Horizon 2020
European Union funding
for Research & Innovation

CERBERO

HW/SW Cyber-System Co-Design and Modeling

Julio OLIVEIRA

Karol DESNOS

Introduction

Who are we?

TNO

IETR

INSA RENNES

Julio de OLIVEIRA

Position:

- TNO - Researcher & innovation scientist

Topic of interest:

- Large, distributed, and autonomous systems

Karol DESNOS

Position:

- INSA Rennes – Assoc. Prof.
- IETR - Researcher

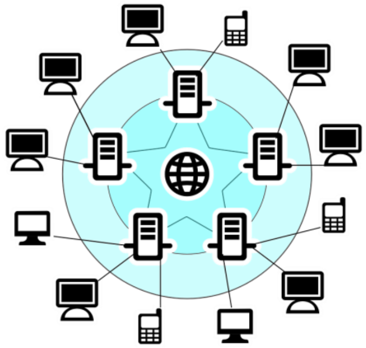
Topic of interest:

- Dataflow Programming
- Embedded MPSoCs

Introduction

Text-book definitions for Cyber-Physical Systems :

- CPS are complex systems integrating:
  - Computation processes
  - Network of communication
  - Physical entities *(actuators and sensors, time, mechanics, temperature, …, and you!)*



- CPS is an engineering discipline, focused on technology, with a strong foundation in mathematical abstractions.

source: Berkeley CPS website, http://cyberphysicalsystems.org/

Introduction

Abstraction?

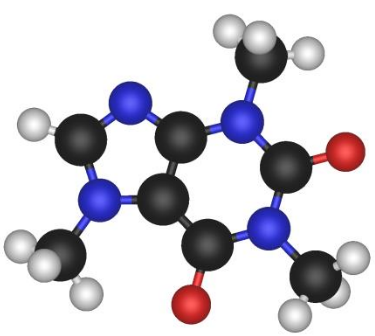
- Tradeoff between level of details and complexity.

Introduction

What is a model?

- Abstract (mathematically grounded) representation capturing predictable characteristics of a “system”.
- Models for similar constituents may take many forms.
  - To capture different characteristics.
  - To be more suitable for a different system size.

Molecules



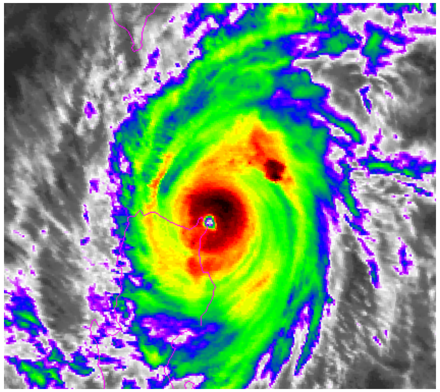
Ideal Gas Law

pressure *moles* *temp.*

$$pV = nRT$$

volume *gas constant**

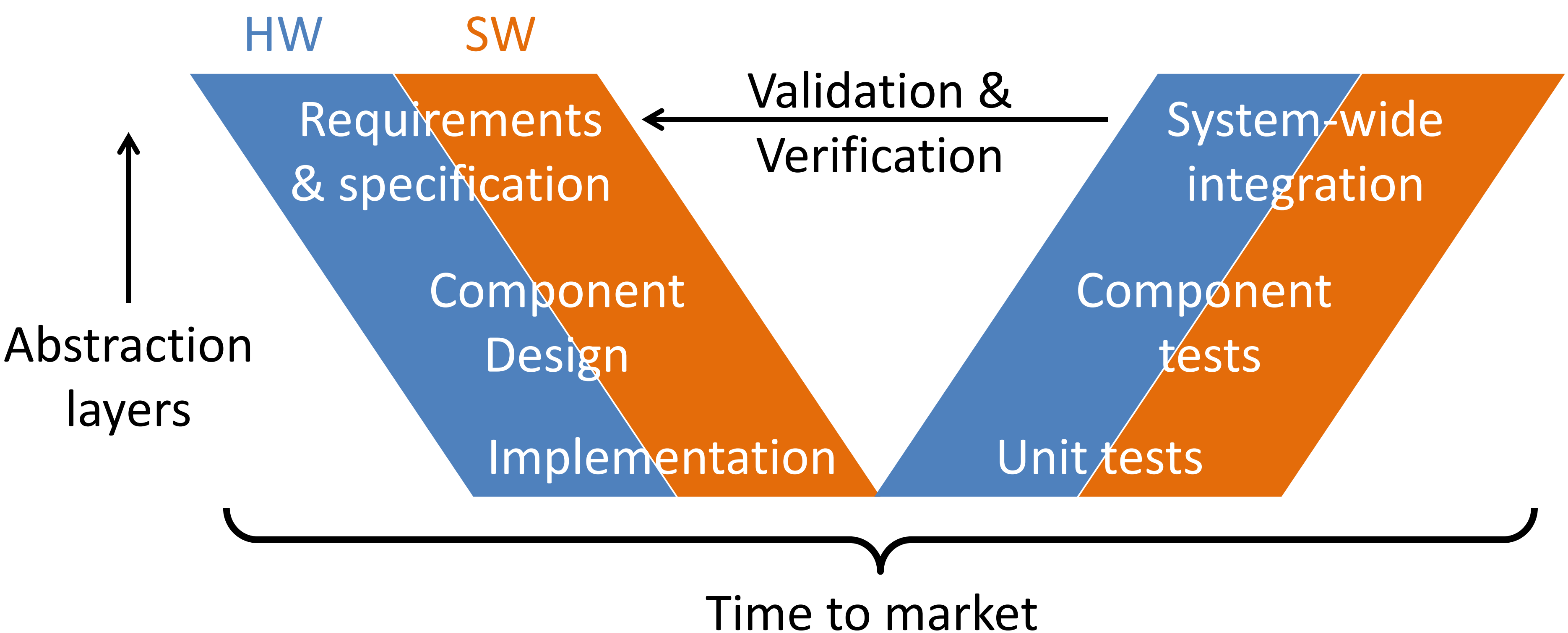
Meteorology



*: Physicist way of saying magic number

CPS Co-Design?

- “Old” embedded system co-design flow:



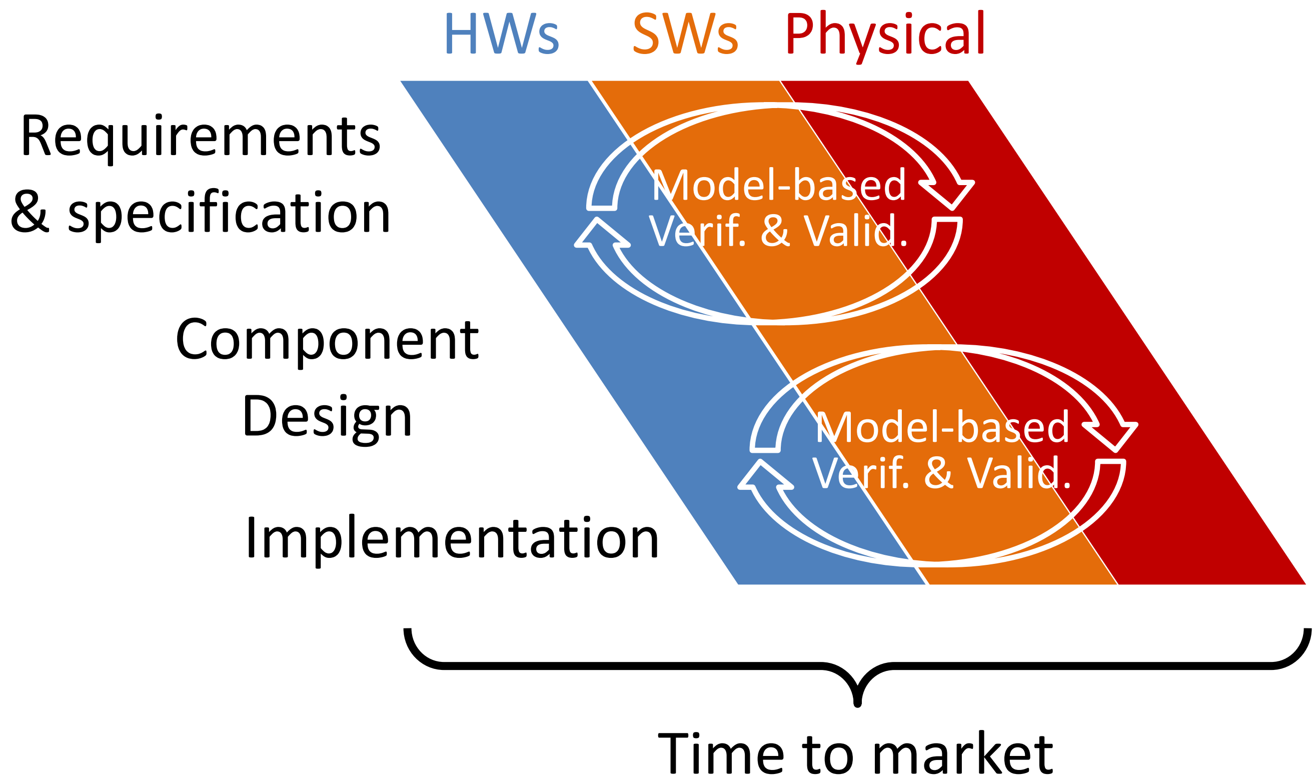
- Where are physical concerns?

Horizon 2020
European Union funding
for Research & Innovation

Introduction

CPS Co-Design?

- Model-based CPS co-design flow:



- Models enable: system assessment, system validation & verification, (automated) system implementation.

Horizon 2020
European Union funding
for Research & Innovation

Introduction

CPS design is complex

- Many building blocks,
- Separate but intricate optimization objectives,
- Many design constraints.

- **Application Constraints**
  - **Real-time requirements**
  - **Reliability constraints**
  - **Limited size and power**
- **Cost Constraints**
  - **Engineering cost**
  - **Production cost**
- **External Constraints**
  - **Regulation and Standards**
  - **Environmental constraints**

Introduction

Course objective

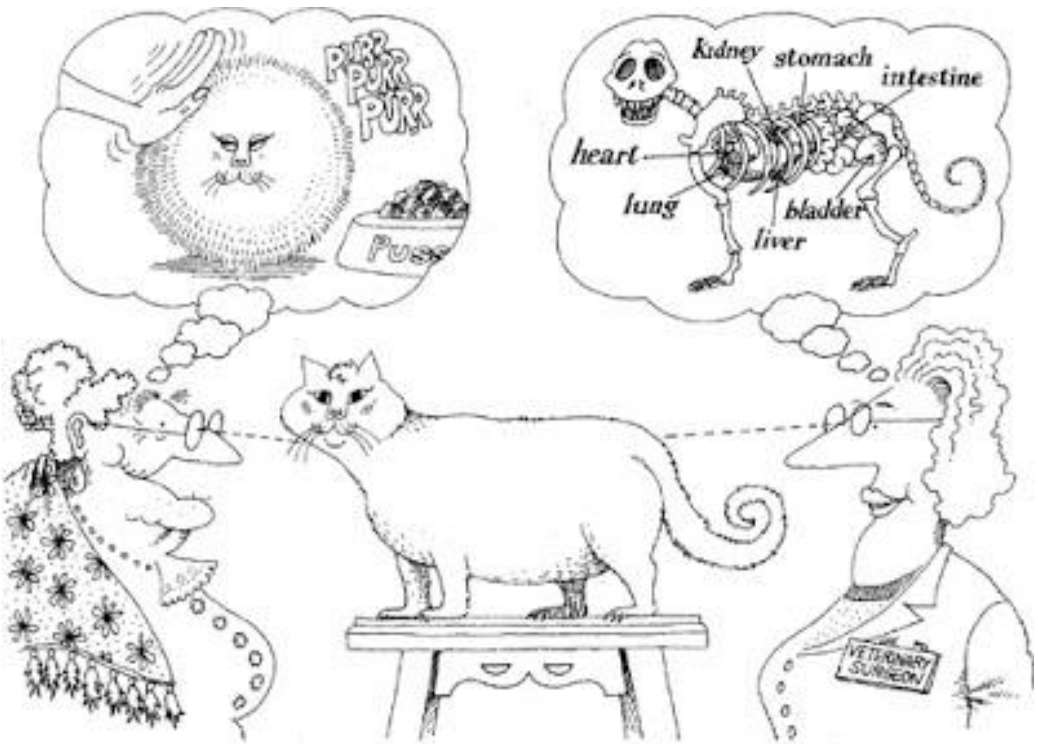
1. **HW/SW Cyber-System Co-Design and Modeling**
   Get a sense of how, why, and which models are used at different levels and steps of CPS design.

2. **HW/SW Cyber-System Modeling Tools**
   What do tools do to (automatically) build actual CPS systems from abstract models.

System Level Modeling

System-level Modeling

Modeling as an engineering activity

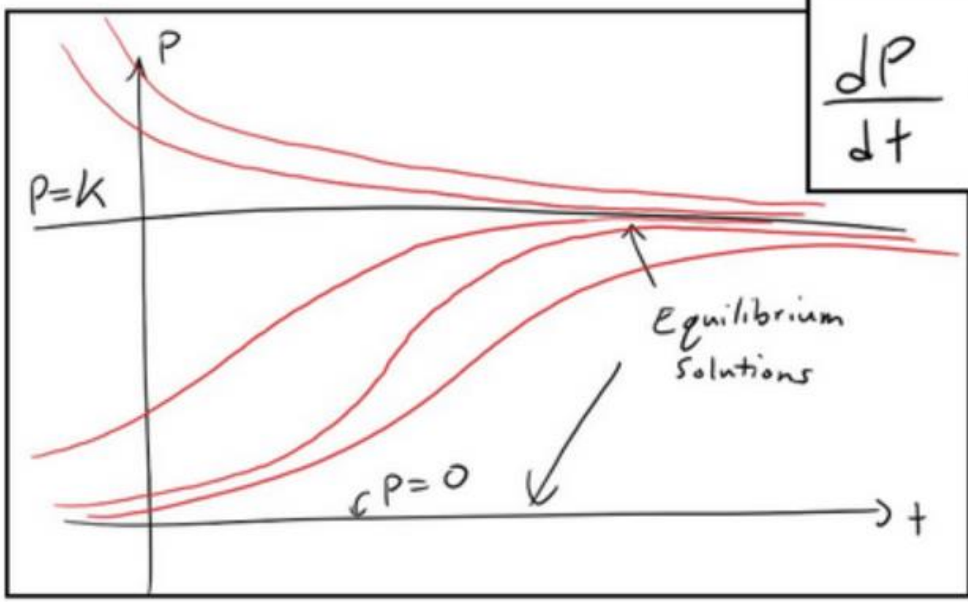


Abstraction (Simplification)

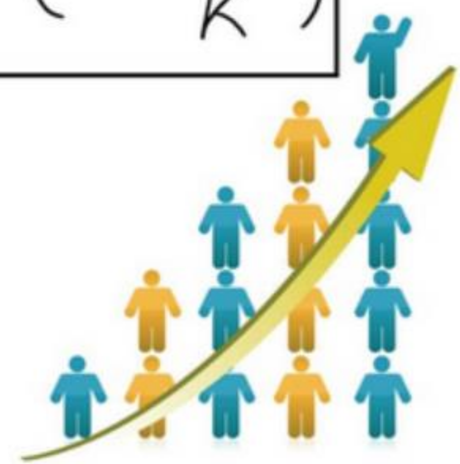
Description (Specification)

Operational (Executable)

Modeling – Example



$$\frac{dP}{dt} = kP\left(1 - \frac{P}{K}\right)$$



System-level Modeling

Why modeling CPS (SoS) is challenging?

