The "5Cs" and other meta-models for Embedded Control Systems

Herman Bruyninckx
University of Leuven

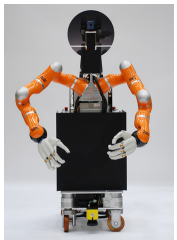
Eindhoven, October 2, 2013

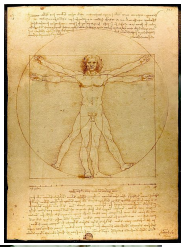
Objectives

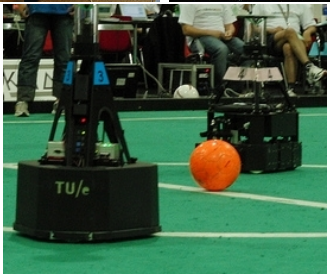
- to bring **structure** in the software:
  - architecture — **structural** composition
  - **behavioural** composition
- to bring **modelling** in the software development process
- to tackle **system-of-systems** complexity
- to evolve from *cleverly programmed* systems to **clever systems**
- to explain the limits of "Simulink"

Responsibilities Embedded Control Systems

- **sensing**
- **control**
- **estimation**
- **perception**
- **coordination**
- **planning**
- **scheduling**
- **monitoring**
- **cognition**
- **modelling**
- **reasoning**
- **. . .**







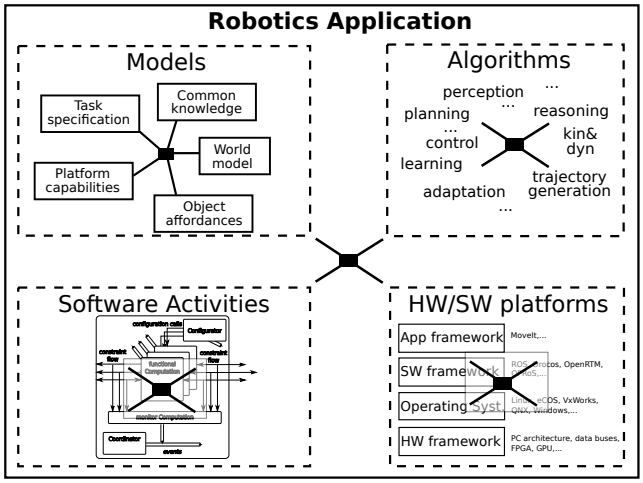
Complexity of "Robotics Application"

integration

=

=

challenge!



Models + Algorithms + SW Activities

Models: "knowledge" **(Not in "Simulink"!)**

- **represent** relationships between primitives
- **integrates** continuous, discrete as well as symbolic parts!

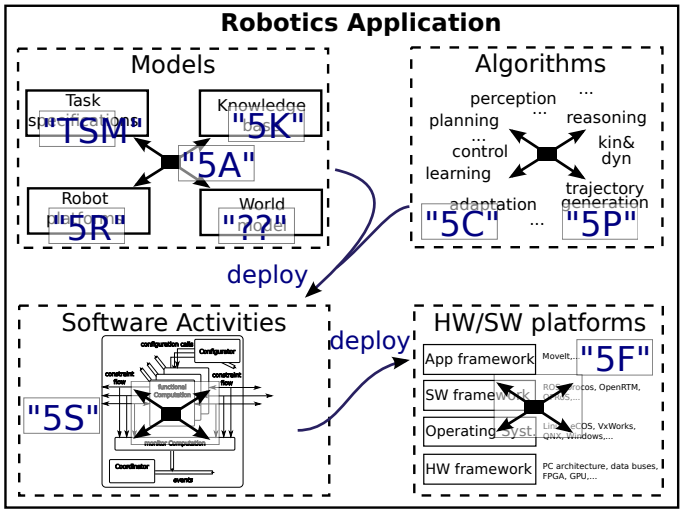
Algorithms: "computations"

- **transform** input data into output data
- conforming to a model of these transformations
- reasoning = model transformation

Software Activities: "behaviour" **(Not in "Simulink"!)**

- **execute** algorithms (continuous behaviour)
- **configure** & **coordinate** the execution (discrete behaviour)
- **communicate** data (interaction behaviour)

Models + Algorithms + SW Activities (2)



SW Activities Coordination — the "5Cs"

How are computations, communications, configurations, and coordinations interacting?

