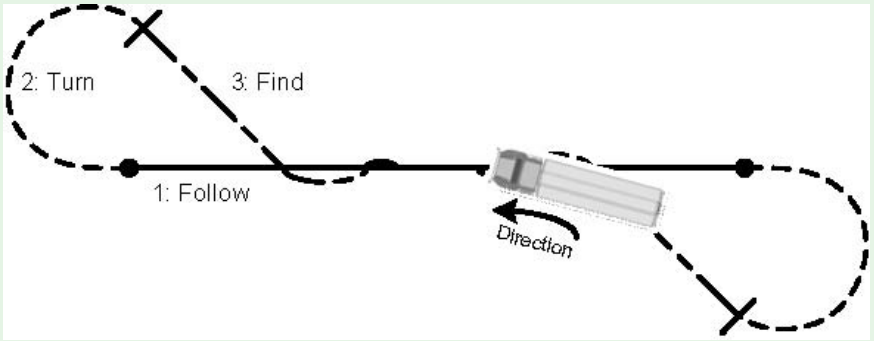


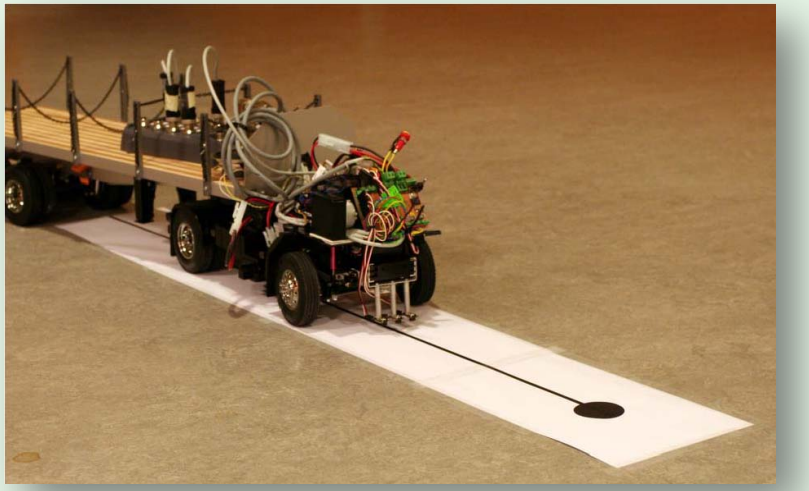
Modelling a ProSys subsystem in ProSave

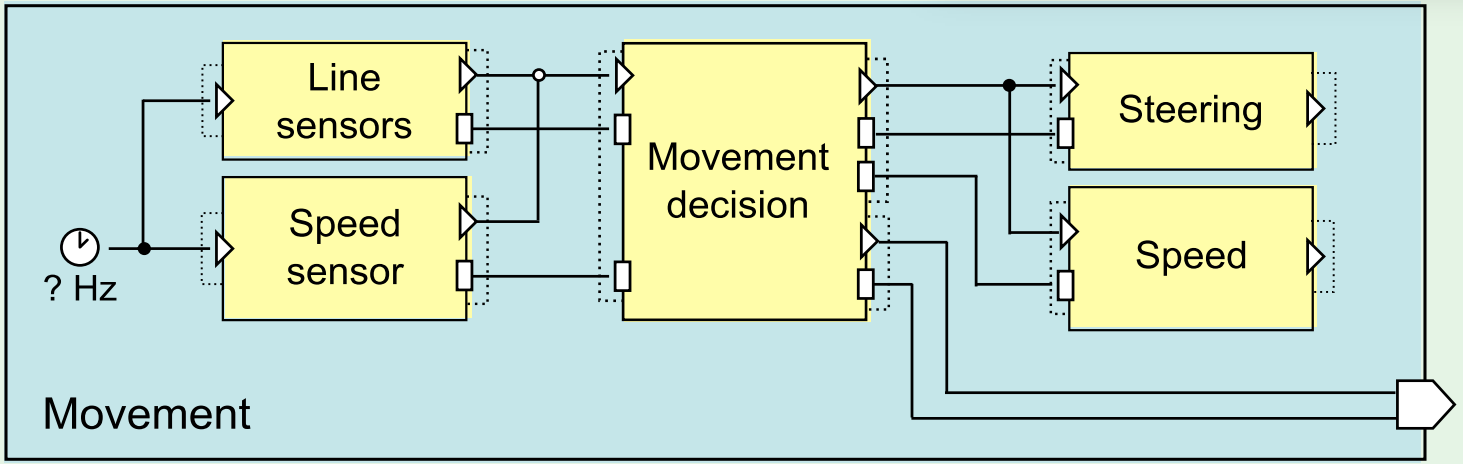


- Message ports ↔ trigger and data
- Clocks and events

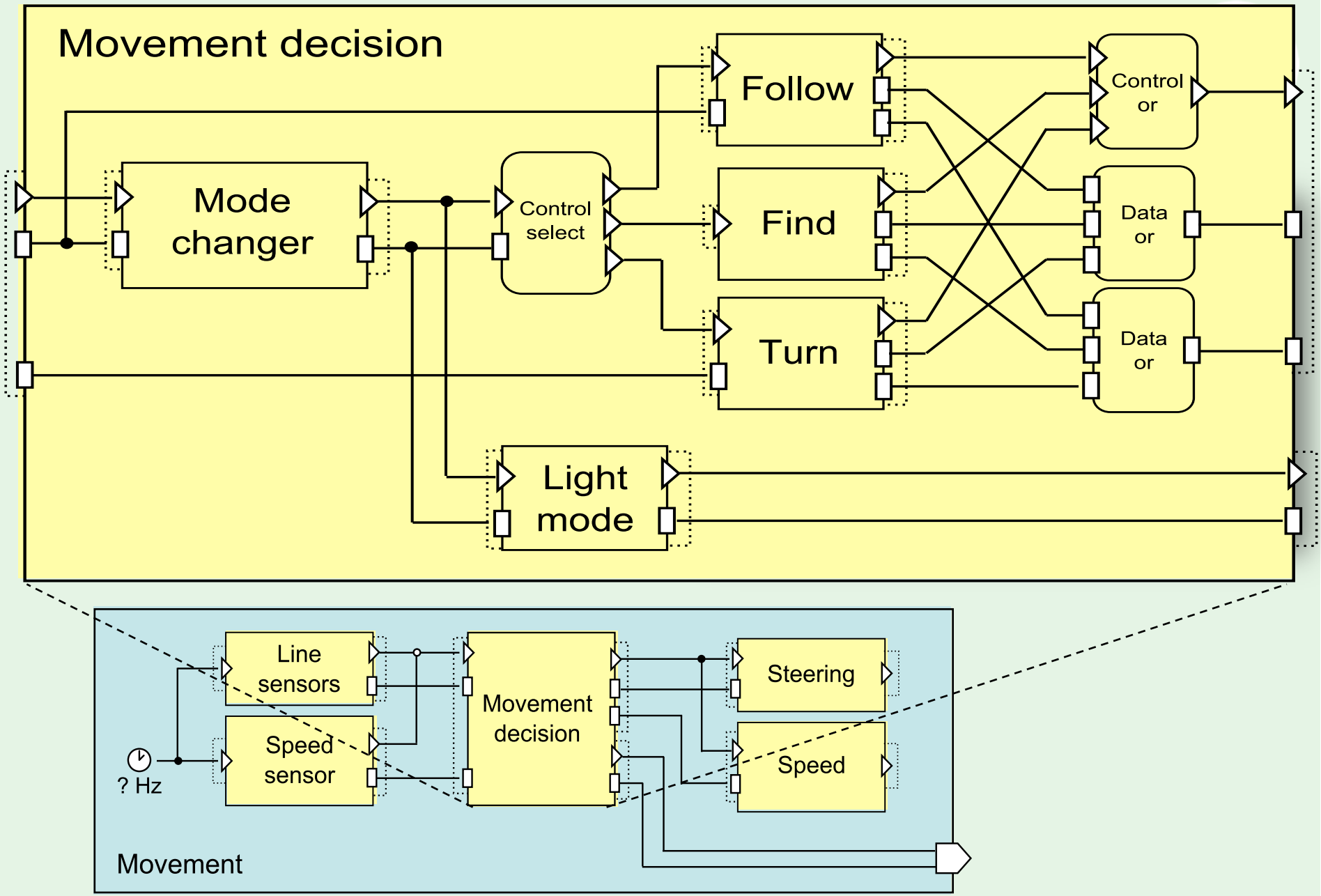
Truck Example





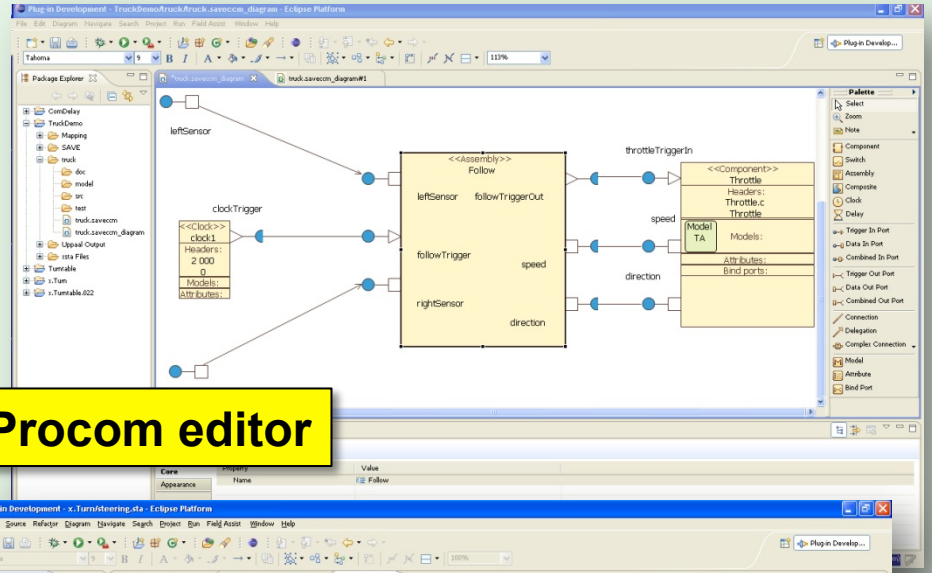


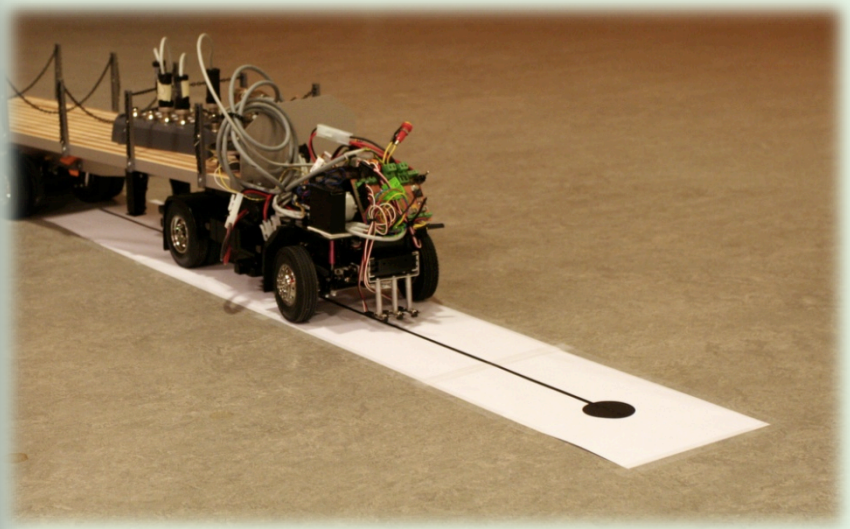
PROGRESS

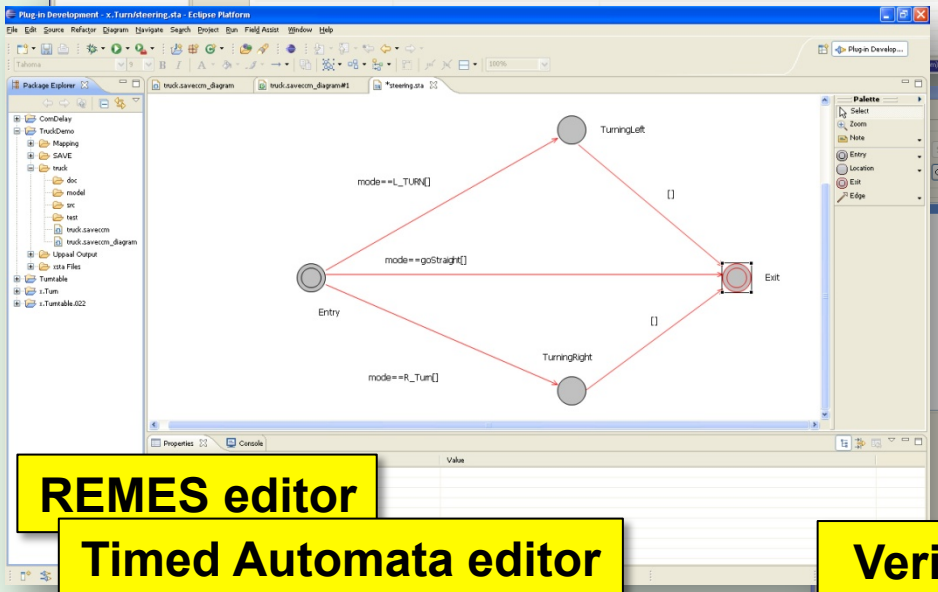

Movement decision
Mode changer
Control select
Follow
Find
Turn
Control or
Data or
Data or
Light mode
Line sensors