Which abstraction?

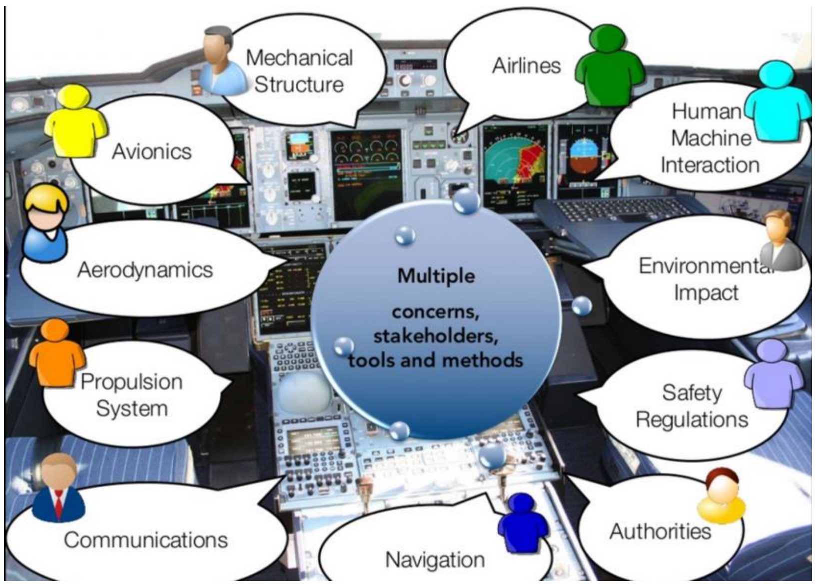
How to describe?

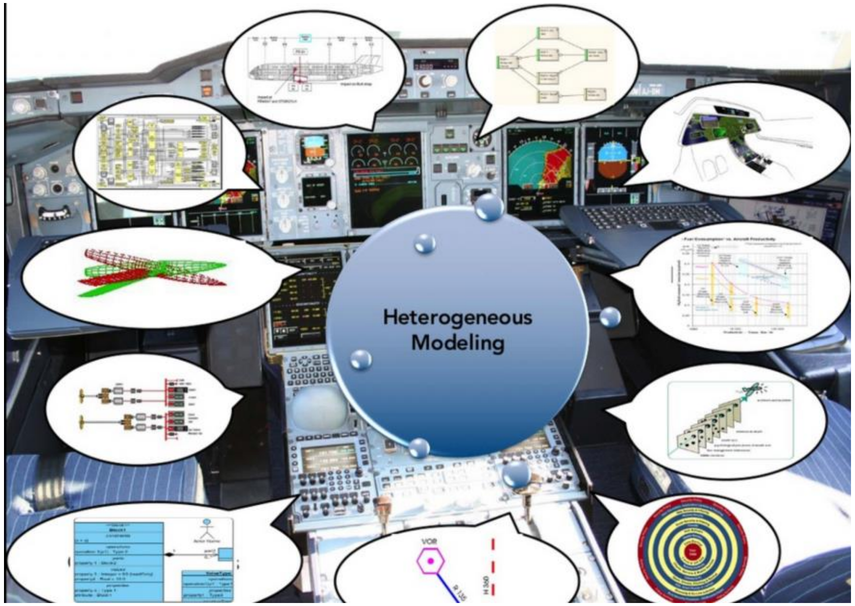
Operational?

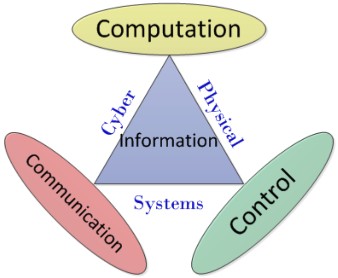
Complexity!

System-level Modeling

What we mean by complexity?



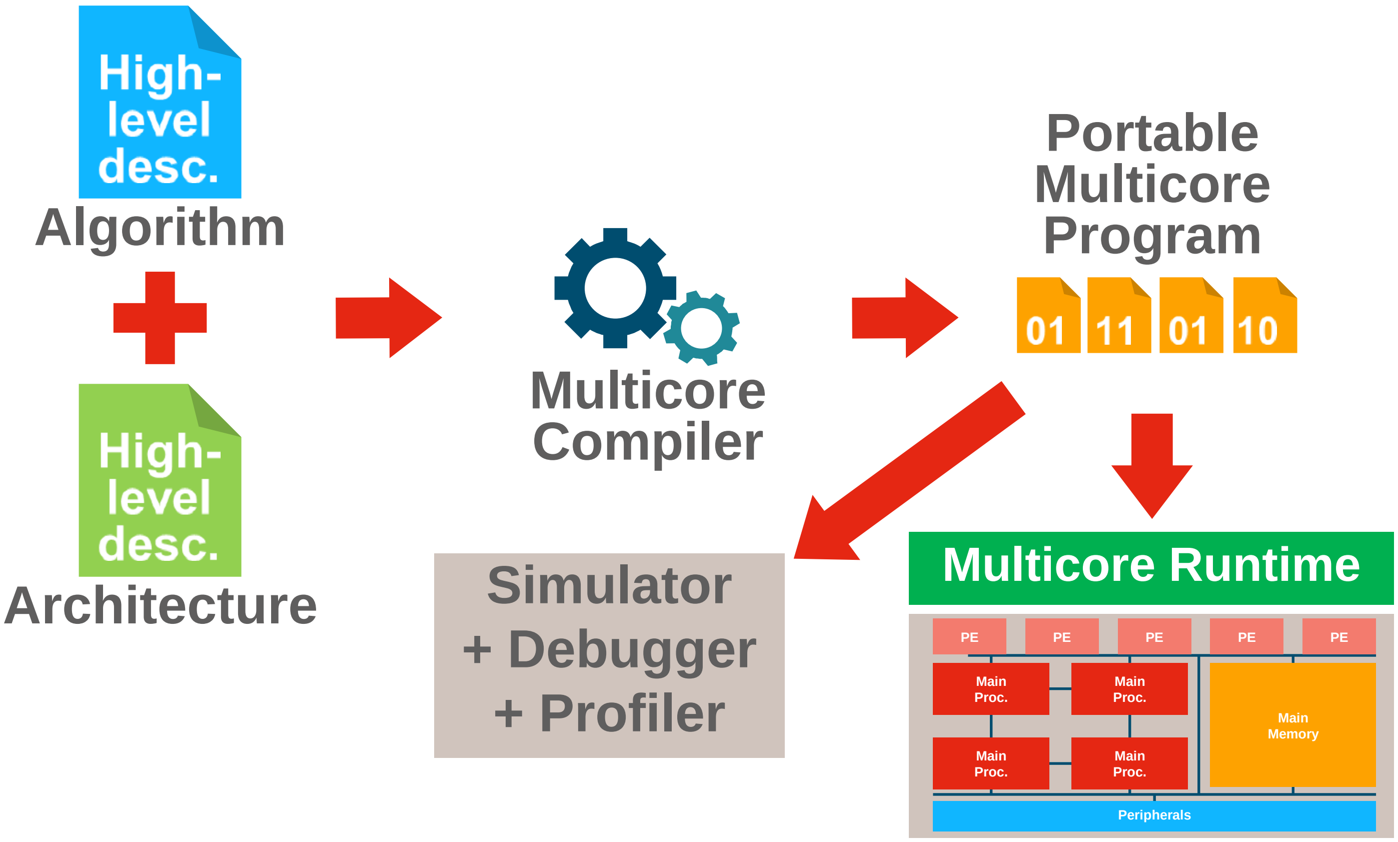




Source: INRIA

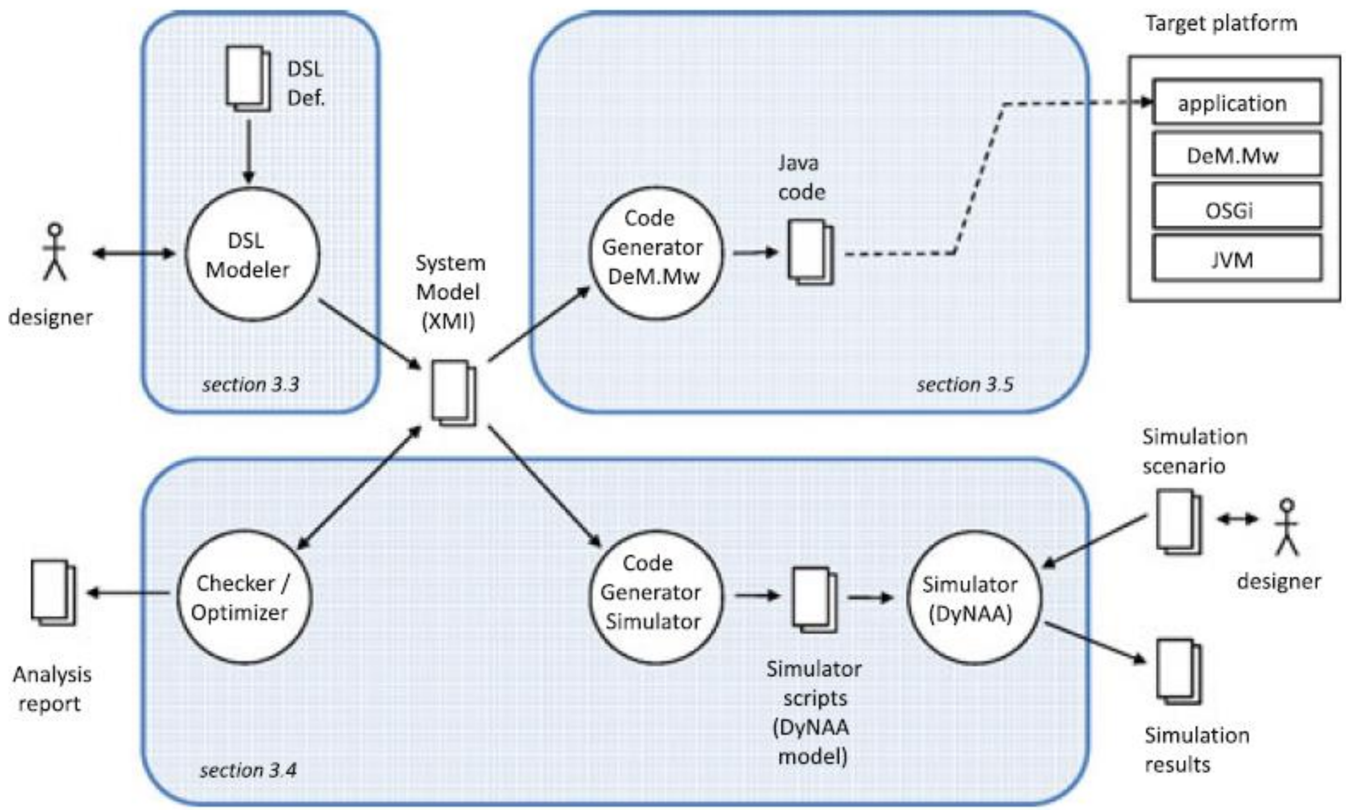
System-level Modeling

Why bother?



System-level Modeling

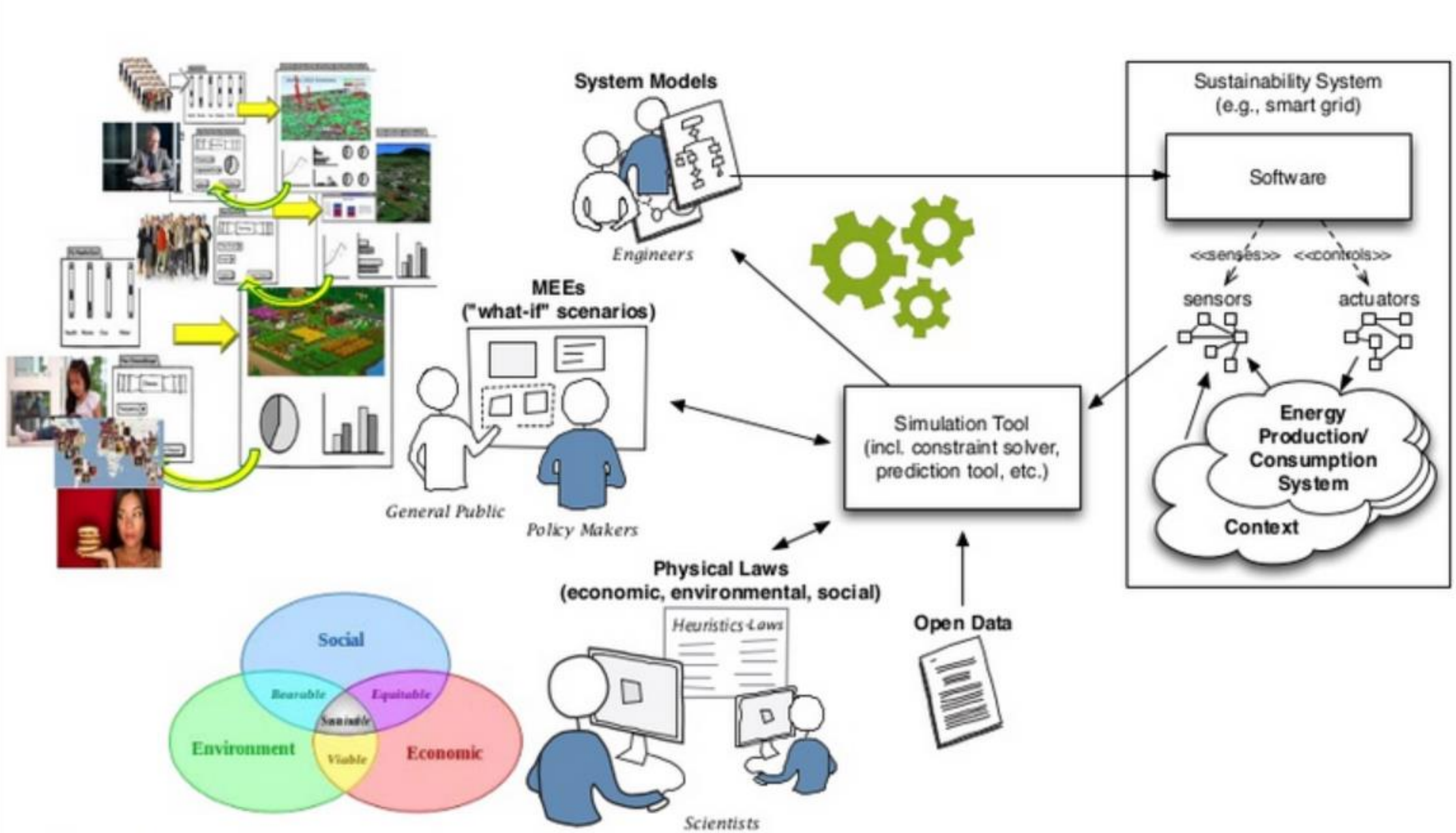
Why bother?