Composition

Coordination	When must components change their behaviour?
Configuration	What parameters define the behaviour of all components?
Computation	What functionality is computed?
Communication	How are results of computations being communicated?

- Computation: continuous behaviour (e.g. controller)
- Coordination: discrete behaviour (e.g. switch controllers)
- Configuration: behaviour-transition behaviour
- Communication: interaction behaviour

5Cs not supported in "Simulink"!

Coordination via "state machines" "5S" System complexity

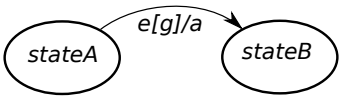
Coordination	
PLC	Strictly serialized behaviour
FSM	Single thread coordination
Orchestration	FSMs in *hierarchical peer-to-peer* structural model
Choreography	FSMs in *friendly hierarchical peer-to-peer* structural model
Competition	FSMs in *adversary peer-to-peer* environments

Only PLC/FSM supported in "Simulink"!

Problem with "Simulink" StateCharts

Statecharts = set of states with transitions "$e[g]/a$":

- e = event name
- $[g]$ = guard condition
- a = action



Upon receiving event e, if there is a transition from current `stateA` on e with $[g]$ evaluating to `true`, then the transition is triggered, executing a and moving the Statechart to the target `statestateB`

Evaluation of guard g can be complex function, not linked one-to-one to transitions, and executed in concurrent activity; execution of action a can be complex, realised by concurrent activities.

⇒ StateCharts behaviour semantics are under-defined!

Computation = constrained optimization

General:

task state & domain	$X \in \mathcal{D}$
desired state	X_d
robot state & domain	$q \in \mathcal{Q}$
objective function	$\min_q f(X)$
equality constraints	$g(X) = 0$
inequality constraints	$h(X) \leq 0$
tolerances	$d(X, X_d) \leq A$
solver	algorithm computes q

Domain: fills in types for f, X, q for a particular domain, a particular "robot", and a particular type of solver

Application:

- fill in parameter values for f, X,...
- fill in concrete solver implementation

Levels of abstraction – Hierarchical structure

Abstractions	
Sensor/Actuator	Direct interactions with the real world
Feature/setpoint	Transformations between real world and software representations
Object	Combination of Features into software properties of, and operations on, objects
Task	Series of activities performed on Objects
Mission	Series of Tasks to achieve a purposeful end-result

Levels are **not** strictly hierarchical **boundaries**:

- one level takes **constraints** from level "above"...
- ...and **objective functions** from level "below"

Literature warning: mostly single-level optimization only!

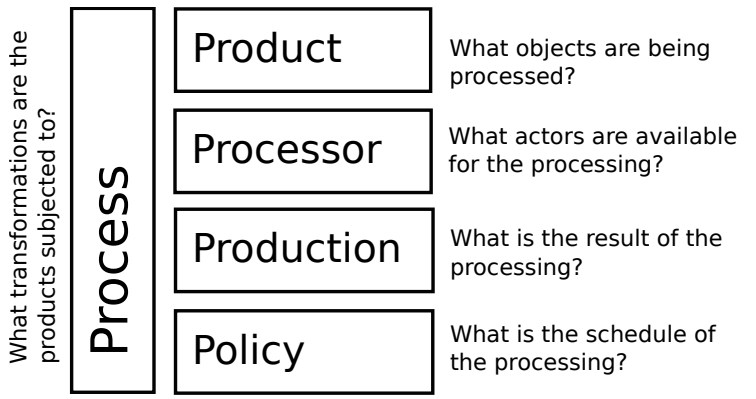
Control

Create/adapt world model based on the other Blocks	Modelling	(control) Blocks	
		Prediction	How to act only on basis of knowledge about world?
		Action	Act on the world!
		Sensing	How does the world react to the Action?
		Monitoring	How well are Prediction and Sensing in agreement?