Speed sensor
? Hz
Movement decision
Steering
Speed
Movement

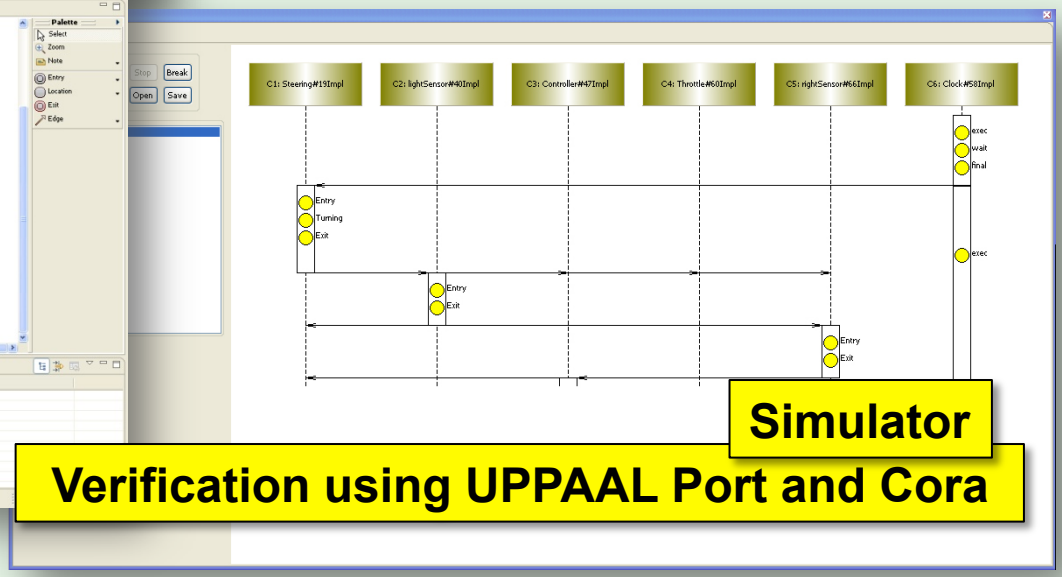
PROGRESS

Progress IDE - PrIDE









PROGRESS

ATTRIBUTE FRAMEWORK

Integration of Extra-Functional Properties in PROCOM

PROGRESS

Attribute Framework

- Integration of extra-functional properties in the component model
- Reuse and composability of EFP
  - often poor support for this in CBD
- Attribute Framework:
  - Manage EFP in a systematic way
  - Store various analysis results (for reuse)
  - Generic for may different type of EFPs