Source: EU DEMANES project

System-level Modeling

Why bother?

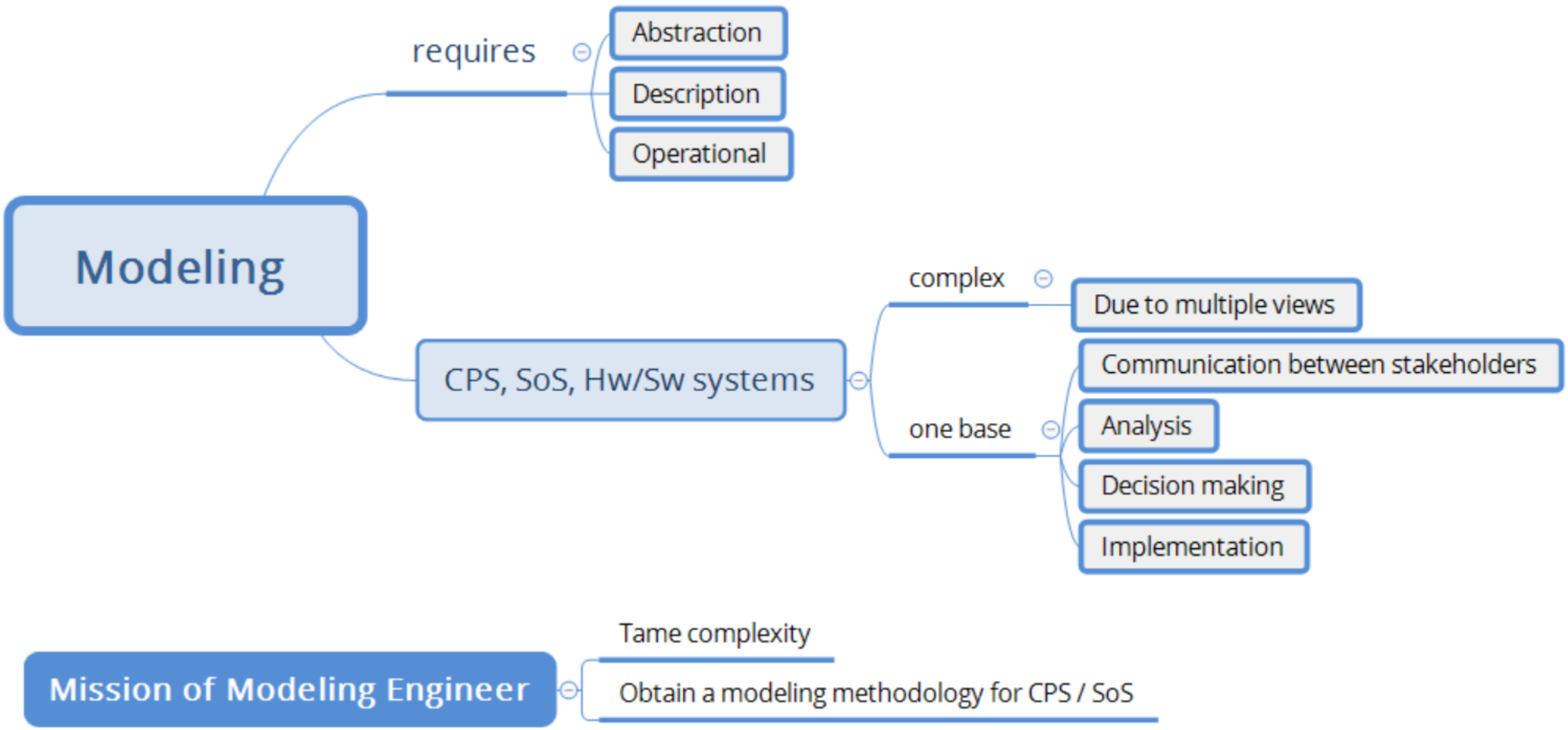


Source: INRIA

Horizon 2020
European Union funding
for Research & Innovation

System-level Modeling

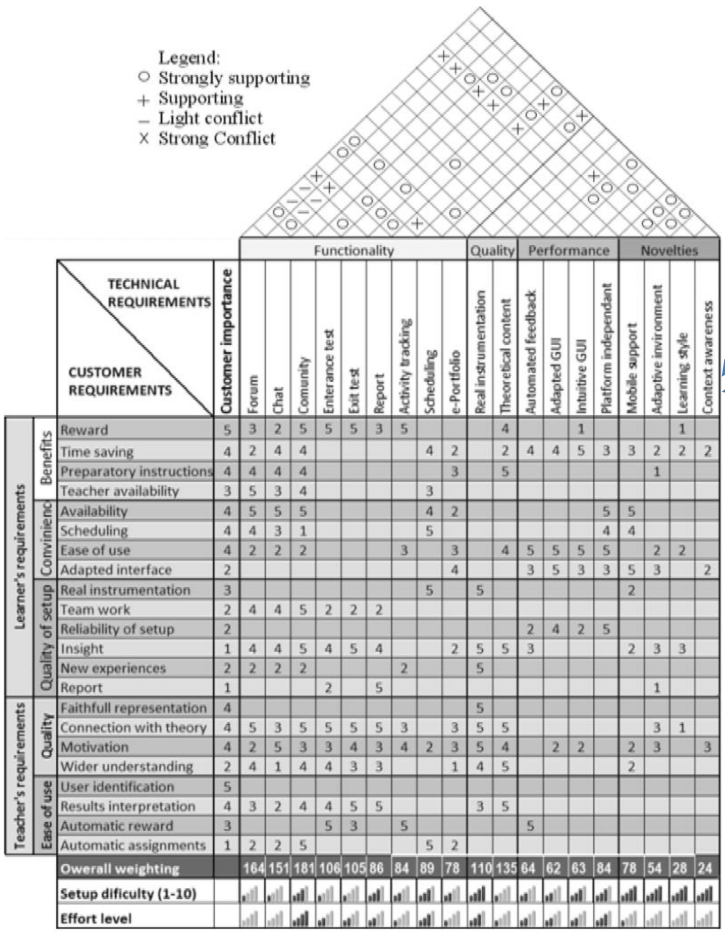
In a nutshell



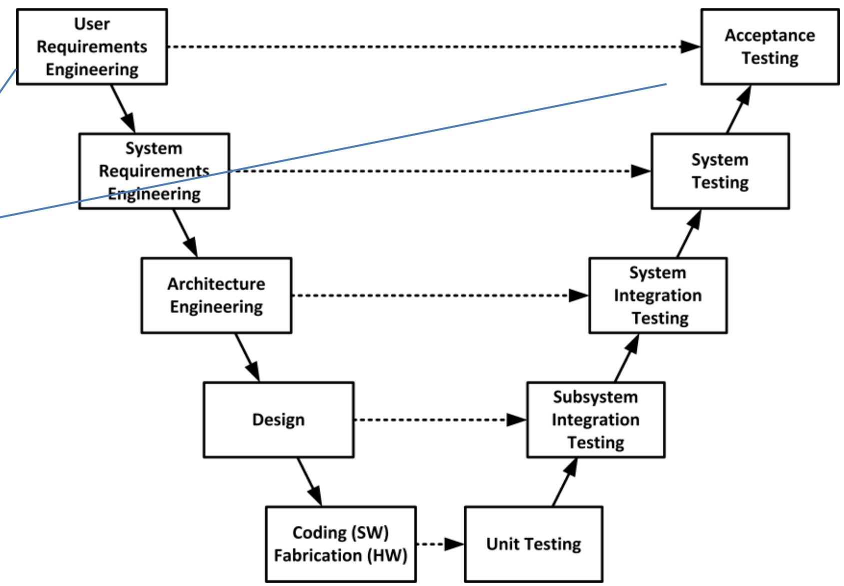
Ways to approach system level modeling

System-level Modeling

Approach 1: Model for the task in hand



		CUSTOMER REQUIREMENTS	Customer importance	Forum	Chat	Comunity	Enterance test	Exit test	Report	Activity tracking	Scheduling	e-Portfolio	Real instrumentation	Theoretical content	Automated feedback	Adapted GUI	Intuitive GUI	Platform independant	Mobile support	Adaptive invironment	Learning style	Context awareness
				Functionality									Quality		Performance				Novelties			
Learner's requirements	Benefits	Reward	5	3	2	5	5	5	3	5				4			1				1	
		Time saving	4	2	4	4					4	2		2	4	4	5	3	3	2	2	2
		Preparatory instructions	4	4	4	4						3		5						1		
		Teacher availability	3	5	3	4					3											
	Convinience	Availability	4	5	5	5					4	2						5	5			
		Scheduling	4	4	3	1					5							4	4			
		Ease of use	4	2	2	2				3		3		4	5	5	5	5		2	2	
		Adapted interface	2									4			3	5	3	3	5	3		2
	Quality of setup	Real instrumentation	3								5		5						2			
		Team work	2	4	4	5	2	2	2													
		Reliability of setup	2												2	4	2	5				
		Insight	1	4	4	5	4	5	4			2	5	5	3				2	3	3	
		New experiences	2	2	2	2				2			5									
		Report	1				2		5											1		
Teacher's requirements	Quality	Faithfull representation	4										5									
		Connection with theory	4	5	3	5	5	5	5	3		3	5	5						3	1	
		Motivation	4	2	5	3	3	4	3	4	2	3	5	4		2	2		2	3		3
		Wider understanding	2	4	1	4	4	3	3			1	4	5					2			
	Ease of use	User identification	5																			
		Results interpretation	4	3	2	4	4	5	5				3	5								
		Automatic reward	3				5	3		5					5							
		Automatic assignments	1	2	2	5					5	2										
		Owerall weighting		164	151	181	106	105	86	84	89	78	110	135	64	62	63	84	78	54	28	24
		Setup dificulty (1-10)																				
		Effort level																				



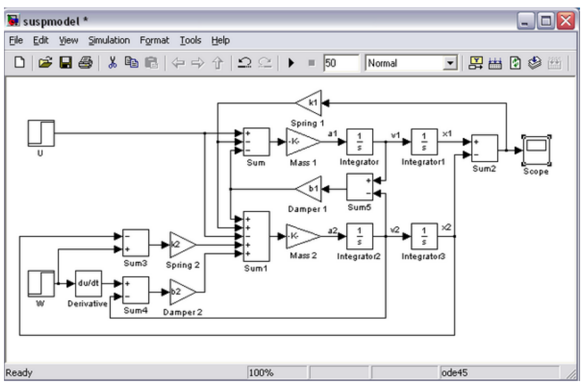
Approach 1: Model for the task in hand

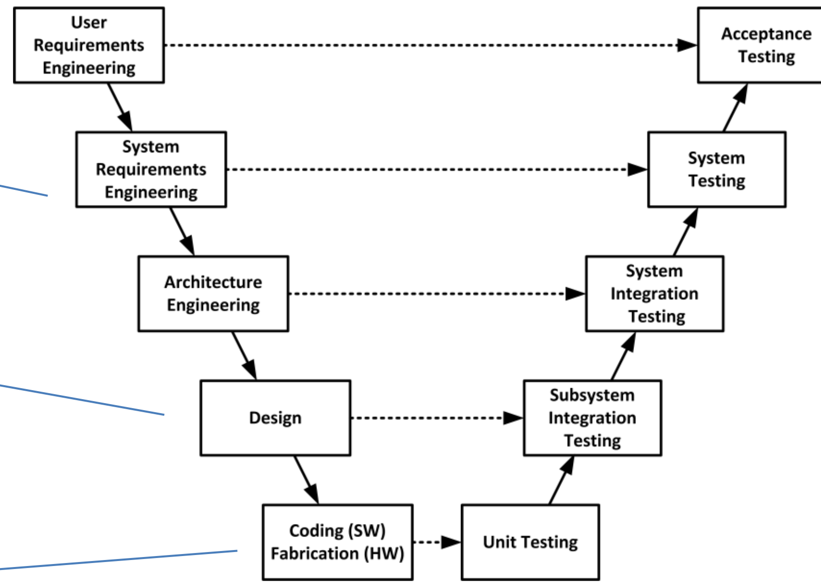
7. Procedure

This section describes the steps to be followed by the user in the Oracle Application with detail screen shots. After successful log in into the Oracle Application the user has to follow the following navigation to create a manual/standalone invoice in the system.

Prerequisite: Before navigating to the application the user should have following:

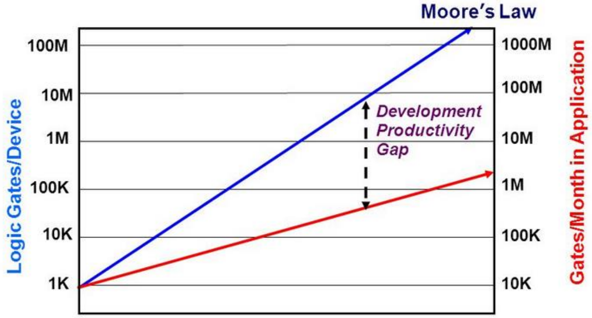
- Original copy of the vendor invoice.
- Copy of the manual PO/WO.
- Certificate of completion/ Proof of receipt of goods.