Control loops exist

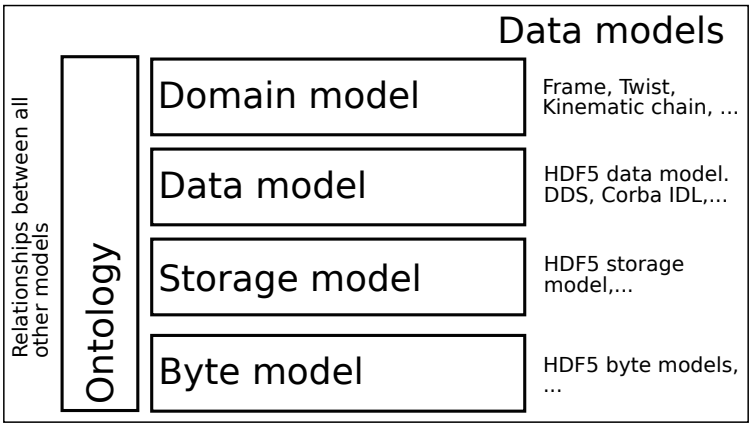
- from/to level above: *Prediction*, *Monitoring*
- at all levels: *Modelling*;
- to level below: *Action*, *Sensing*.

Computations: Process model ("5Ps")

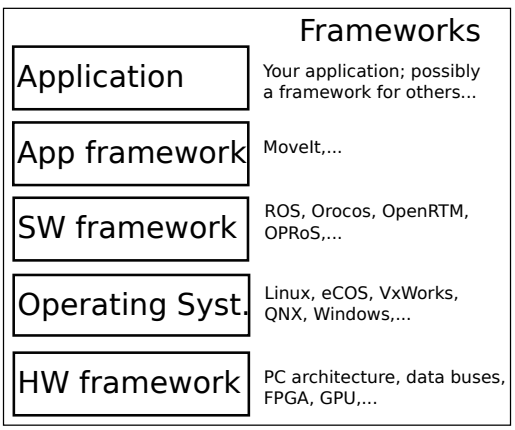


- **pure functional** programming!
- **scheduling** of computations! ($\leftrightarrow$ "Simulink")
- **scheduling** of data communication! ($\leftrightarrow$ "Simulink")

Data communication models ("5Ds")

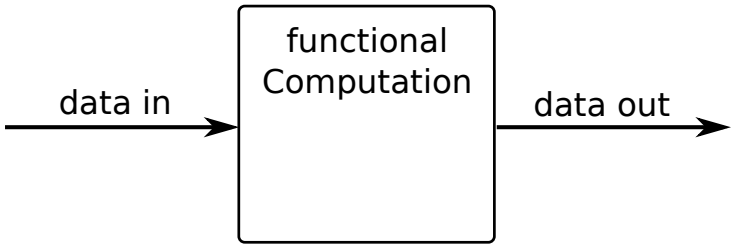


Frameworks ("5Fs")



QoS[1]-aware Compostion pattern

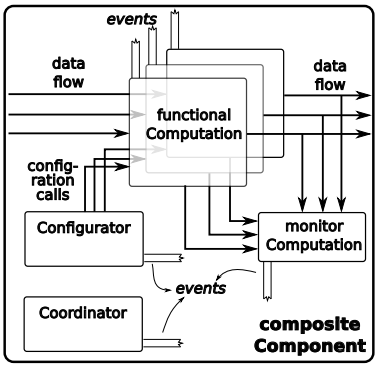
From current mainstream "data flow" composition...



[1]QoS = Quality of Service

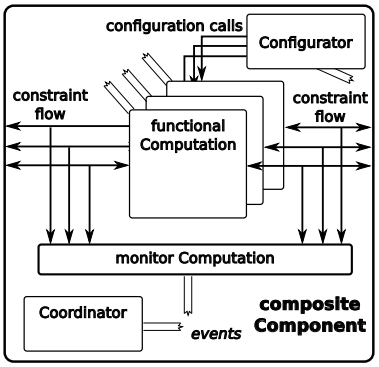
QoS-aware Compostion pattern (2)

...to composition pattern with uni-directional interaction...



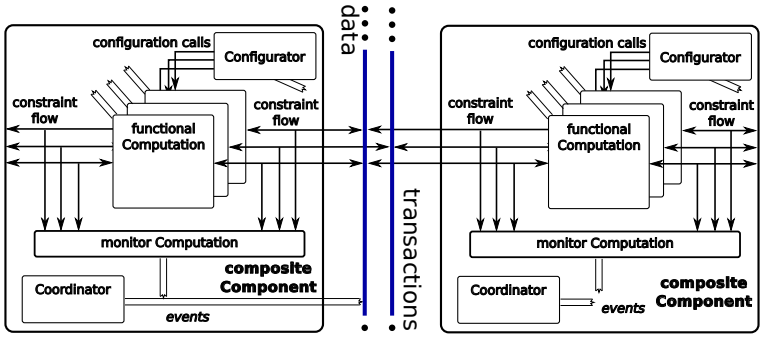
QoS-aware Compostion pattern (3)

...to full bi-directional, QoS-aware interaction...