Huge List of Properties…

Nb of reuse

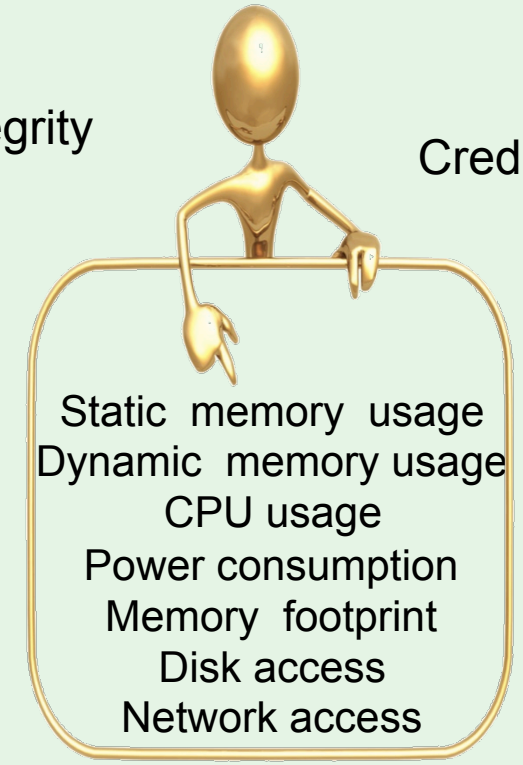
Integrity

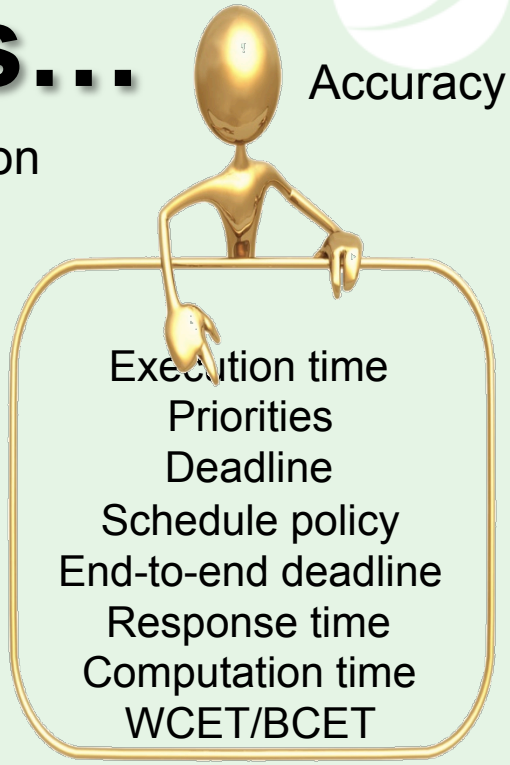
Precision

Accuracy

Throughput

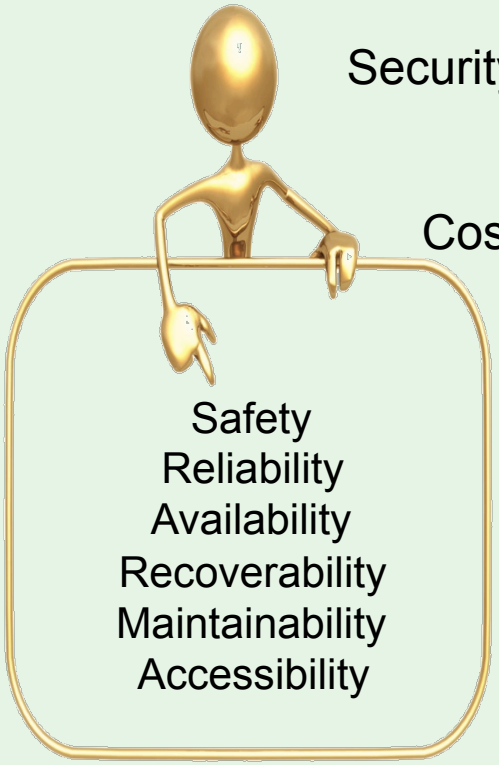
Credibility





Security

Cost



LoC

Confidentiality

Evolvability

Extensibility

Value range

Nb of tests

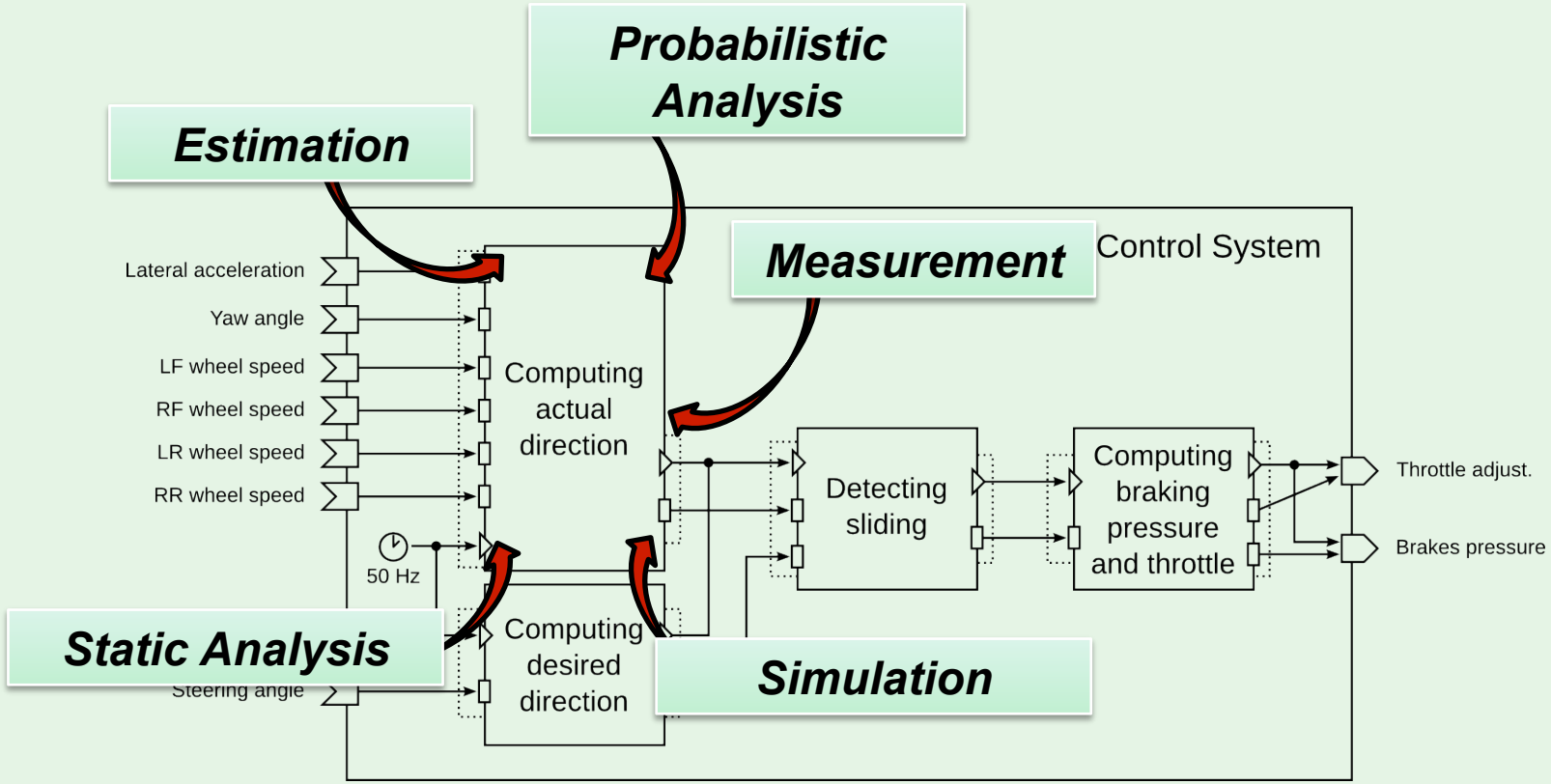
Confidentiality

...