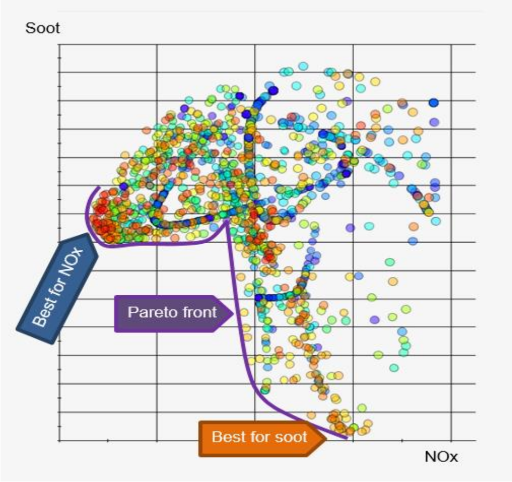

```
<?php

namespace SampleApp\Common;

class ServiceLocator implements RegistrableInterface
{
    protected $_resources = array();

    /**
     * Set the specified resource
     */
    public function set($key, AbstractResource $resource)
    {
        if (!isset($this->_resources[$key])) {
            $this->_resources[$key] = $resource;
```

Model for the task in hand fails



Major problem for the development productivity

Introduction of errors :
Human failure or mis-interpretation

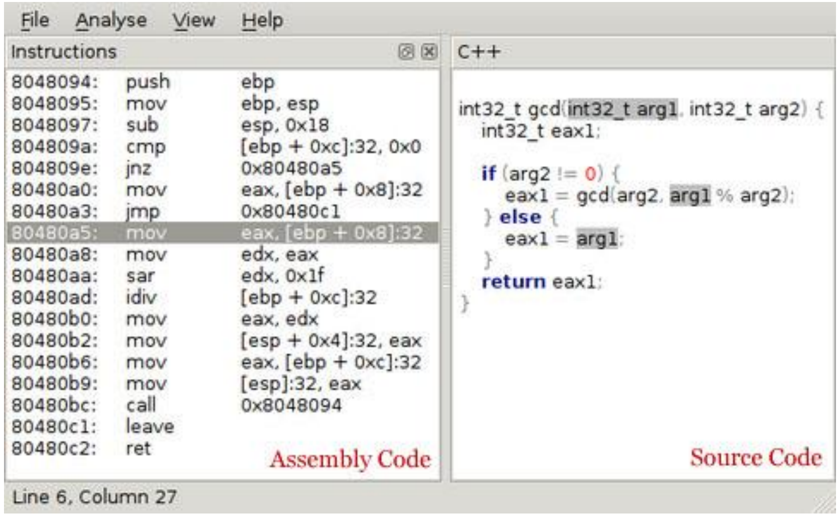


Almost impossible to optimize at system level

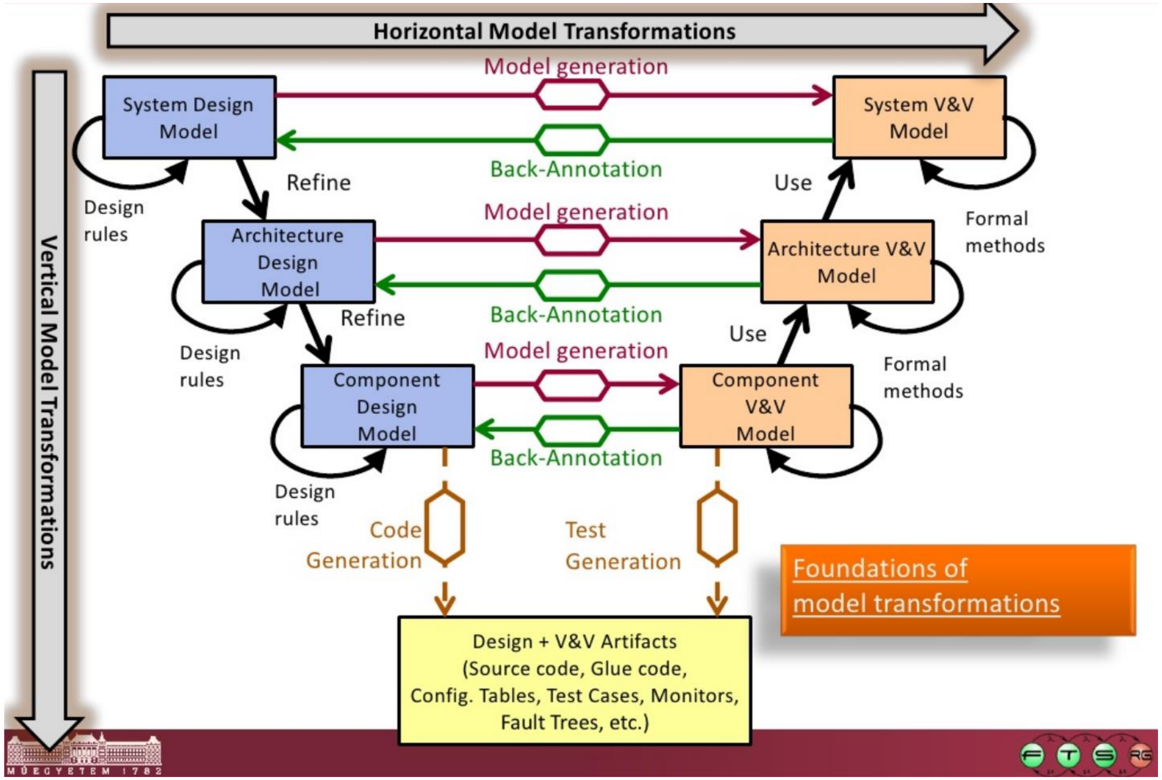
Approach 2: Model transformation

A **model transformation** is an automated way of modifying and creating models.

(Best) Example: Compilers



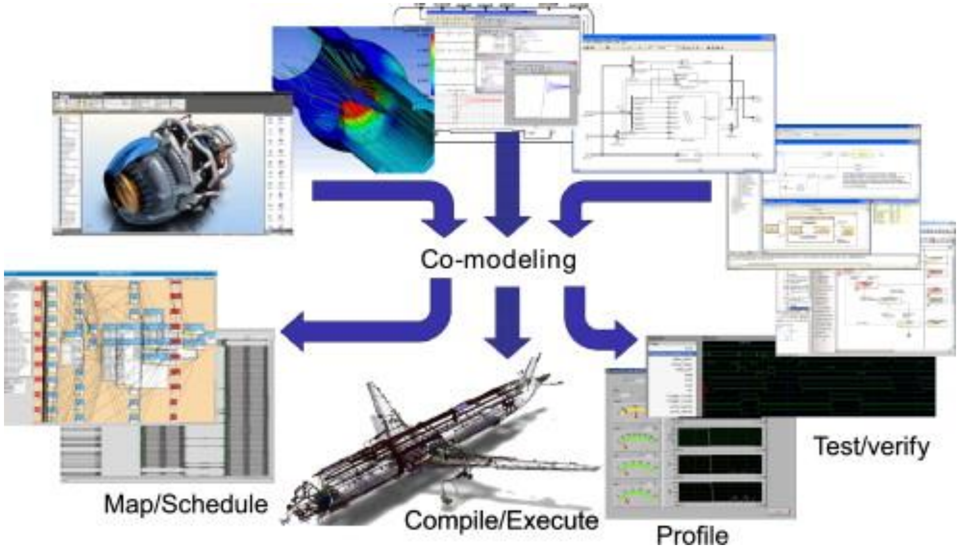
Model transformation and the design process



Source: Daniel Varro, CSMR2012

Approach 3: Multi-aspect modeling

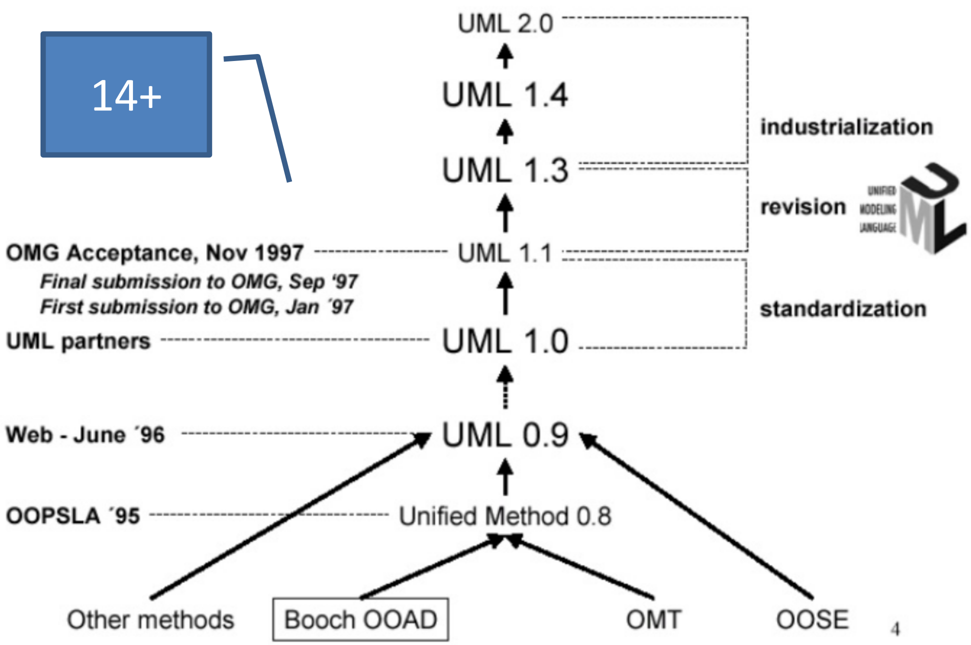
A *system aspect*, or *system view*, is a way to look at or describe a system as a whole. Each system aspect has its own associated semantic domain and can provide
an exhaustive description of the system, but only from that particular point of view.



System-level Modeling

Examples

14+



Methods war!