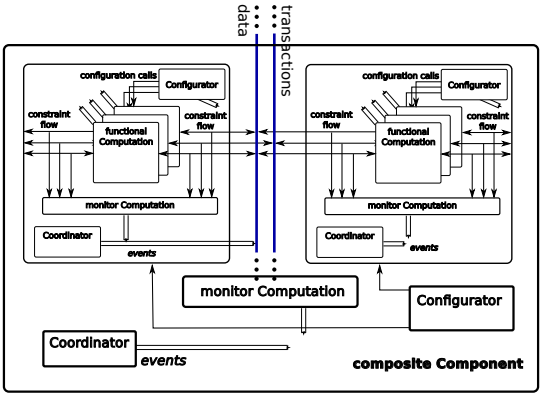
QoS-aware Compostion pattern (4)

. . . and eventually to data and transaction busses with runtime (re)configuration over all "5K" communication!

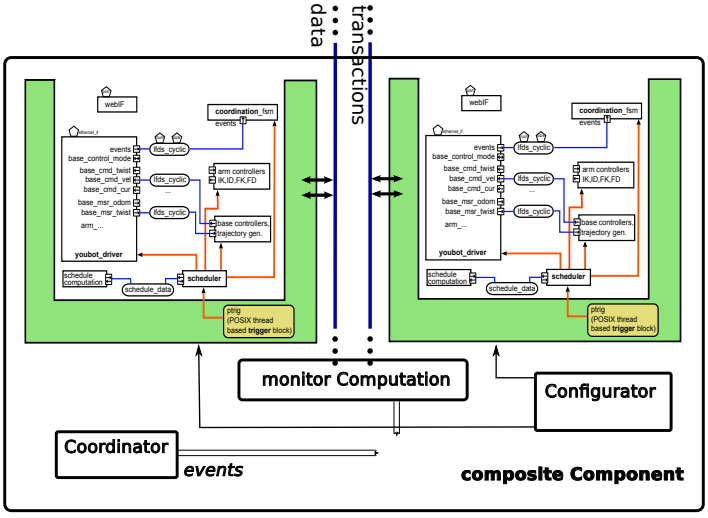


QoS-aware Compostion pattern (5)

. . . to full system-to-system bi-directional, QoS-aware Coordination interaction. . .



SW Activities — "micro" components



Conclusions

- Embedded Control Systems: become connected **system-of-systems**
- Trend towards more **"code generation"**
- But "code generation" is also:
  - **composition** of many system-level **models**
  - **model-based template filling** of architectural patterns
- Model-based **runtime introspection** for **adaptability**
- "Simulink" misses:
  - adaptability via **computational schedule**
  - deployment on **architectural patterns**

Thank you for your attention!

Herman Bruyninckx
University of Leuven
http://people.mech.kuleuven.be/~bruyninc

The author acknowledges the support from the KU Leuven Geconcerteerde Onderzoeks-Acties (**GOAs**) *Model based intelligent robot systems* and *Global real-time optimal control of autonomous robots and mechatronic systems*, and from the European Union's 7th Framework Programme (FP7/2007–2013) projects **BRICS** (FP7-231940, *Best Practice in Robotics*), **ROSETTA** (FP7-230902, *Robot control for skilled execution of tasks in natural interaction with humans; based on autonomy, cumulative knowledge and learning*), **RoboHow.Cog** (FP7-288533, *Web-enabled and experience-based cognitive robots that learn complex everyday manipulation tasks*), **SHERPA** (FP7-600958, *Smart collaboration between Humans and ground-aErial Robots for im-Proving rescuing activities in Alpine environments*), **PicknPack** (FP7-311987, *lexible robotic systems for automated adaptive packaging of fresh and processed food products*), and **euRobotics** (FP7-248552, *European Robotics Coordination Action*).