Compliance to standard

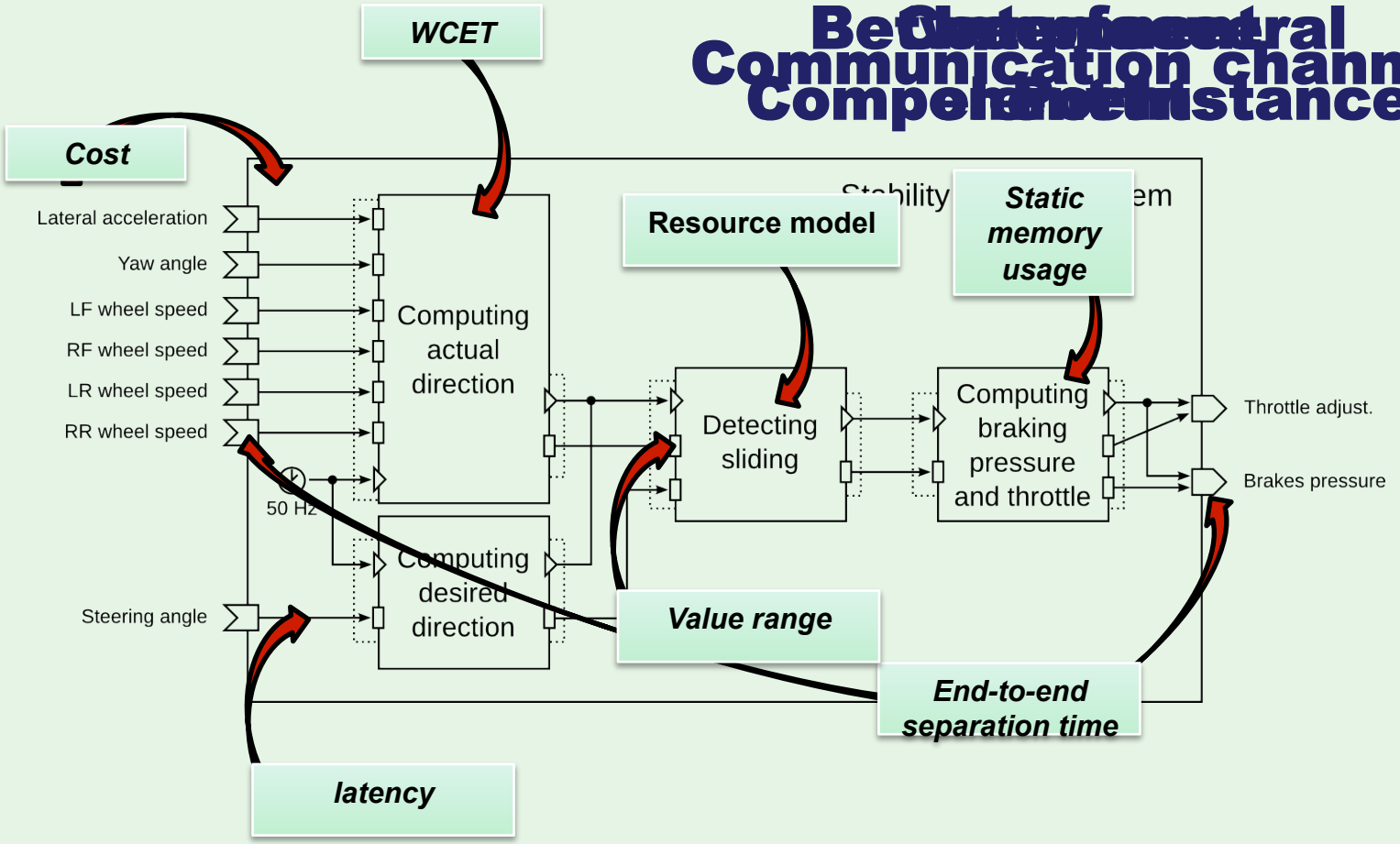
PROGRESS

Origin differs…



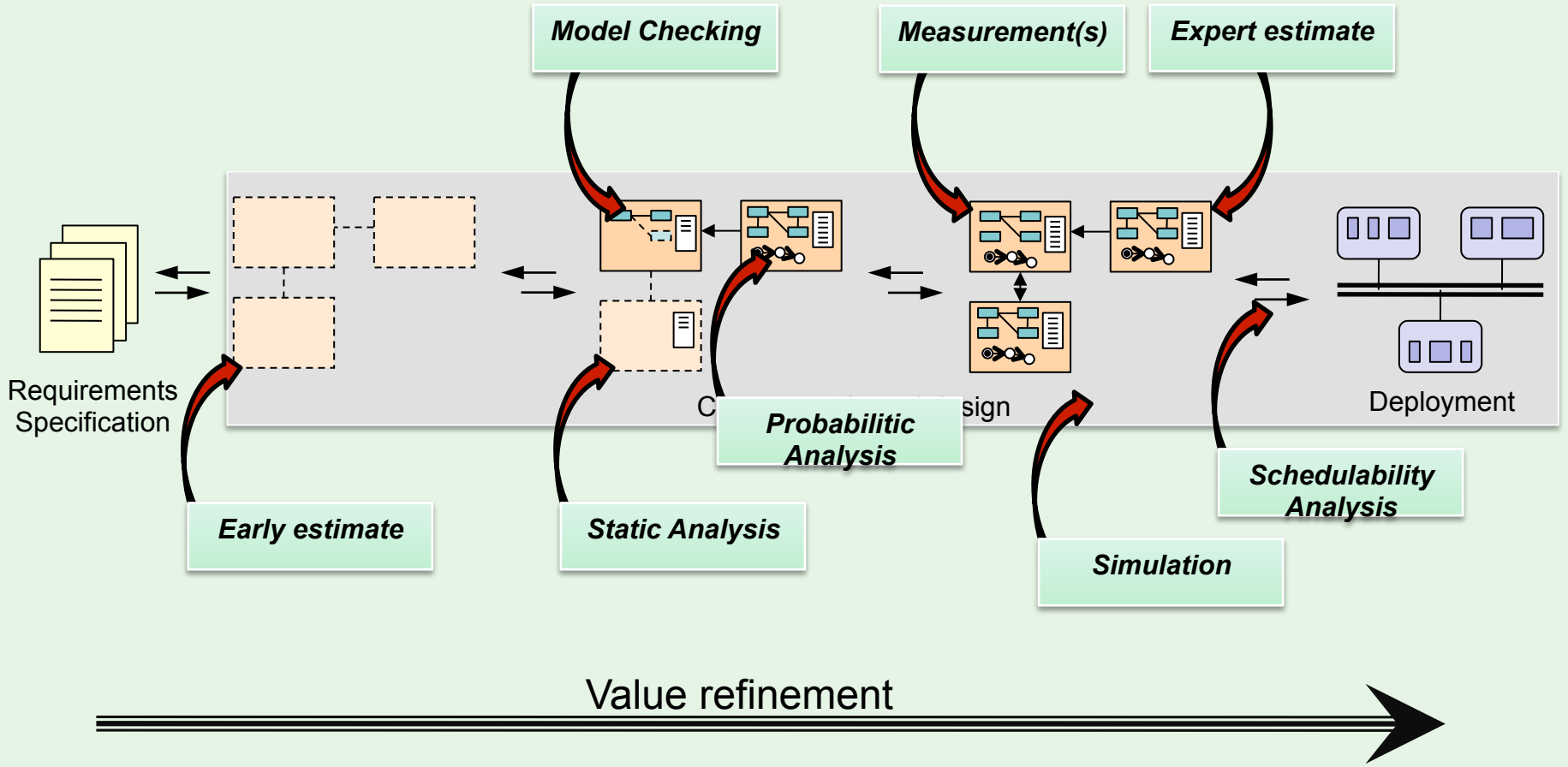
PROGRESS

Relation to different component model entities differs…

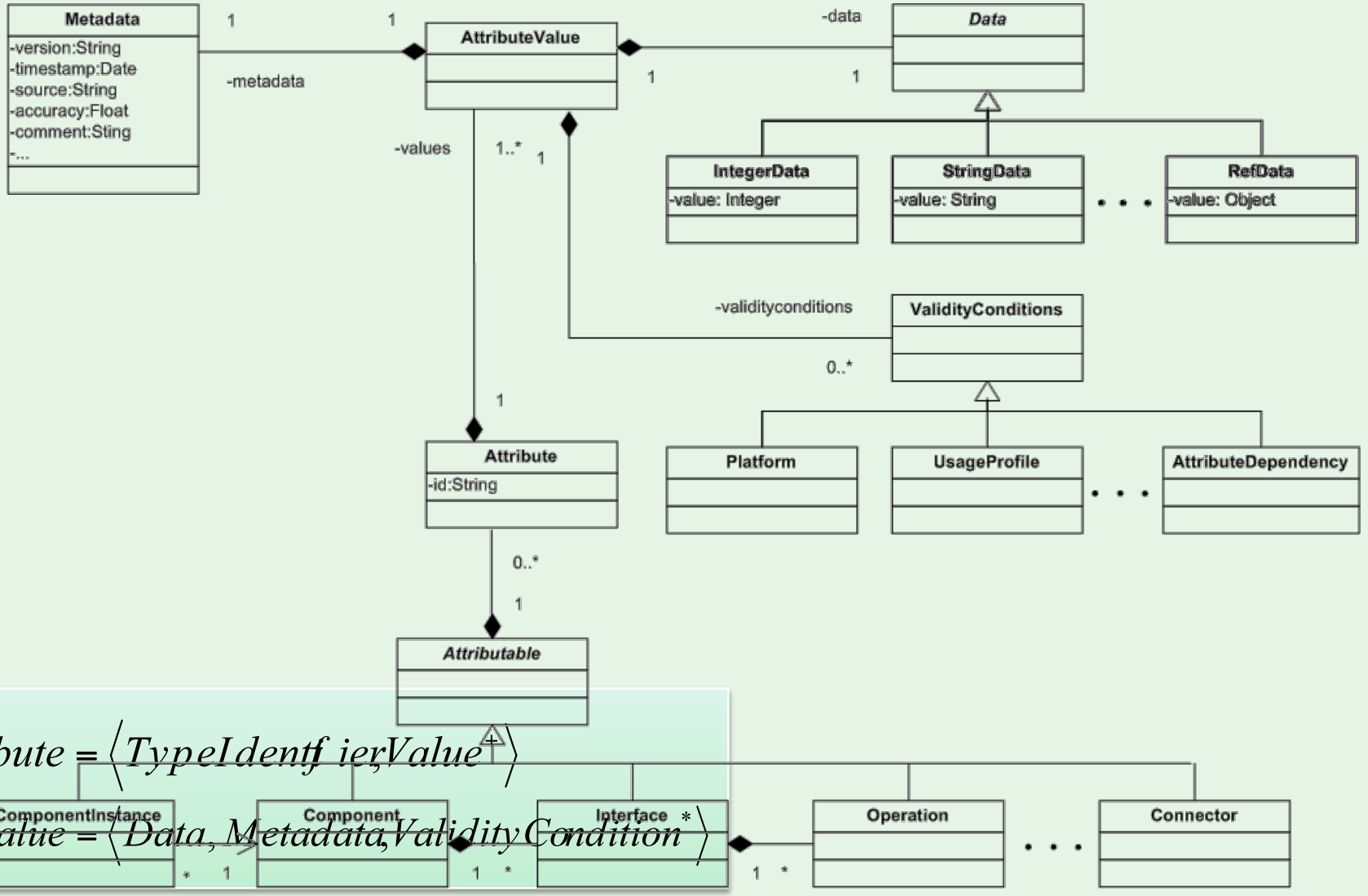


PROGRESS

Development phase differs…



Attribute Structure

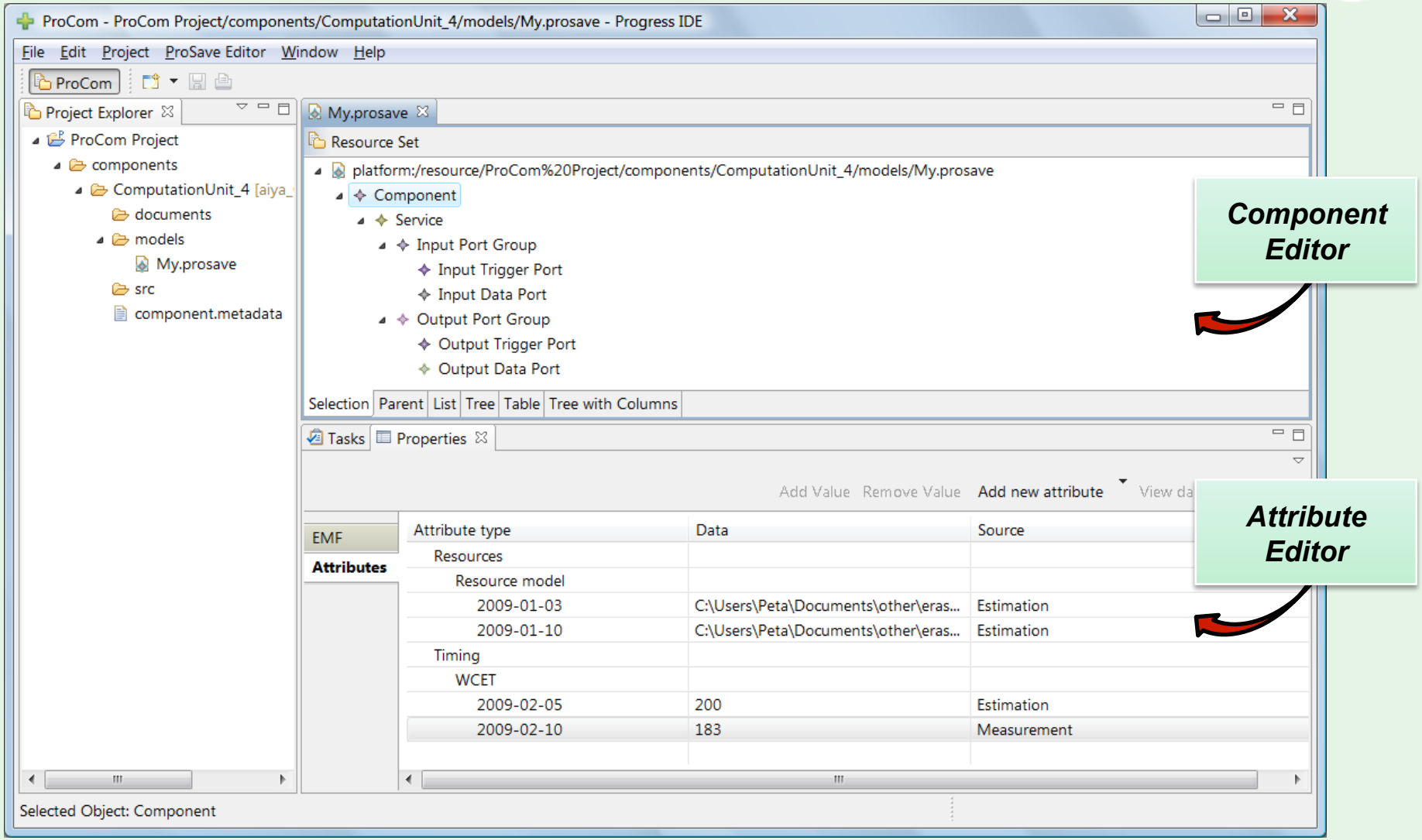


$$Attribute = \langle TypeIdentifier, Value \rangle$$

$$Value = \langle Data, Metadata, ValidityCondition \rangle$$

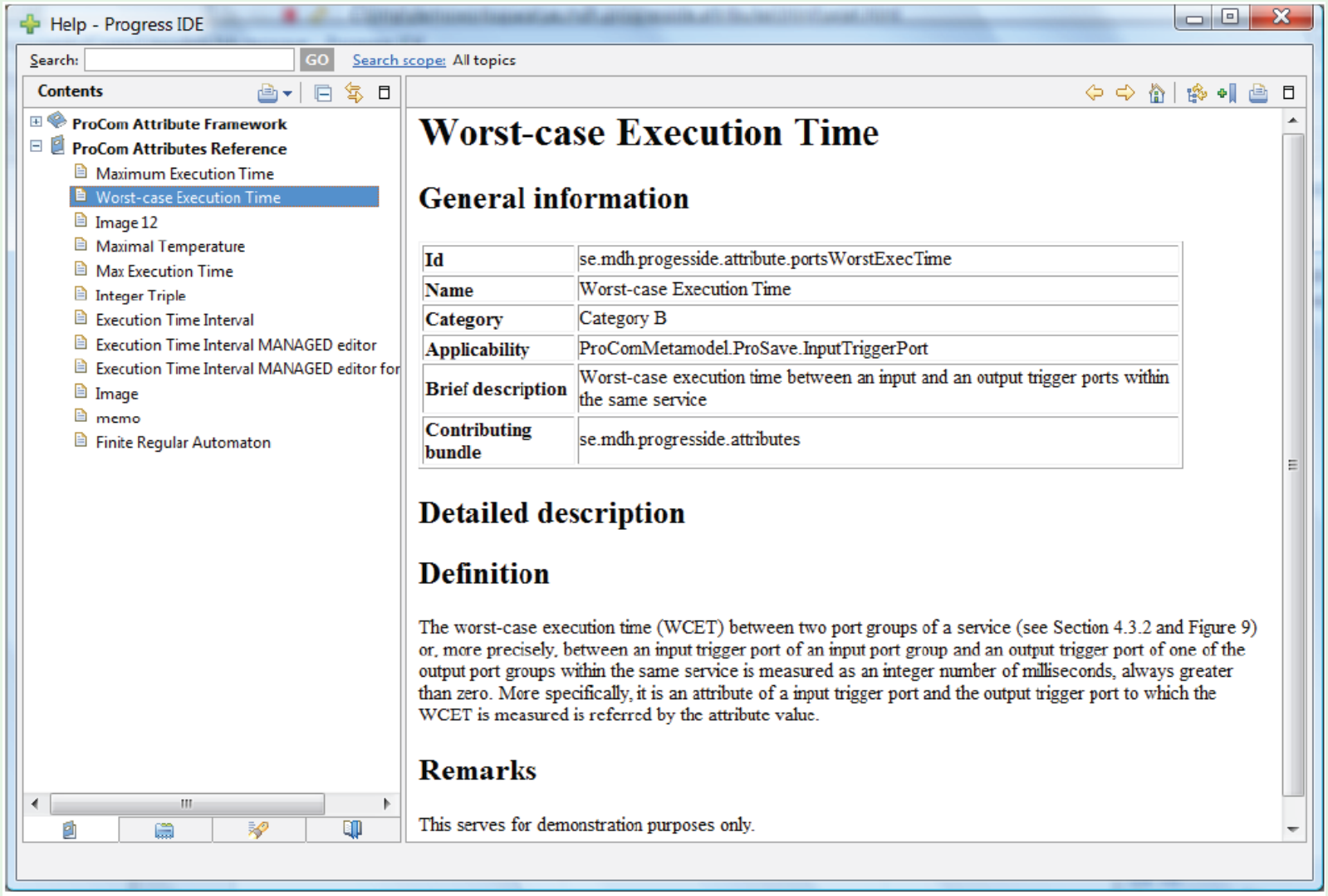
PROGRESS

PROGRESS IDE



PROGRESS

Attribute Documentation



PROGRESS

UPPAAL PORT

Modelling and verification of Real-time components

PROGRESS

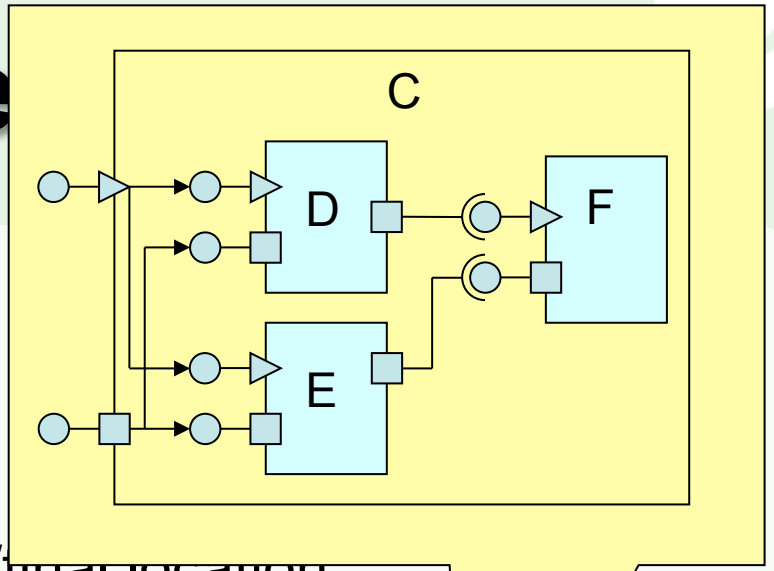
Modelling and Verification

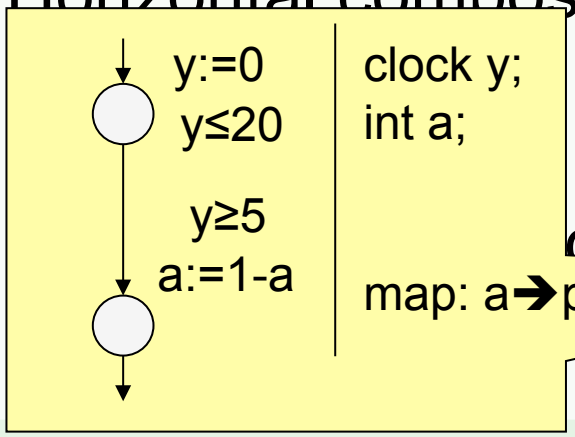
- Establish correctness at design time of models
- Functional and timing properties of components
  - Model of functional and real-time behaviour
  - Verification of safety liveness properties
- Difficulties/complexity:
  - hierarchical model, communication structure, functional behaviour, timing

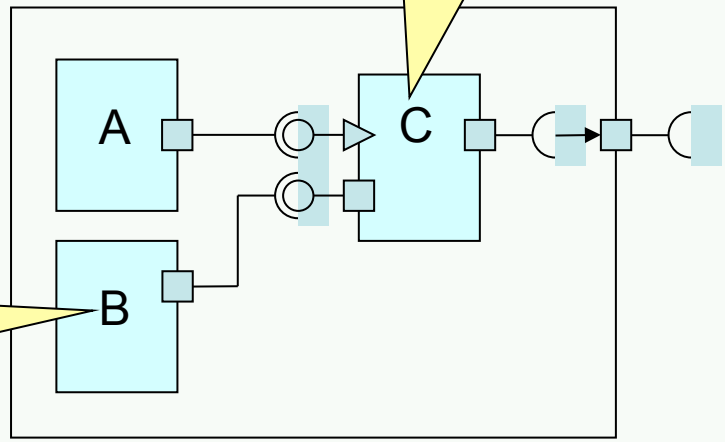
- Common approach:
  - Perform analysis of component model on equivalent ”flat” model