Source: Emertxe Ltd

12

AADL

5

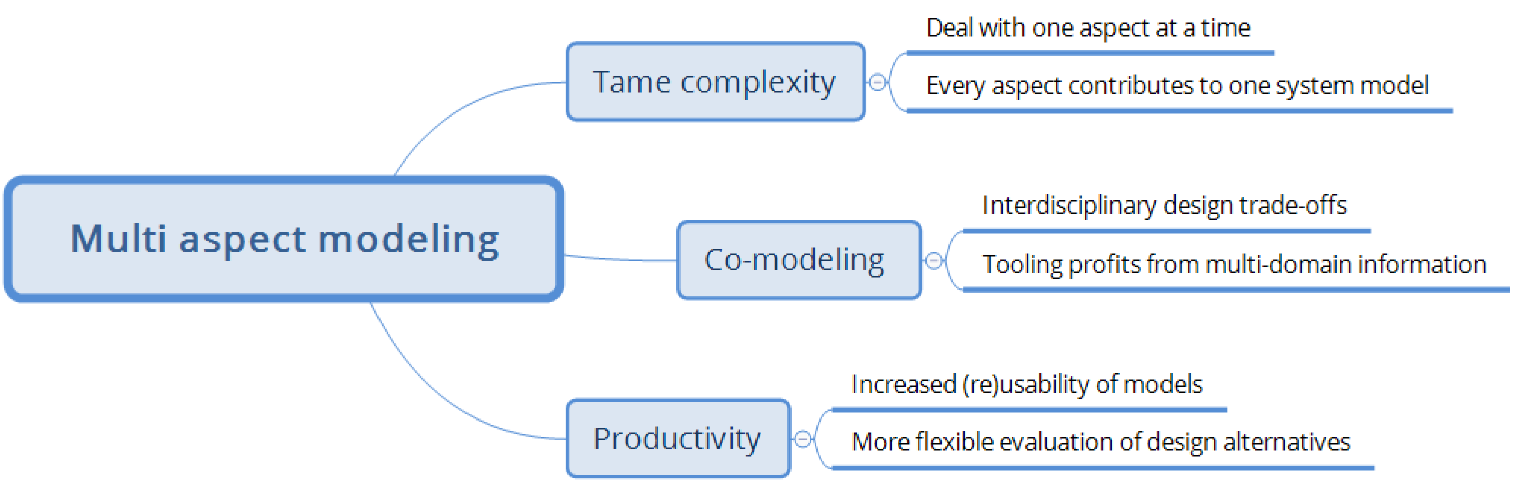
ISO IEC ITU RM-ODP

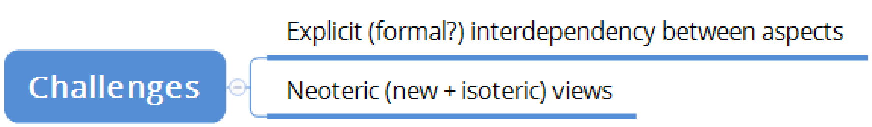
9

OMG SYSTEMS MODELING LANGUAGE

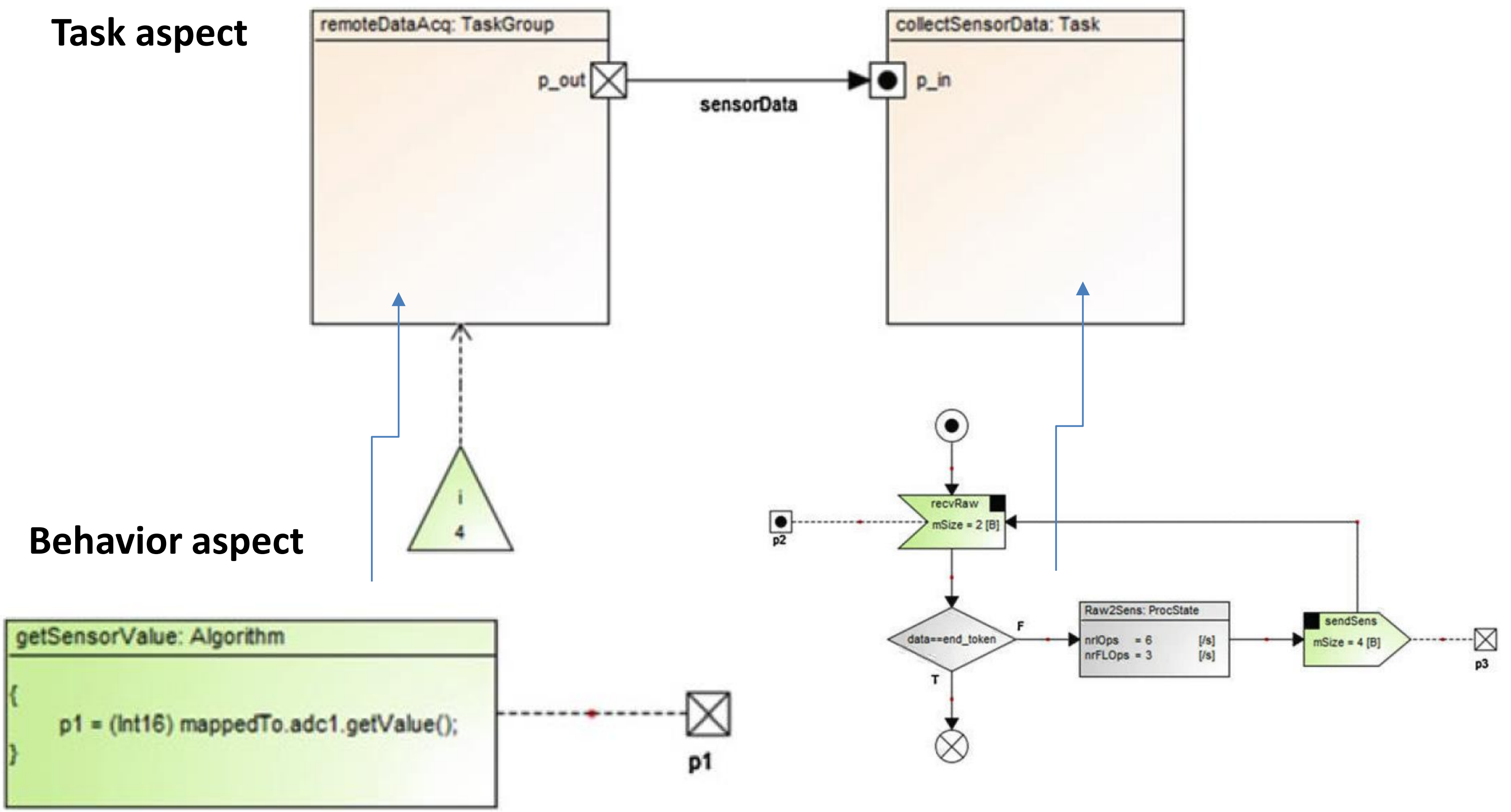
UML MARTE

Advantages of multi-aspect modeling



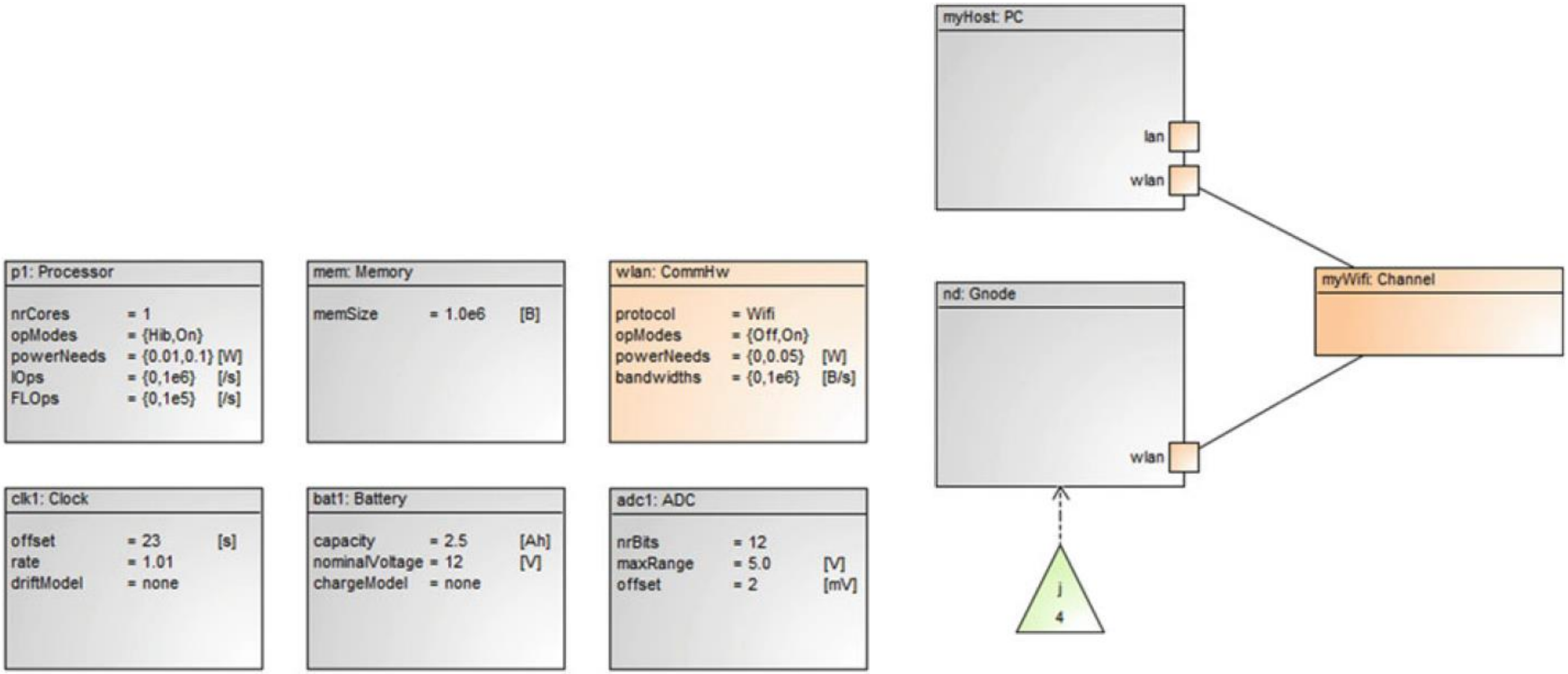


An example from CERBERO 1/3



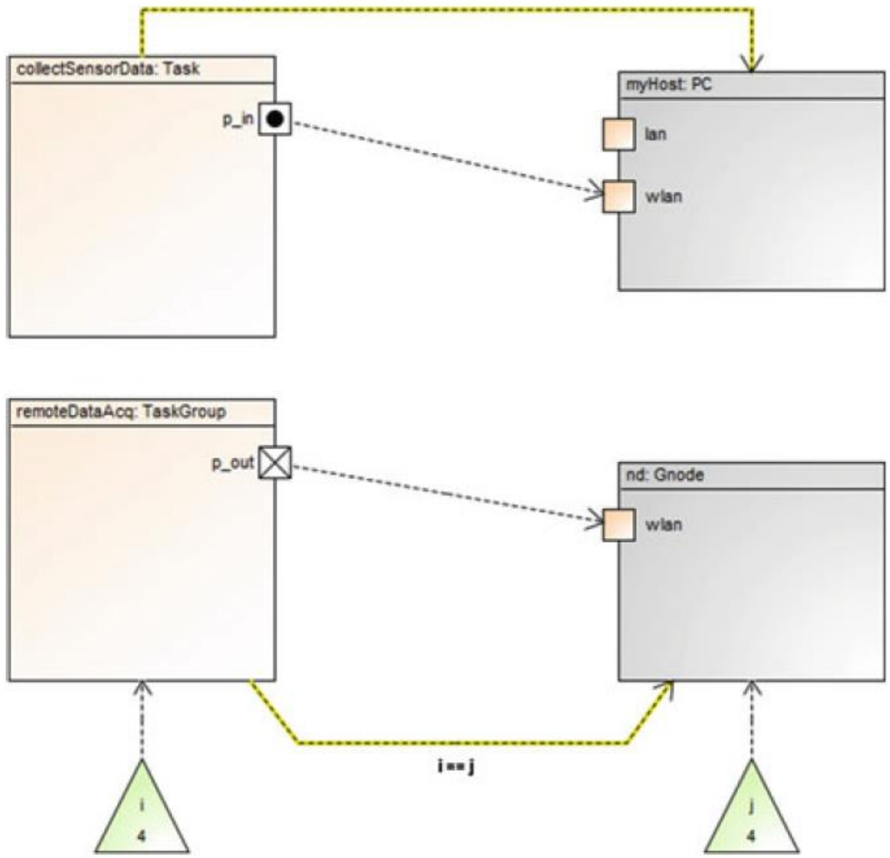
An example from CERBERO 2/3

Physical aspect

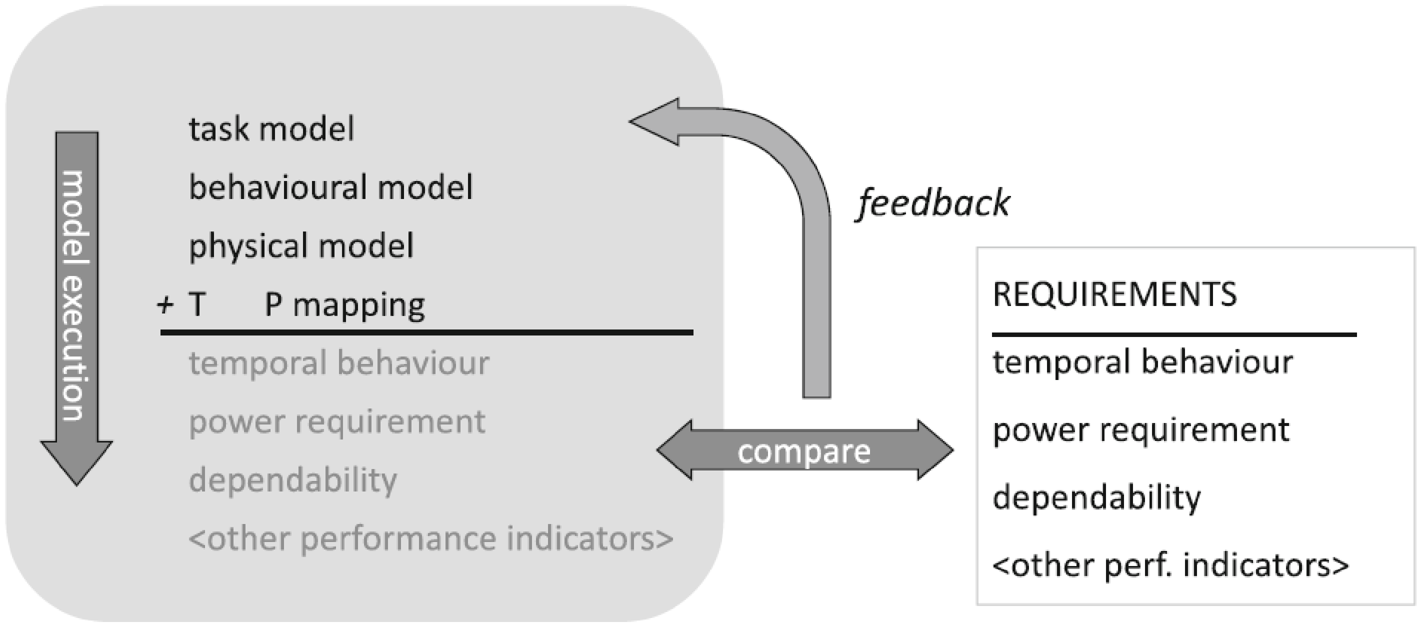


An example from CERBERO 3/3

Mapping view



Using the models together to assess KPIs



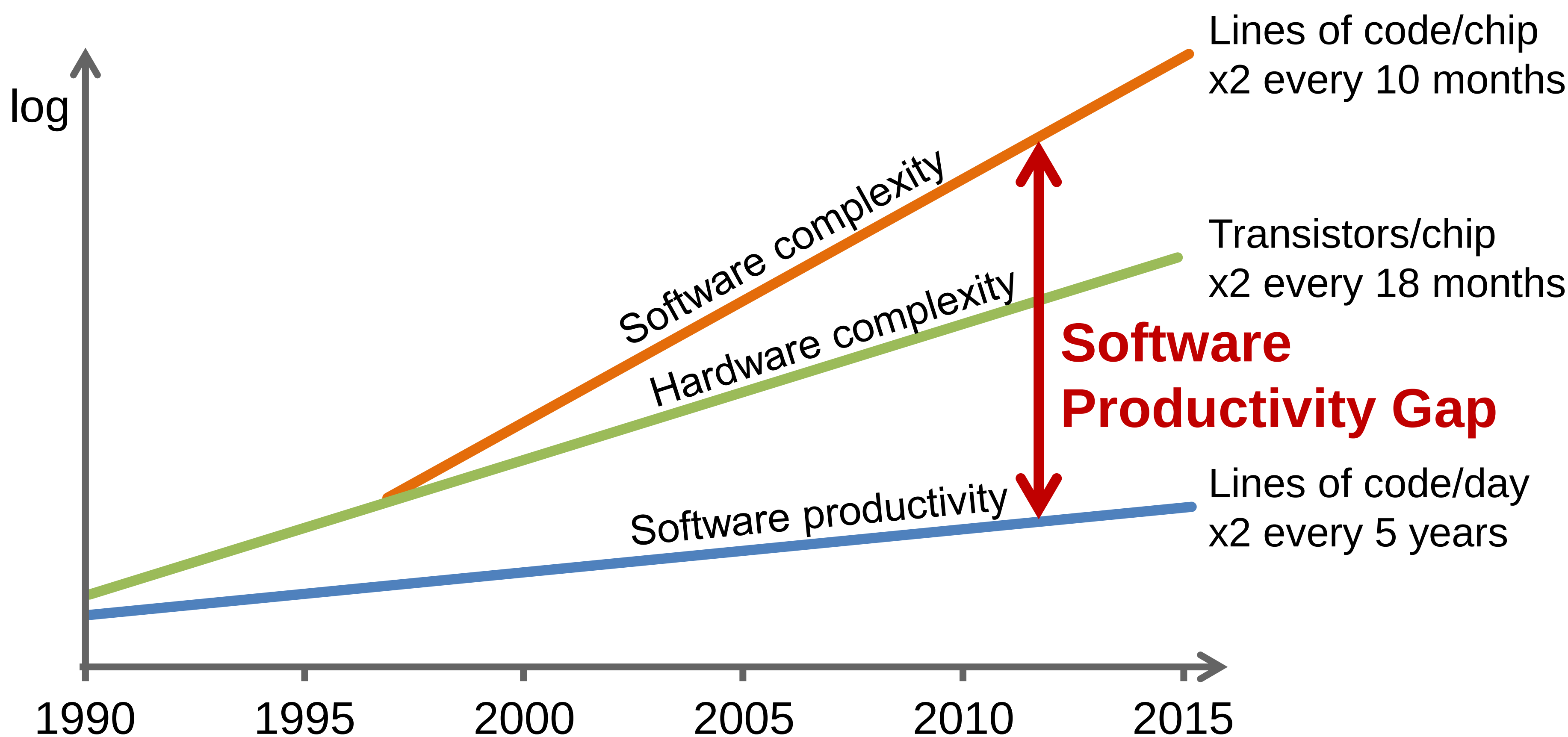
Some CERBERO contributions (expected)

- Challenges
  - Explicit (formal?) interdependency between aspects
    - Research: Aspects can also be used to formalize the dependency between other aspects.
    - Research: A (semantic) intermediary representation layer as connection point between modeling aspects (modeling languages)
  - Neoteric (new + isoteric) views
    - Research: How to model reconfigurability ? Adaptivity ? Scalability and Cardinality ?
    - Research: How to model key performance indicators in a more formal way ?