  - Performance problems
- Our approach:
  - Perform analysis directly on hierarchical component model
  - Apply reduction techniques to exploit component model structure
  - Partial-order reduction

ProSave + Time

- ProSave
  - Ports, read-execute-write
  - Function and timing
    - timed automaton with start/final location
    - ports mapped to data variables
    - analysis model
  - Horizontal composition
  -







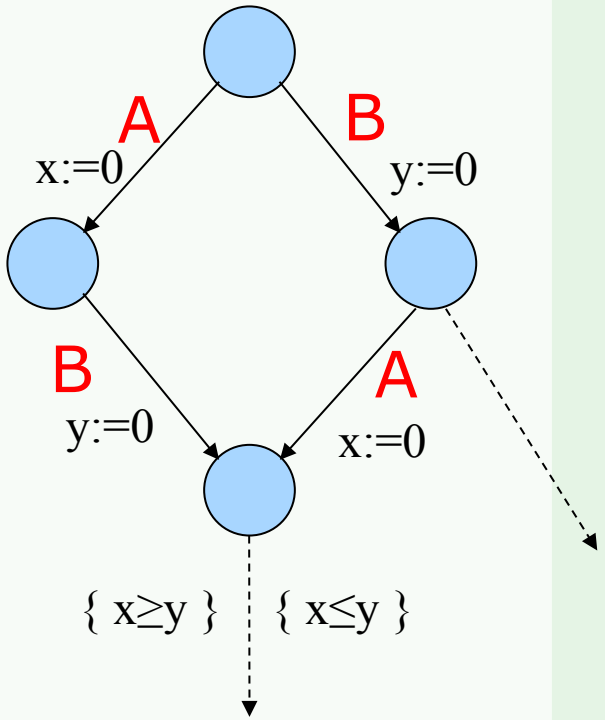
PROGRESS

PORT for Timed Systems

- Attempted before:
  - Bengtsson et. al. (’98), Minea (’99)
    - Local time semantics
    - Allows time to progress independently in parallel automata
  - Niebert et al (event zones, ’04-), Maler et al (interleavings ‘06)
- Our approach
  - Based on local time semantics
  - Structured model → more information
  - Components execute independently
    - read → execute → write semantics

Partial Order Reduction

- interleaving semantics:
  - A and B parallel actions: explore AB *and* BA
- reduce interleaving
  - A and B are independent
  - commutativity, enabledness, …
- explore representative traces
  - maintains correctness
  - BA representative of AB
- timing adds problems

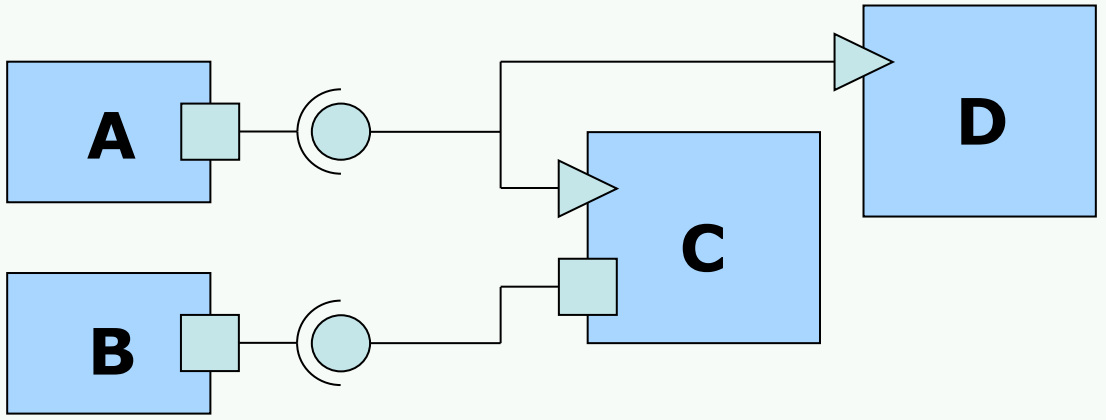


Independence in Components

- **active** component (in a transition):
  - delay A = { A }
  - internal A = { A }
  - write A = { A } ∪ { K | *connection from* A *to* K) }
- **independent:** transitions α and β if
  - no component active in both:
    active(α) ∩ active(β) = ∅
- example:

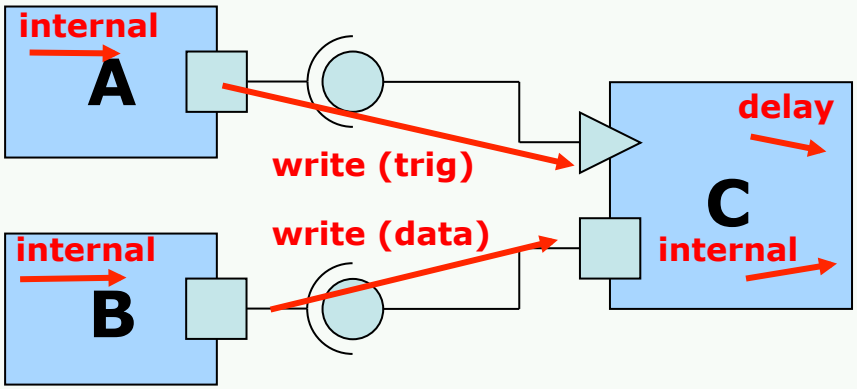
Dependent:

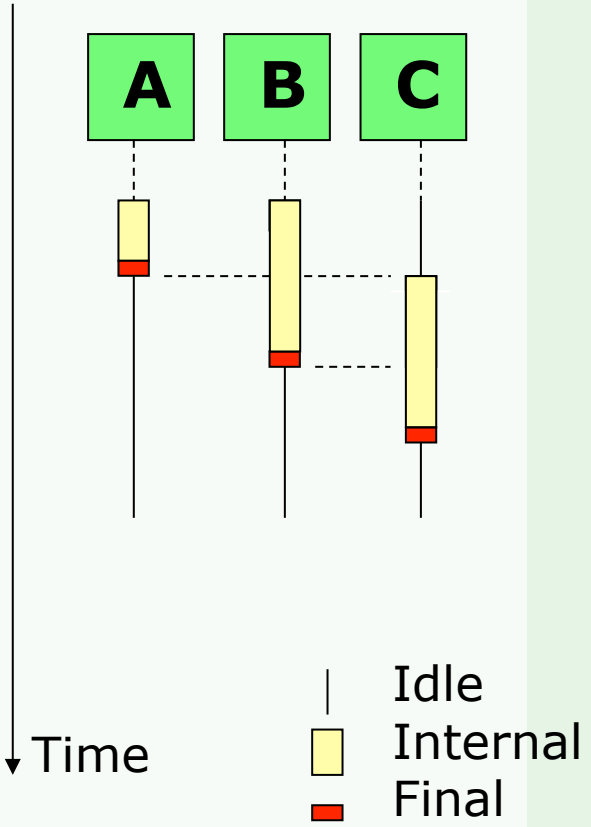
internal A – write A
internal C – write A
internal D – write A
write A – write B
internal B – write B
internal C – write B



Local Time Semantics

- local clocks c^A, c^B, c^C
- local transitions: delay K, internal K,
  - delay K advances clock c^K
- write transition of A
  - synchronize reference clocks c^A and c^K if A is connected to K



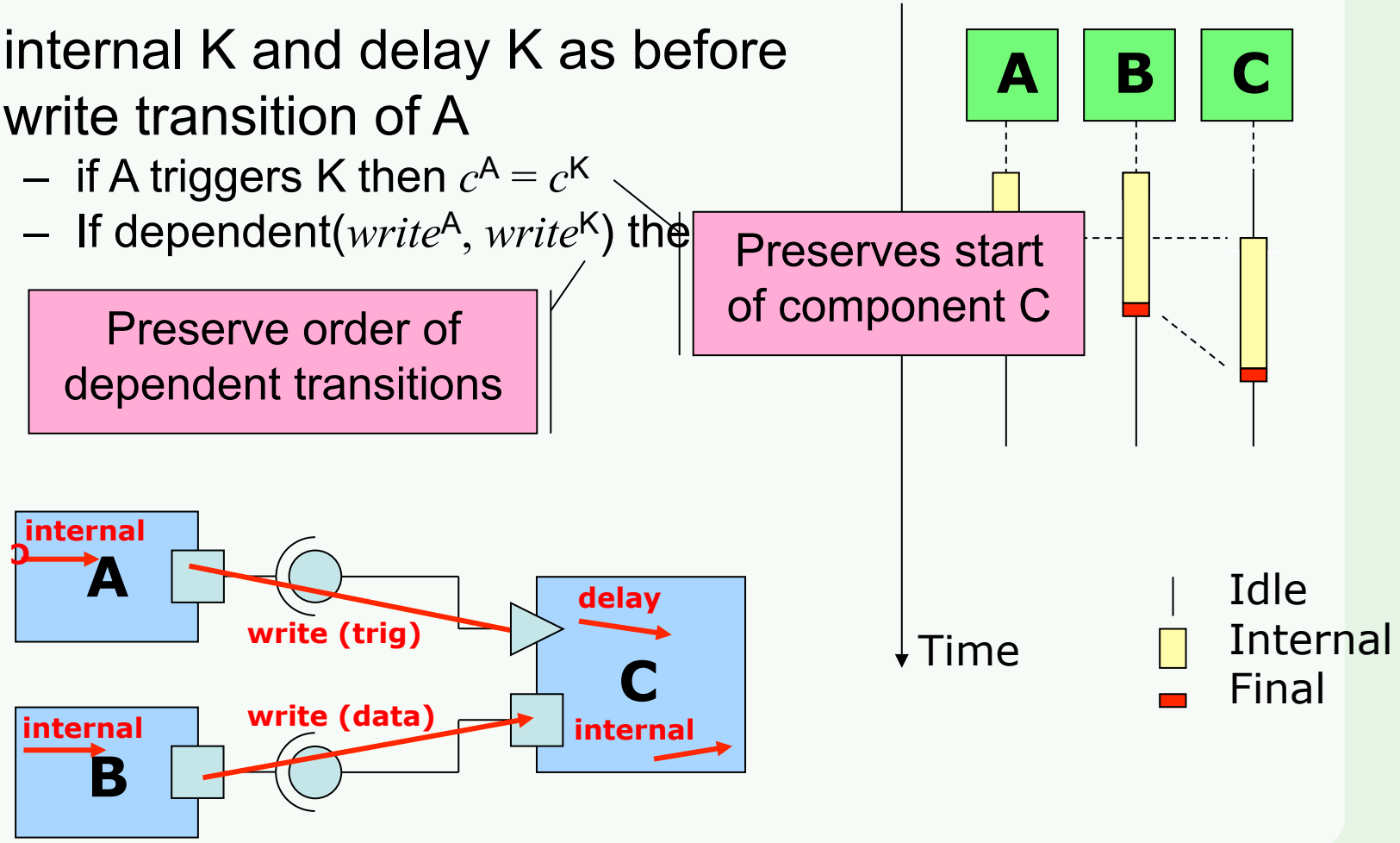


Further Relaxed Synchronization

- internal K and delay K as before
- write transition of A
  - if A triggers K then $c^A = c^K$
  - If dependent($write^A$, $write^K$) the

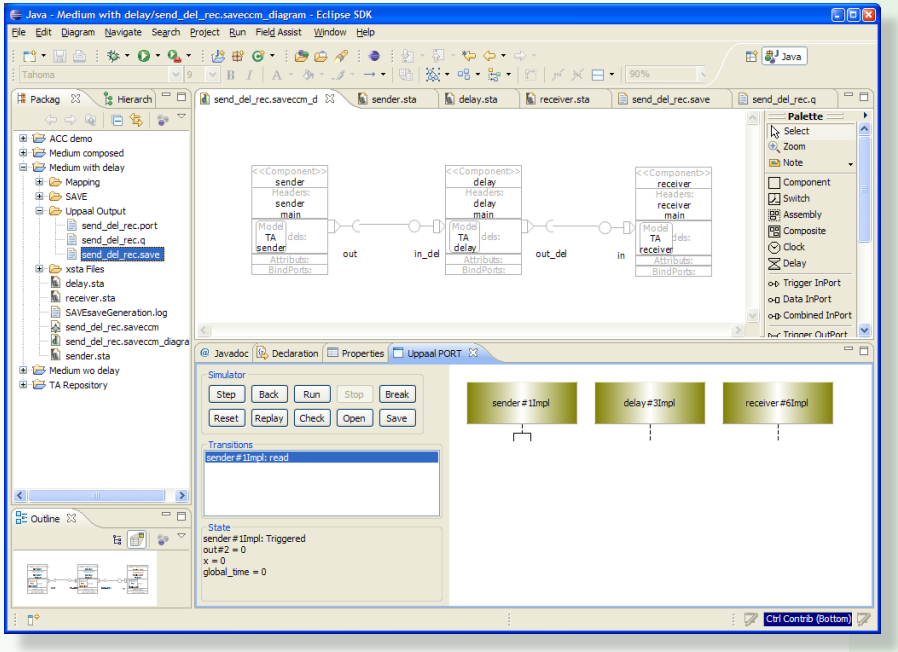
Preserve order of dependent transitions

Preserves start of component C



Implementation

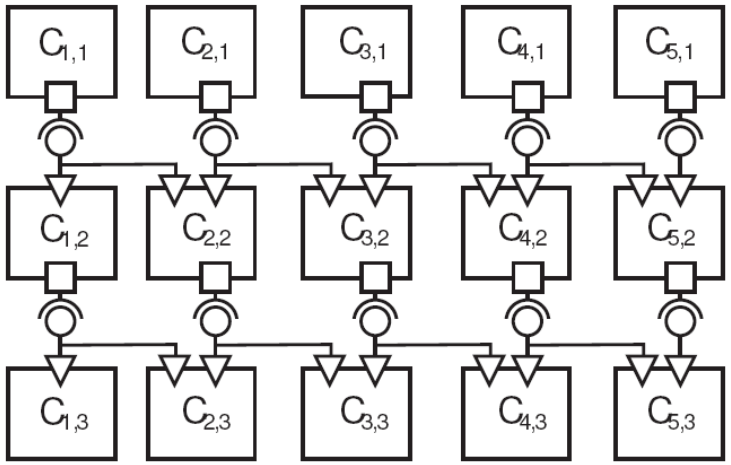
- UPPAAL PORT: Extension of model-checking tool UPPAAL with
  - partial order reduction
  - native support for component model ProSave
- IDE support for
  - editing and simulation of ProSave
  - PORT model-checking of local reachability
  - model-checking of safety and liveness