Component level
SW/HW (co-)design

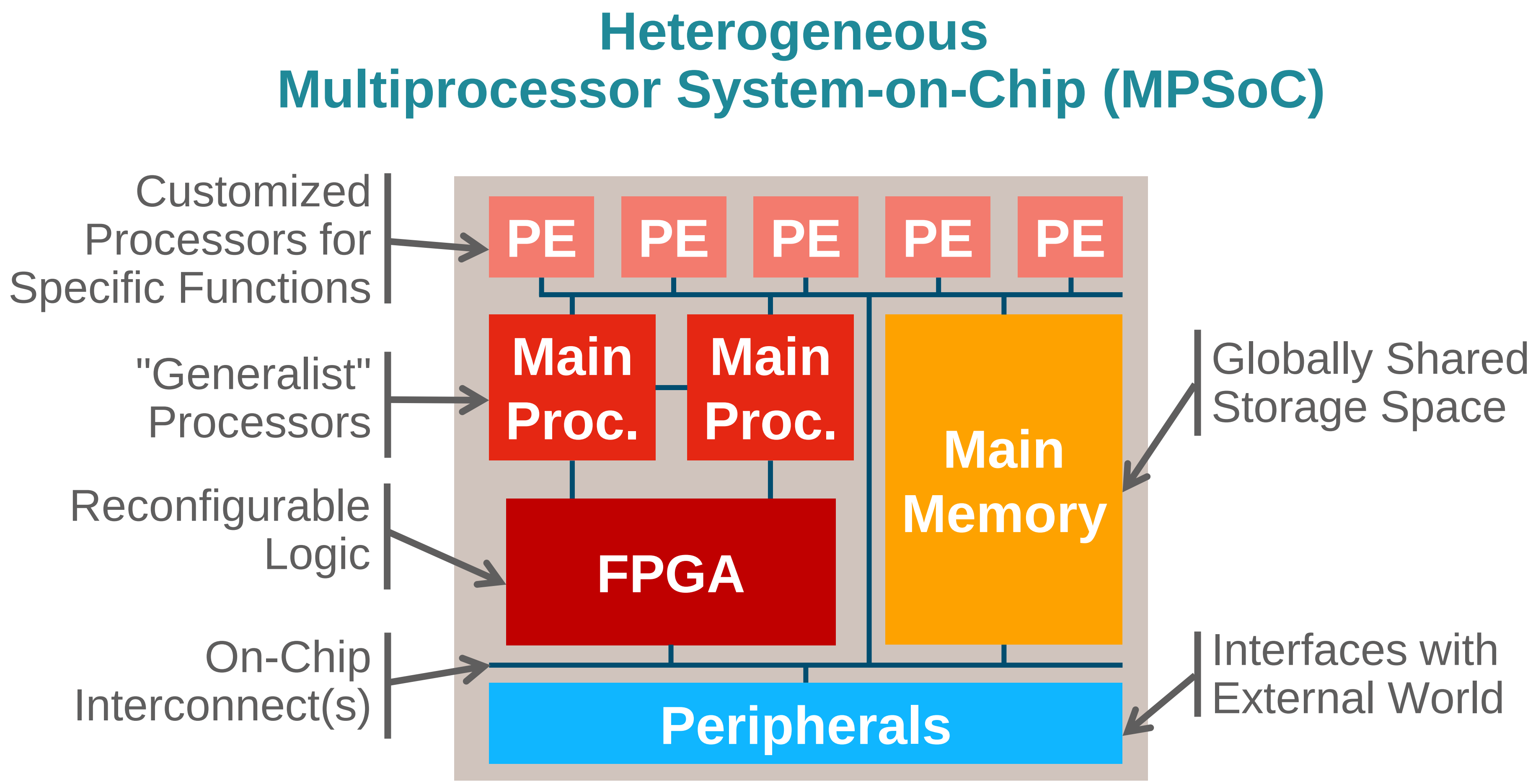
1. State-of-the-Art
2. Models of Computation
3. Models of Architecture

Need for a new HW/SW design approach!

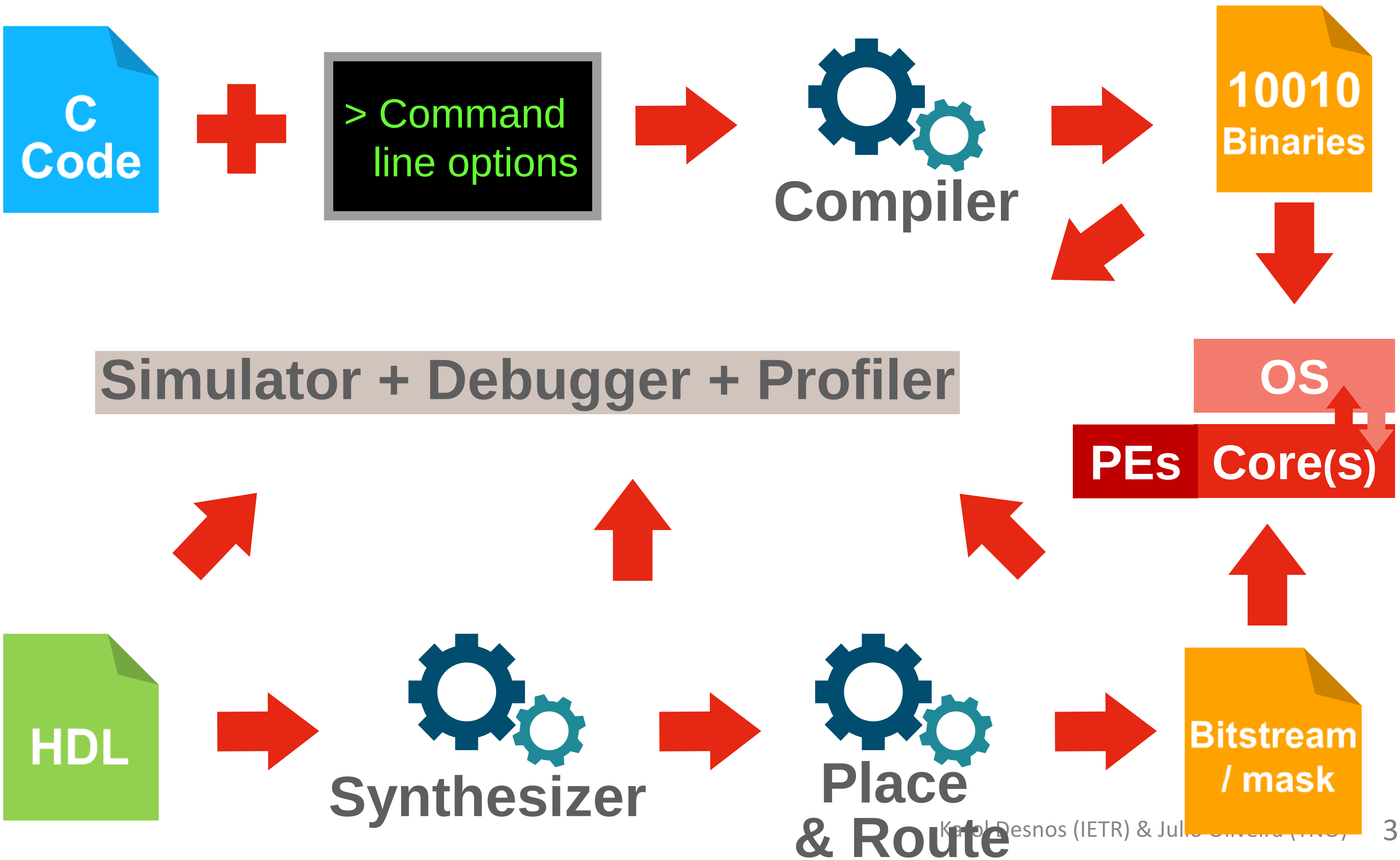


Source: ITRS & *Hardware-dependent Software*, Ecker et al., Springer

Typical HW/SW component



Typical development flow



Horizon 2020
European Union funding
for Research & Innovation

C Language is:

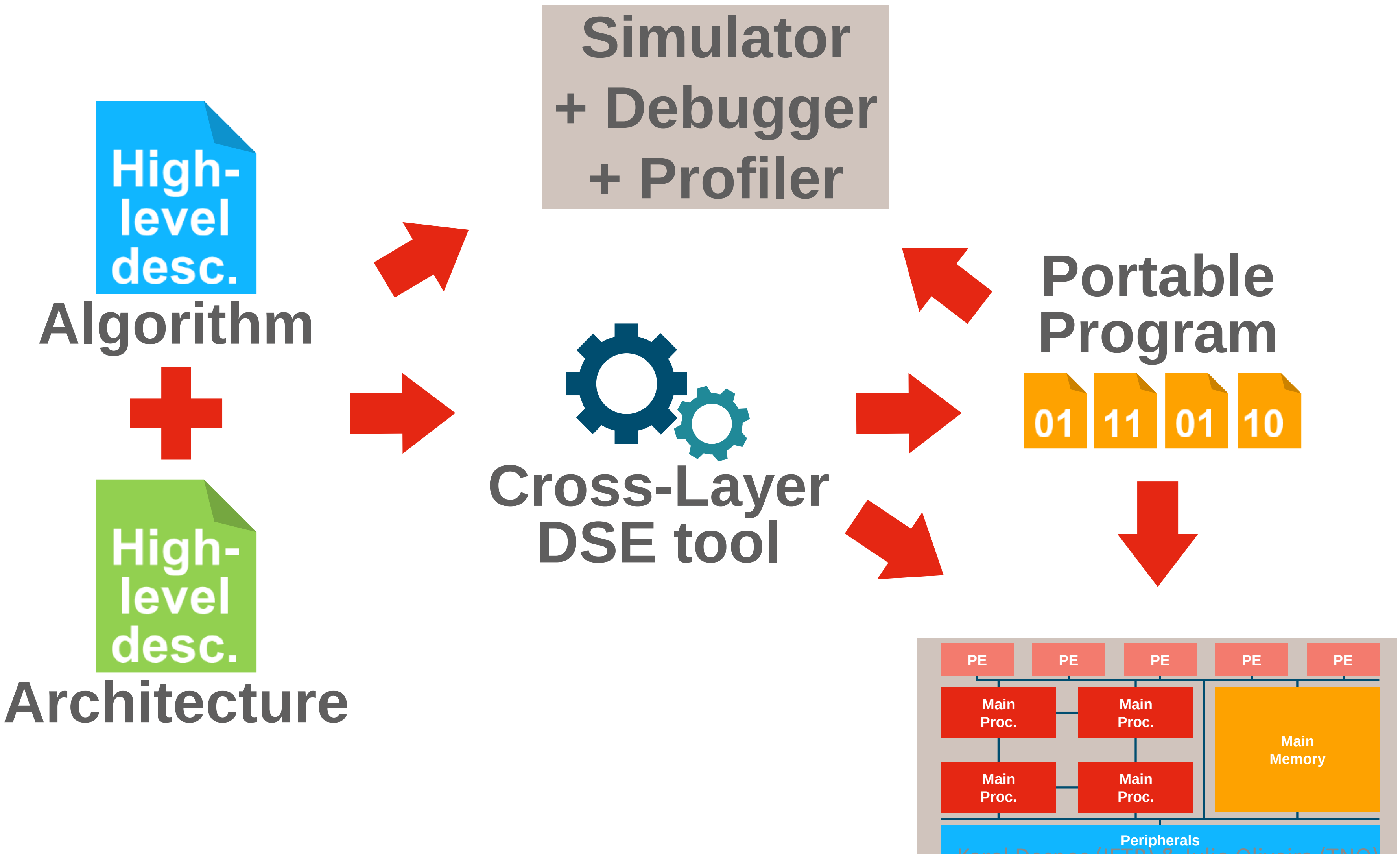
- Good for abstracting core architecture
  - Amount of registers
  - Number of pipeline stages
  - Instruction parallelism
- Bad for expressing coarse-grain parallelism
  - Inspired by Turing Machine
  - Global state in a program

VHDL/Verilog Languages are:

- Good for abstracting
  - Transistors
  - Analog concerns (signal propagation time)
- Bad for abstracting
  - Software concerns
  - … (more reasons in HLS and HW courses)

Horizon 2020
European Union funding
for Research & Innovation

What we want?



Component level
SW/HW (co-)design

1. State-of-the-Art
2. **Models of Computation**
3. Models of Architecture

source: J.E. Savage. Models of computation. Addison-Wesley Reading, MA, 1998

Model of Computation (MoC)

a.k.a. programming paradigm

Definition:

- A set of operational elements that can be composed to describe the behavior of an application.

→ **Semantics of the MoC**

Objective:

- Specify implementation-independent system behavior.
- Ease specification, implementation, verification of system properties.

How:

- MoCs act as the interface between computer science & mathematical domain.

⚠ A MoC is not a language ⚠

Language

Definition:

- A set of textual/graphical symbols that can be assembled respecting a well defined grammar to specify the behavior of a program

  → **Syntax of a the language**

Objective:

- Ease system description and maximize developer productivity.
- Be developer-friendly: readability, reusability, modularity, ...

How:

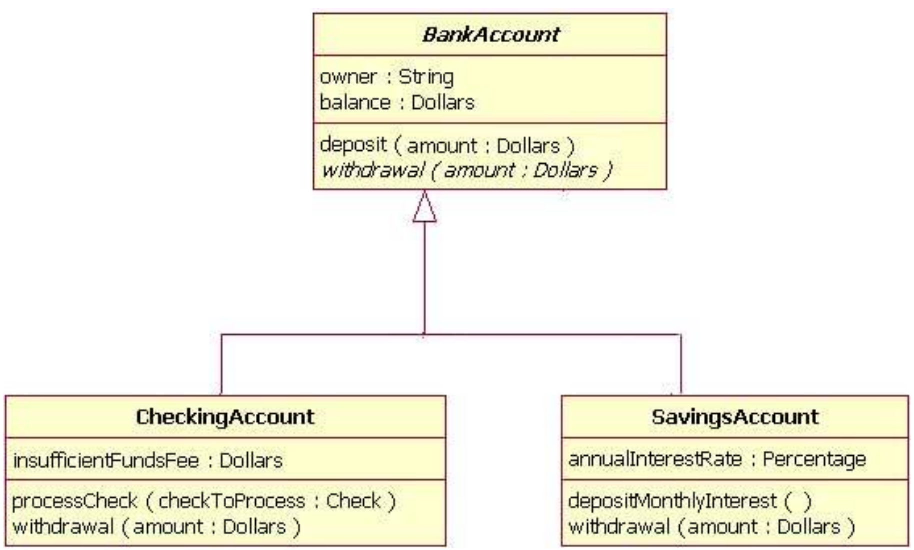
- Languages are the interface between the programmer & the Machine (through the complier).

A Language implements one or several MoCs

MoC Semantics and Language Syntax

- UML implements object-oriented semantics
- C++/Java implements object-oriented semantics
- They share semantics but not syntax

UML



Java

```
public class SavingsAccount
extends BankAccount {
  private int annualInterestRate;

  public void withdrawal(int v){
     …
  }
}
```

A few MoCs

Finite State Machine MoCs

Semantics

- States
- Transitions (possibly conditional)

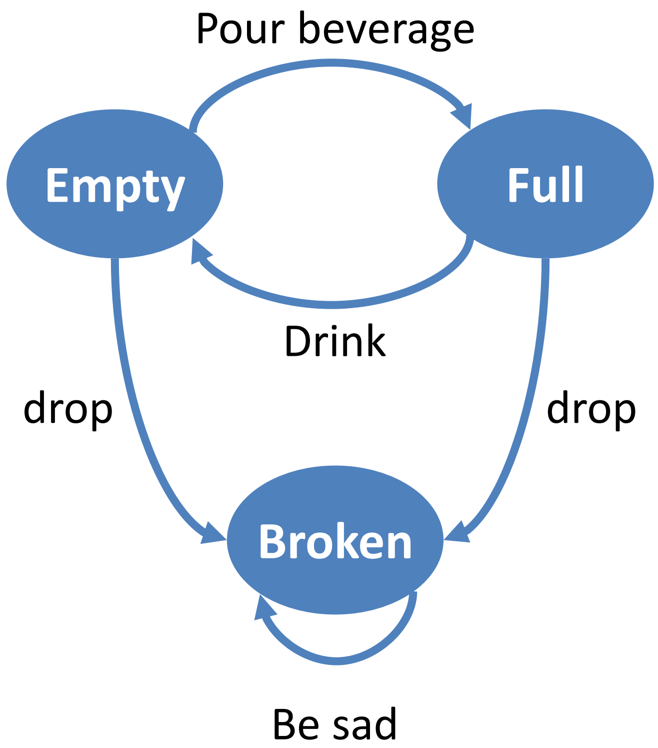
Used for

- Sequential logic
- System-level behavior
- Communication protocols
- ...