PROGRESS

Benchmark

- Each component:
  - initial
  - read
  - delay ∈ [0,4]
  - write



Method	3x2	3x3	3x4	4x3	4x4
global	264/0.15s	499/0.19s	838/0.25s	2553/0.45s	4778/0.88s
local	52/0.14s	78/0.17s	104/0.22s	145/0.25s	195/0.43s
	4x5	5x4	5x5	6x5	6x6
global	8146/1.7s	26108/5.2s	48096/11.7s	481318/2m29s	1087216/7m30s
local	245/0.76s	326/0.94s	411/2.0s	637/4.8s	770/9.8s
	6x7	7x4	7x5	7x6	7x7
global	2350413/19m40s	5166020/26m50s	⊥	⊥	⊥
local	903/17.7s	736/4.6s	932/10.7s	1128/21.5s	1324/39.0s

Table 1. Benchmark results. We use ⊥ to denote that the experiment did not terminate in 30 minutes. Based on example of [Salah, Bozga and Maler'06]

global = no partial order reduction
local =relaxed partial order semantics

PROGRESS

REMES

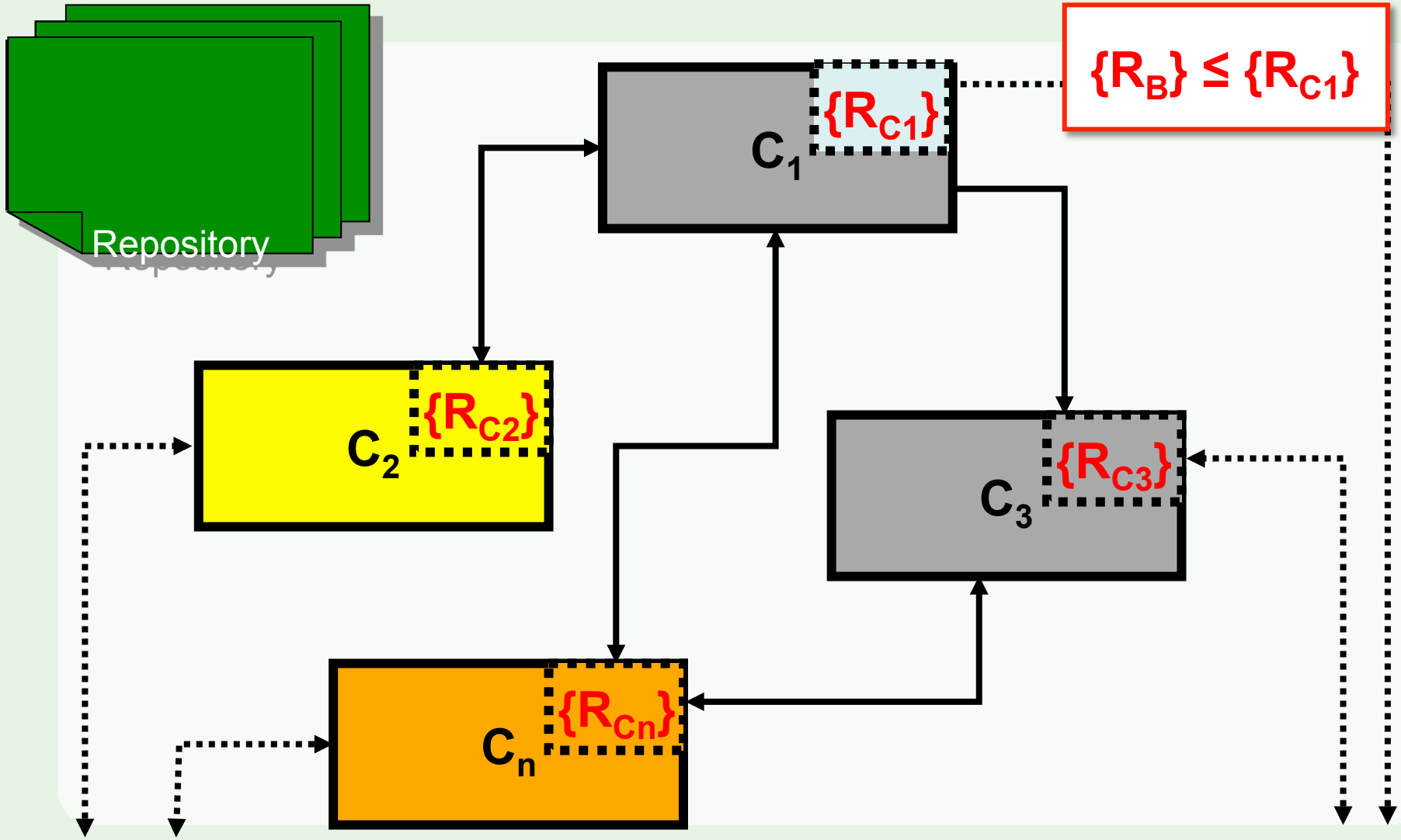
Modelling and verification of Embedded Resources

PROGRESS

Embedded Resources

- Embedded systems typically designed w.r.t. resource constraints
  - computational power (CPU), memory, energy, bus bandwidth, ports, etc.
- Challenge
  - provide early design stage modelling and prediction methods
  - model resource usage and provide analysis techniques
  - resource-wise feasibility, optimal resource usage, and trade-off analysis

Embedded Resources



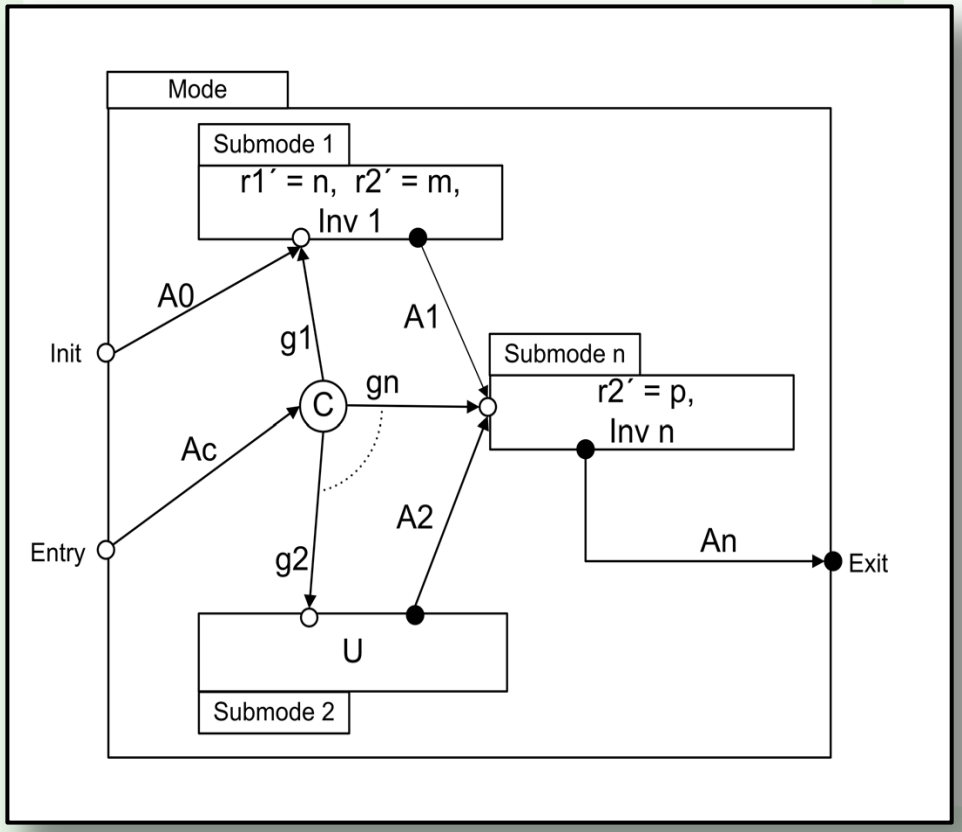
PROGRESS

Resource Analysis Problems

- Feasibility
  - Accumulated resource usage within provided resource bounds
- Optimal and worst-case consumption
  - Min/max accumulated resource usage
- Trade-off analysis:
  - Decide the best trade-off between multiple resources, possibly dependent, e.g., memory and cpu

REMES

- **RE**source **M**odel for **E**mbedded **S**ystems



- modes
  - atomic or composite
  - models behaviour and timing of component
- control points
  - init, entry, exit
- variables
  - integer, clocks (arrays)
  - global, local
- actions
- constraints (invariants)
- conditional connectors

PROGRESS

Embedded Resources

- r – accumulated resource consumption
- r’ – rate of consumption over time
  - discrete or continuous
  - referable or non-referable
- Examples:
  - r’=0 and referable : discrete, e.g. memory
  - r’=0 and non-referable : discrete, e.g., CPU
  - r’=n, n$\in$ Z, and non-referable : energy or bandwidth

Analysing REMES based ES

- REMES modes have access to resources $R_1,\ldots,R_n$
- Goal to analyse scenarios of resource usage
- Analysis model:

$$r_{tot} = (w_1 * r_1) + \ldots + (w_n * r_n)$$

  - r_{tot} : accumulated resource consumption for $R_1,\ldots,R_n$
  - $r_1,\ldots, r_n$: accumulated consumption of $R_1,\ldots,R_n$
  - $w_1,\ldots, w_n$: weights, relative importance of $R_1,\ldots,R_n$

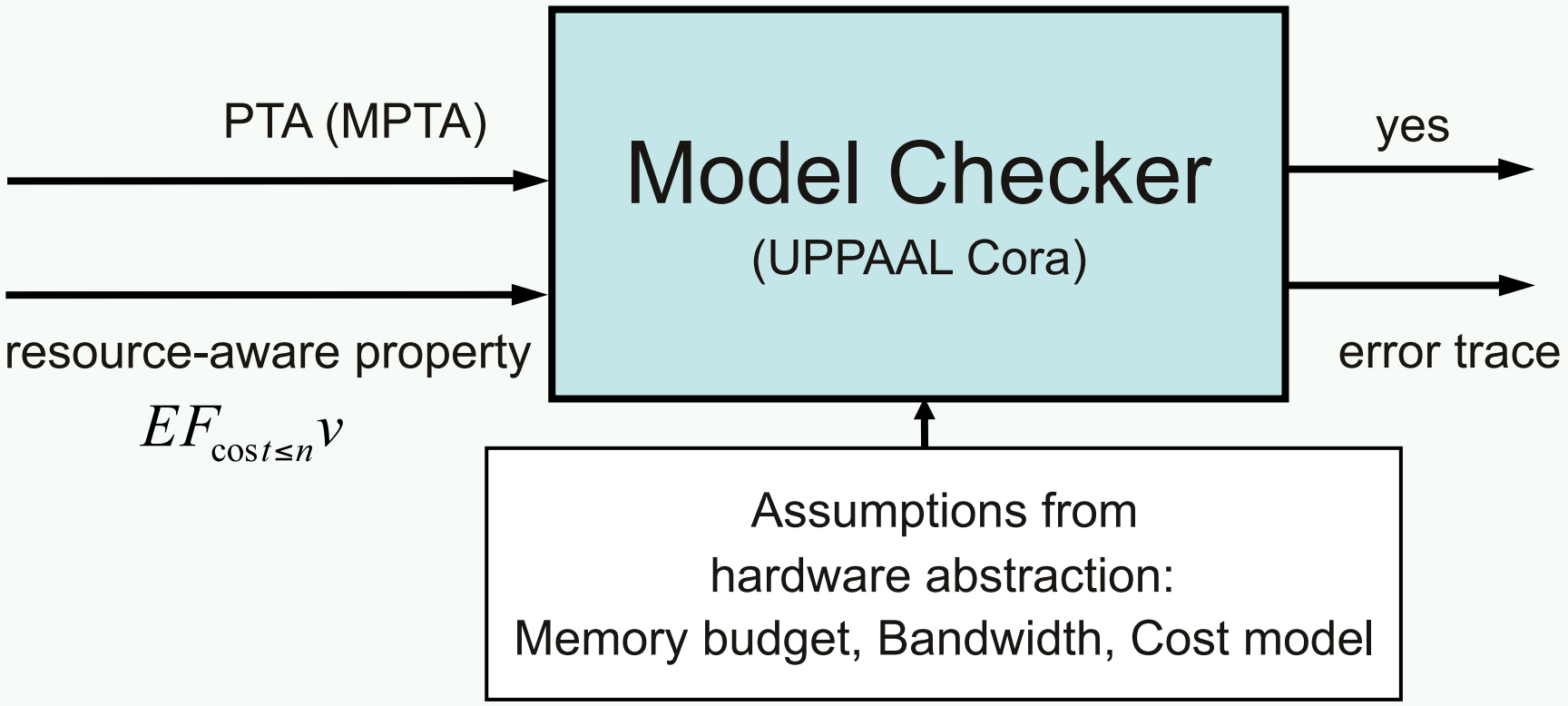
REMES to PTA

- Translation to Priced Timed Automata [Alur et al'01, Behrman et al'01]
  - Timed automata extended with linear cost variable
  - Minimum reachable, etc decidable
- Mapping of REMES modes, edges, variable, etc to PTA locations, edges, etc.
- Multiple resources $r_1, \ldots, r_n$:

$$cost = (w_1 * c_1) + \ldots + (w_n * c_n)$$

  - c1, …, cn : cost of resource $r_1, \ldots, r_n$
  - for each location: $cost' = (w_1 * c'_1) + \ldots + (w_n * c'_n)$ where $w_1, \ldots, w_n$ constants and $c'_1, \ldots, c'_n$ static

Analyzing REMES models



Resource Analysis

- Feasibility analysis:
  - are the accumulated values of consumed resources within the provided resource amounts?
  - one cost variable encoding all accumulated resources
    - Strong feasibility: $AF_{cost \leq n} v$

      $AG(q \Rightarrow AF_{cost \leq n} v)$
    - Weak feasibility: $EF_{cost \leq n} v$
    - Live feasibility: $AG(q \Rightarrow EF_{cost \leq n} v)$

Resource Analysis (2)

- Optimal and Worst-Case Resource Consumption
  - minimum/maximum cost for reaching given location or predicate
  - minimizing/ maximizing the one-cost function

$$cost = (w_1 * c_1) + \ldots + (w_n * c_n)$$

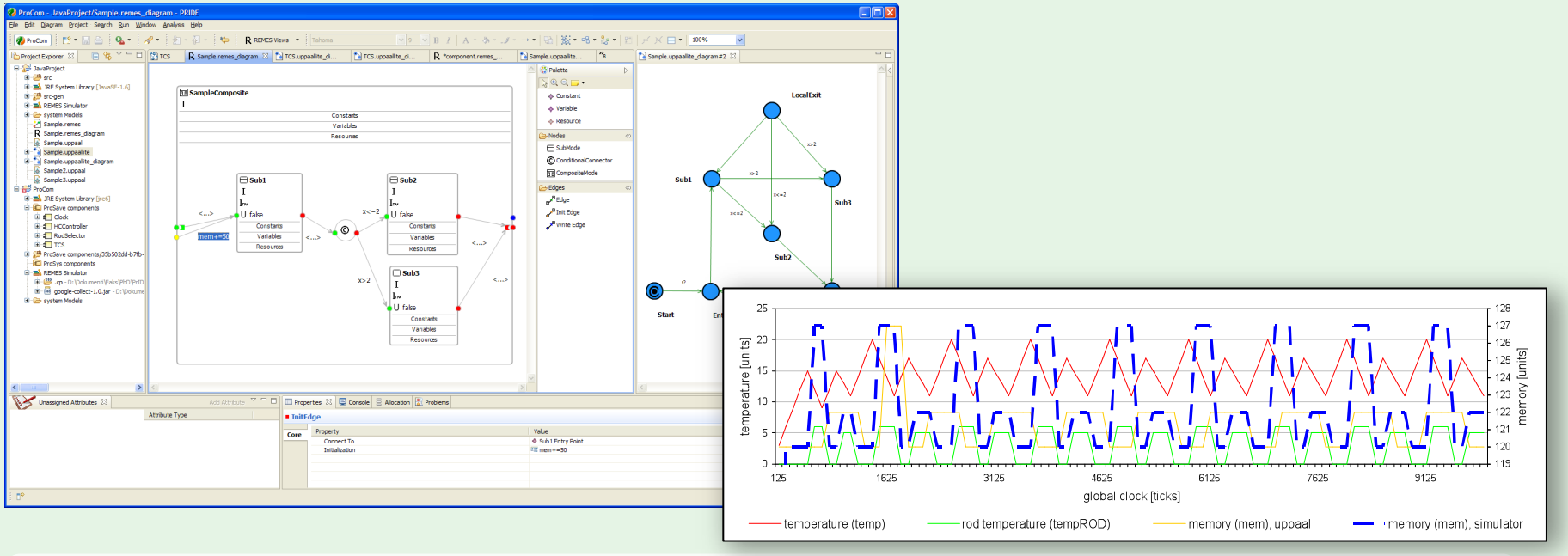
  - decidable also if cost is not a monotonically increasing function [Bouyer et al. “On the optimal reachability problem of weighted timed automata” 2007]

Resource Analysis (3)

- Trade-off analysis:
  - more than one property to satisfy
  - Pareto analysis
  - adjust weights and use weighted sum
    - weights can be set by AHP analysis
  - minimize a primary cost, while imposing an upper bound on secondary cost (two costs)
    - Optimal conditional reachability of MPTA [Larsen & Rasmussenm 2005]. E.g. energy minimized, cpu bounded, location v:

$$EF_{(w_{energy} \times c_{energy}) \leq n)}(v \wedge (w_{cpu} \times c_{cpu}) \leq m)$$

Integrated in PrIDE



- REMES and PTA editor/visualizer
- Transformation to TA or PTA
- Simulation (using UPPAAL)
- Verification by model-checking in UPPAAL, UPPAAL Port (partial-order) and UPPAAL Cora (PTA)

PROGRESS

Conclusion

This talk:

- Procom:
  - Progress component model
  - ProSys and ProSave
  - attriute framework
- PORT + REMES:
  - partial order reduction
  - modelling and analysis of embedded resources
  - feasibility, optimality and trade-off analysis
- PrIDE: Progress IDE
  - modelling, simulation, model-checking (functional, timing, resources)

Not in this talk:

- Modelling:
  - UML state machines + MARTE
  - connection to EAST-ADL2
  - modelling patterns
  - error modelling
- Predictability analysis
  - static analysis
  - WCET
  - dependability
- Platform
  - scheduling
- Case studies, …

Thanks!

Acknoledgments: Tomáš Bureš, Jan Carlson, Ivica Crnkovic, John Håkansson, Marin Orlić, Cristina Seceleanu, Séverine Sentilles, Petr Stepan, Jagadish Suryadevara, Aneta Vulgarakis