Property

- Non-deterministic, sequential

Glass FSM

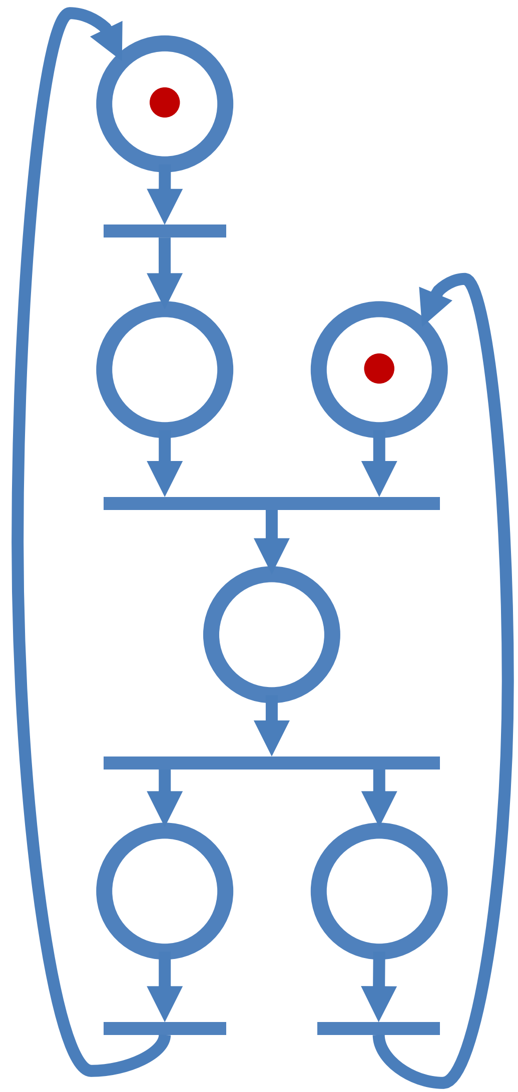


Horizon 2020
European Union funding
for Research & Innovation

Component level > MoCs

A few MoCs

Petri Nets



Semantics
- Places
- Transitions & Arcs

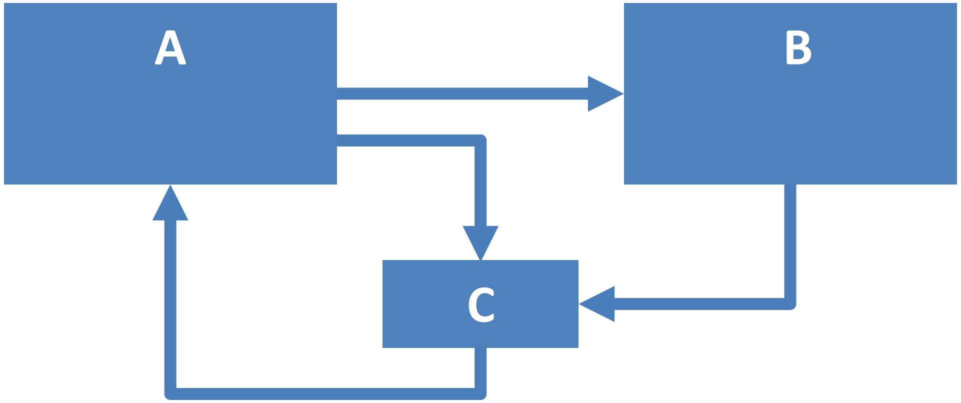
Used for
- Synchronization protocols
- Parallel computations
- …

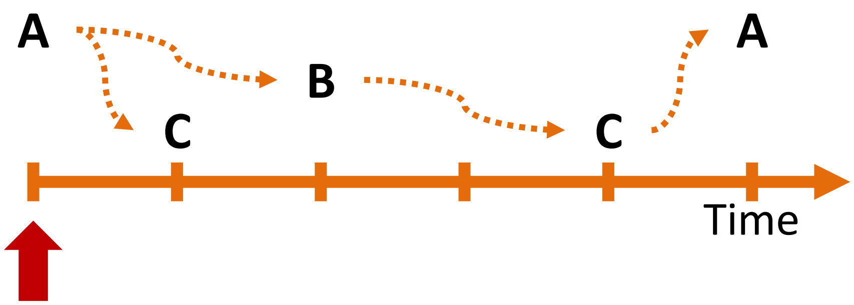
Property
- Parallelism
- Liveness, Boundedness, Reachability

A few MoCs

Discrete Event MoCs

Semantics
- Modules
- Signals
- Timed events
- Global clock





Used for
- Hardware Description
- “System” Simulation

Properties
- Timed, Non-deterministic *(if badly used)*

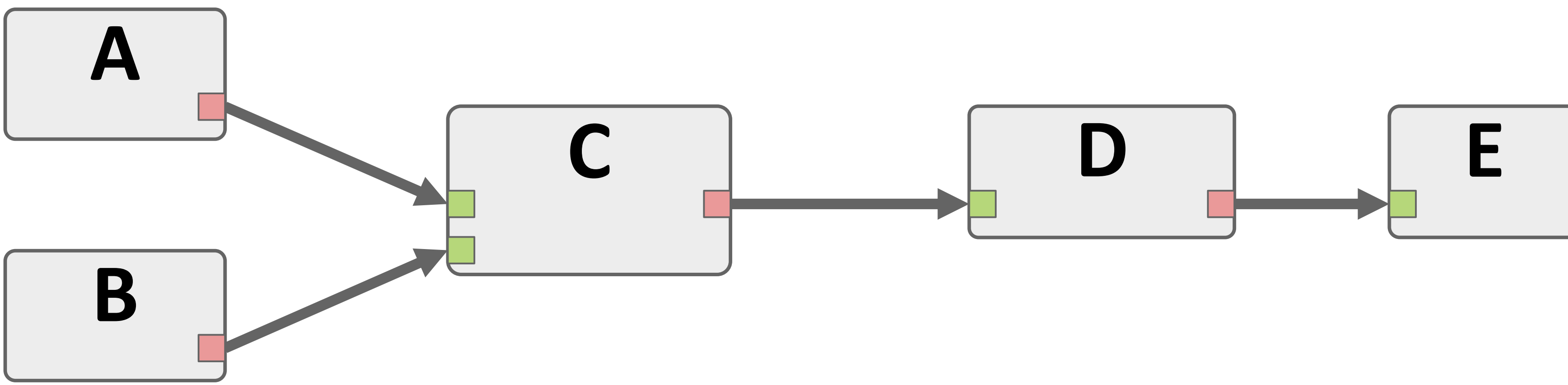
A few MoCs

Kahn Process Network

Semantics
- Actors & ports
- FIFO queues

Used for
- Parallel computations
- Stream processing

Properties
- Deterministic
- Untimed

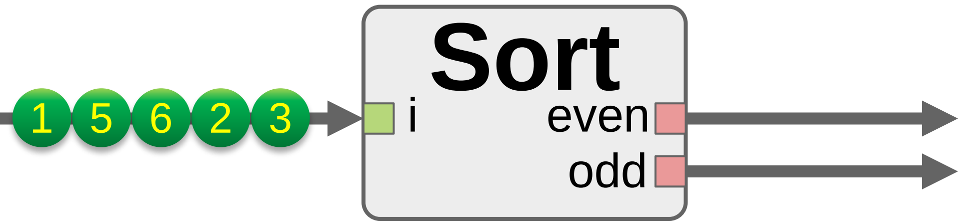


G. Kahn. The semantics of a simple language for parallel programming. IFIP', 1974.

A few MoCs

Kahn Process Network (KPN)

Determinism

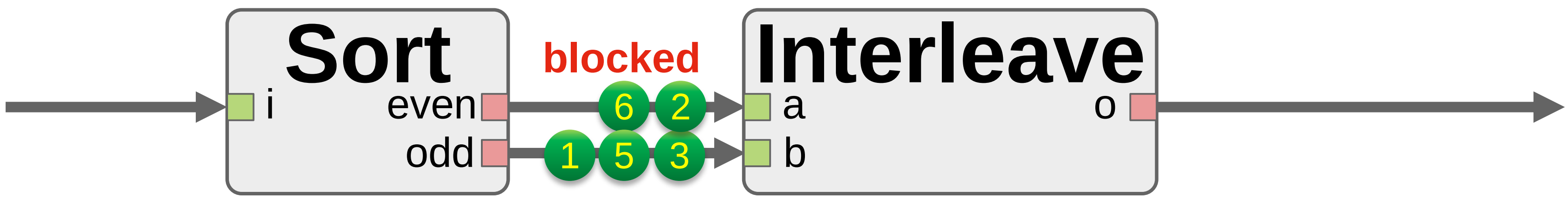



```
Process sort(in int i, out int even, out int odd) {
  int value = i.read(); // Blocking
  if( value % 2 == 0)
    even.write(value);
  else
    odd.write(value);
}
```

A few MoCs

Kahn Process Network (KPN)

Determinism




```
Process inteleave(in int a, in int b, out int o) {
   static bool = true;
   int value = (bool)? a.read() : b.read();
   bool = !bool;
   o.write(value);
}
```

A few MoCs

Dataflow Process Network (DPN)

Non-Determinism


```
Process inteleave(in int a, in int b, out int o) {
   static bool = true;
   int value = (bool)? a.read() : b.read();
   bool = !bool;
   if(no_timeout) o.write(value);
}
```

A few MoCs

Synchronous Dataflow

Semantics

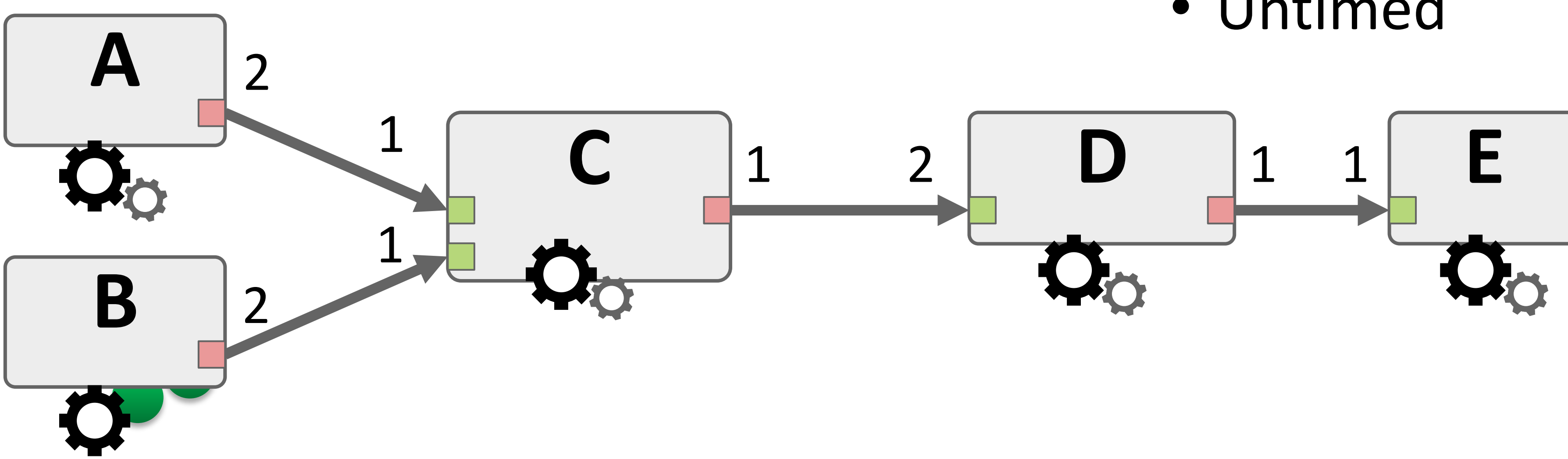
- Actors & ports
- FIFO queues

Used for

- Parallel computations
- Stream processing

Properties

- Liveness
- Boundedness
- Deterministic
- Untimed



Source: E. Lee and D. Messerschmitt, “Synchronous data flow”, Proceedings of the IEEE, 1987.

Horizon 2020
European Union funding
for Research & Innovation

Component level > MoCs

MoC Properties are important.

You need to know them to select the MoC suiting your needs

Feature	SDF	ADF	IBSDF	DSSF	PSDF	PiSDF	SADF	SPDF	DPN	KPN
Expressivity	Low				Med.		Turing			
Hierarchical			X	X	X	X				
Compositional			X	X		X				
Reconfigurable					X	X	X	X	X	X
Statically schedulable	X	X	X	X						
Decidable	X	X	X	X	(X)	(X)	X	(X)		
Variable rates		X			X	X	X	X	X	X
Non-determinism							X	X	X	

SDF: Synchronous Dataflow
ADF: Affine Dataflow
IBSDF: Interface-Based Dataflow
DSSF: Deterministic SDF with Shared Fifos
PSDF: Parameterized SDF

PiSDF Parameterized and Interfaced SDF
SADF: Scenario-Aware Dataflow
SPDF: Schedulable Parametric Dataflow
DPN: Dataflow Process Network
KPN: Kahn Process Network

Component level
SW/HW (co-)design

1. State-of-the-Art
2. Models of Computation
3. Models of Architecture

Models of Architecture

Definition:

- Formal representation of the operational semantics of networks of functional blocks describing architectures.

Abstraction

- VHDL/Verilog
- SPIRIT IP-XACT
- SystemC TLM
- UML Marte
- AADL
- S-LAM

Source: Kienhuis, B., Deprettere, E. F., Van Der Wolf, P., & Vissers, K. A methodology to design programmable embedded systems. In Embedded processor design challenges. Springer, 2002.

Horizon 2020
European Union funding
for Research & Innovation

Component level > MoAs

Disclaimer:

The topic of MoA is not as extensively covered in the scientific literature/industrial tools.

Ideas presented in following slides are based on recent work by Pelcat et al.

Unfortunately, these ideas can not (yet) be considered as a globally accepted reference.

M. Pelcat *et al.*, "Models of Architecture: Reproducible Efficiency Evaluation for Signal Processing Systems," IEEE *Workshop on Signal Processing Systems (SiPS)*, Dallas, TX, 2016, pp. 121-126.

Models of Architecture

Definition:

- Formal representation of the operational semantics of networks of functional blocks describing architectures

Yes, but

- The SDF MoC is a formal representation of the operational semantics of networks of functional blocks describing applications
- What if application = architecture?
- What if we do not want to model the architecture as a network?
- We need a more precise definition for MoAs!

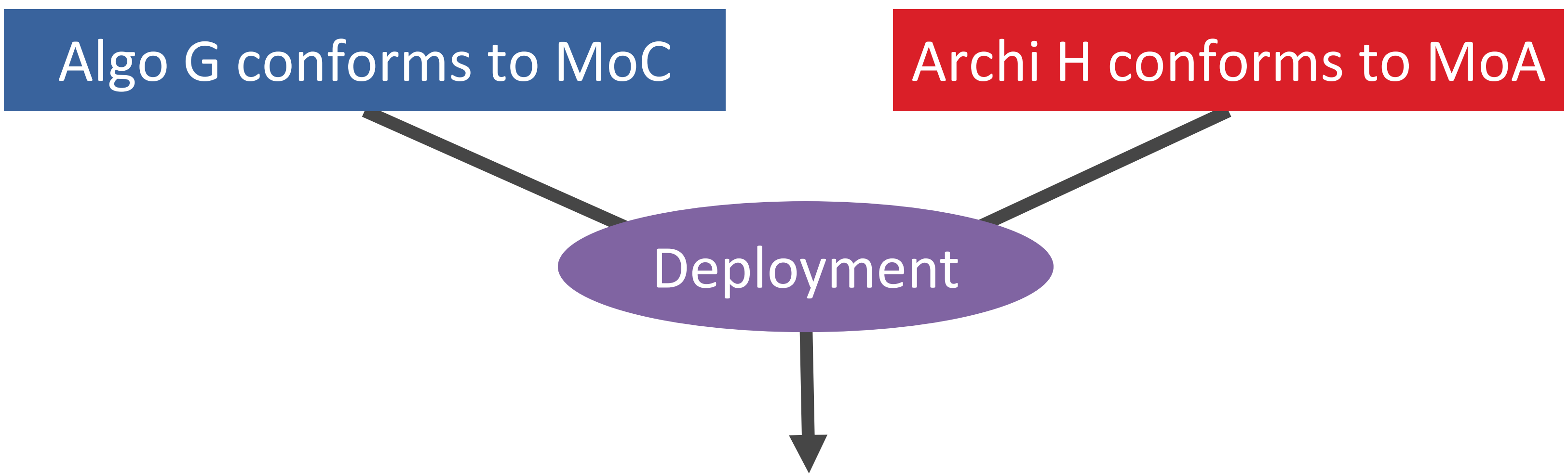
Horizon 2020
European Union funding
for Research & Innovation

Component level > MoAs

Models of Architecture

Definition

- an abstract efficiency model of a system architecture that provides a unique, reproducible cost computation, when processing an application described with a specified MoC.

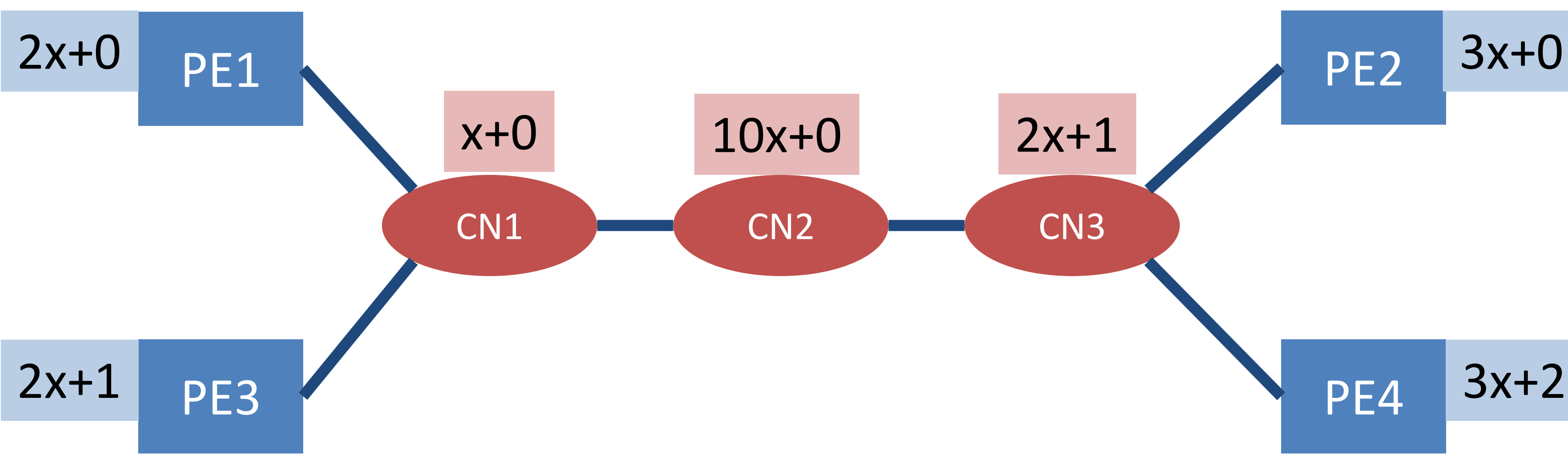


One and always the same performance number

LSLA MoA Example

Linear System Level Architecture (LSLA)

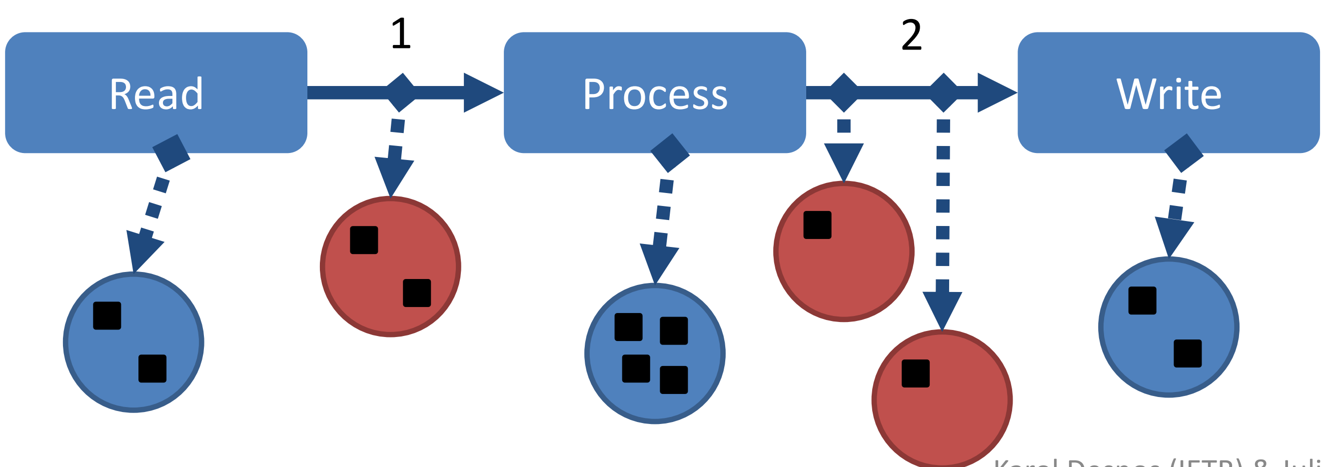
- A Model of Architecture
- Computing additive costs from application activity
- That can be used, for instance, to predict energy



LSLA MoA Example

Application Activity for cost computation

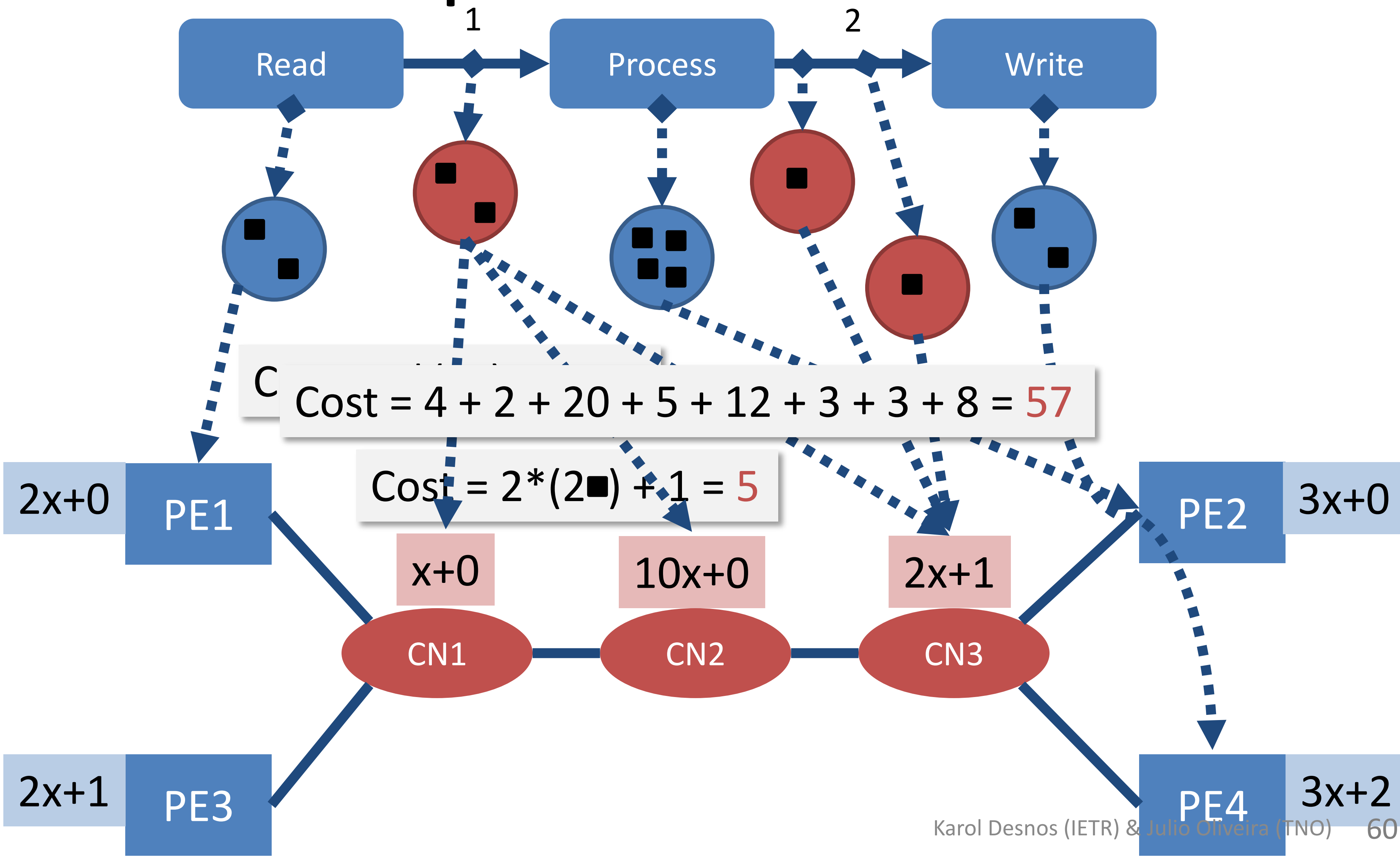
- AA is the amount of effort to execute an application
- From the MoC, we derive
  - processing and communication tokens (e.g. tasks and messages)
  - processing and communication quanta (e.g. cycles and Bytes)



Horizon 2020
European Union funding
for Research & Innovation

Component level > MoAs

LSLA MoA Example



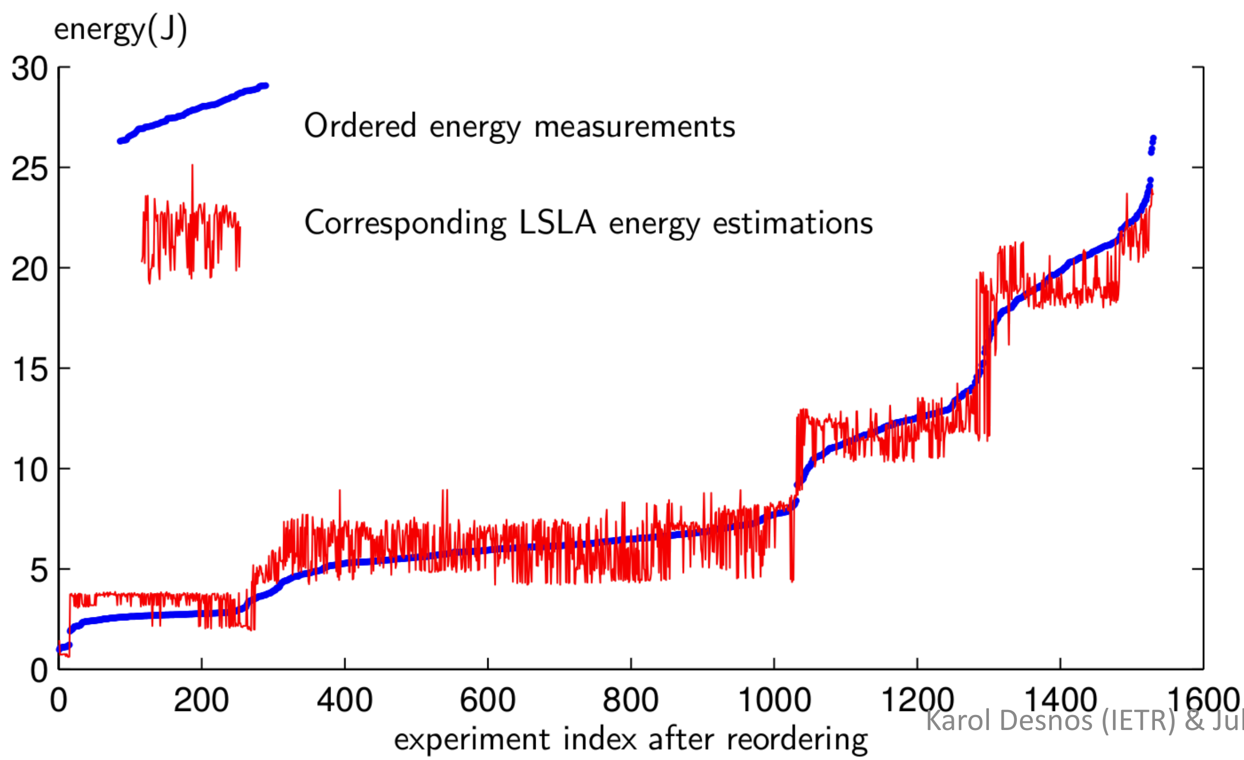
Horizon 2020
European Union funding
for Research & Innovation

Component level > MoAs

LSLA MoA example

Experiments

- On an ARM big LITTLE architecture
- Running a stereo matching application
- A simple LSLA reaches 85% of fidelity



Conclusion

Models

- offer various levels of abstraction for designing system-of-systems, system, SW and HW components
- are backed by mathematical models, providing a base for verification of system properties
- are based on semantics can be translated into actual implementation. (cf. next lecture)