Multi-Paradigm Modeling for Cyber-Physical Systems: Implications for Multidirectional Transformations and Synchronizations


Holger Giese
System Analysis & Modeling Group,
Hasso Plattner Institute, University of Potsdam, Germany
holger.giese@hpi.uni-potsdam.de
1. Cyber-Physical Systems & Integration

[Networked] Cyber-Physical Systems

Smart Factory - E.g. Industry 4.0

Smart Logistic

Micro Grids

Internet of Things

Smart City

Ultra-Large-Scale Systems

Smart Home

E-Health

Ambient Assisted Living

COST Action IC1404 for Multi-Paradigm Modeling for Cyber-Physical Systems (MPM4CPS)

Challenge: Integrate Models of Computation

- Problem to integrate models within one layer as different models of computation are employed.
- Leaky abstractions are caused by lack of composability across system layers. Consequences:
  - intractable interactions
  - unpredictable system level behavior
  - full-system verification does not scale

Integration has to cover multiple layers and their paradigms
Each model $M_j$ is an abstract representation of a part or multiple parts of an existing or envisioned original used for a specific purpose.

**Benefit:** For purpose $j$, we replace the original $O$ by a suitable model $M_j$ that does not contain any irrelevant information (reduced complexity!)

**Drawback:** Does an original $O$ consistent with both models $M_1$ and $M_2$ really exist (consistency)? – simple existence is often not enough!
How to Handle Multiple Models?

Try for each purposes to find a model $M_j$ that replace the original $O$, does not contain any irrelevant information (reduced complexity!), and integrate the models systematically to establish consistency.

Key questions:
- How many models are helpful (tradeoff benefits vs. integration effort)?
- When and how is integration happen for these models?
The explicit composition brings together subsystems which have been developed in parallel. In the ideal case all relevant system or subsystem characteristics are captured during the decomposition and are guaranteed when doing the composition. However, often this is not the case. For example, when using separation of concerns several aspects are often not covered during decomposition but become relevant when doing the composition (potentially in a later development stage) or when the composition not only exhibits the characteristics of its components but also characteristics which are determined by the composition (sometimes called emergent). It is particularly relevant for the integration that all system requirements that have not been broken down into subsystems requirements are checked for the composition result. This includes that characteristics such as deadlocks which can often not be predicted when doing the decomposition have to be addressed when doing the composition. Therefore, depending on the question of which characteristics are compositional or not resp. which requirements have been broken down to local properties of the subsystems more or fewer characteristics of the composition have to be checked at composition time to ensure a proper integration.

The standard case for composition is that the individual constituent parts are simply combined by some generic form of composition (e.g., scheduling in the case of processes on an operating system). More advanced cases employ declarative constraints contained in the specification of the components to ensure that the composition behaves properly (e.g., scheduling with guaranteed deadline in case of processes on a real-time operating system).

The resulting interplay of decomposition and composition is depicted in Figure 2 (a). At a rather high level of abstract the system is decomposed into two or more subsystems that are developed in parallel. These subsystems, which are then further elaborated in parallel, are composed later on according to the decomposition done upfront.

Fundamental Techniques for Integration:

- Decomposition
- Abstraction
- Composition (a) composition
- Abstraction (b) abstraction
- Consistency (c) consistency
- Parallel-development
Level of Integration

- **Representation-level**: integration efforts only guarantee that a joint representation is reached
- **Syntax-level**: integration efforts lead to correct syntax
- **Semantics-level**: integration efforts lead to compatibility at the level of the semantics
3. CPSLab & Integration:

Methodology

Tool landscape

Hardware

MT/MiL

Simulation stage

RP

Prototyping stage

SiL

HiL

Pre-production stage

ST

Matlab/ Simulink/ Stateflow

Robotino®View

Robotino®SIM

TargetLink

SystemDesk

v-rep

2018 | Giese | MPM4CPS: Implications for Multidirectional Transformations and Synchronizations
- Vertical **enrichment** of functional models (consistency manually)
- Horizontal **integration** of functional and plant models
- Horizontal **integration** of multiple functional models, an architecture model, and a plant model
- Vertical **enrichment** of multiple functional models, an architecture model, and a plant model (to realize functions while meeting resource constraints)
Model in the Loop (MiL)

- Layer: Abstract Control Algorithm + Idealized Plant
- Domain: Control/Software + Physics
- Multi-Paradigm: Yes, if control is discrete
- Cyber-Physical system: Yes, as control is cyber world and plant is from the physical world
- Integration: Decomposition & Composition + parallel development; semantics-level
Scenario: Complex Horizontal Integration

- Horizontal combination of multiple functional models by the architecture via the generated software (integration by composition for functions, integration by abstraction for OS)
- Downwards propagation can be expected, but must be managed
- Upwards propagation is usually forbidden (suppressed)
- Horizontal propagation is therefore also forbidden (suppressed)
Scenario: More Complex Horizontal Integration

- Horizontal combination of multiple specific structures (Autosar: software; VHDL: hardware, Matlab/Modelica: plant) via a generic structure (SysML)
- Downwards propagation can be expected, but must be managed
- Upwards propagation is usually forbidden (suppressed)
- Horizontal propagation is therefore also forbidden (suppressed)
Scenario: More Complex Horizontal Integration

- **Vertical decomposition** via a generic system structure (SysML) containing multiple specific structures (Matlab: control; Autosar: software; VHDL: hardware, Matlab/Modelica: plant; ...)

- Consistency between models and in the models interact, which may lead to transitive propagation/conflicts
Observations:

- A horizontal composition is often mainly done to establish consistency at the semantics-level to ensure that the different models fit together ("virtual integration"). Keep syntax-level consistency throughout the development for a horizontal composition of $n$ models (a multidirectional transformation or synchronization) is not really an issue.

Implications:

- We can help as semantics-level checks for the horizontal composition of $n$ models requires syntax-level consistency as prerequisite!
Needs for Integration

Observations:
- Often propagation between multiple models (multidirectional transformation or synchronization) was not wanted/permitted.

Implications:
- Need for concepts to manage permission to do only changes as permitted (interfaces?)
- To unleash the full potential of multidirectional transformation or synchronization we have to study the context (processes, activities, ... = mega models / paradigm) and identify how processes and activities can be improved.
Needs for Integration

Observations:
- The overlapping parts are linked to not overlapping parts and therefore **conflicts** may also result w.r.t. not overlapping parts.

Implications:
- For the overlapping parts we cannot expect to achieve more than has been achieved for the **merging of multiple versions** and also related finding may be relevant to us (limits for merging, living with inconsistencies, ...). => **semantics-level** likely not feasible
5. Conclusion & Outlook

- **Multiple models** and their integration is the heart of the matter for developing complex systems.

- **In case of cyber-physical systems** it holds:
  - models employ **different paradigms** specific for their layer
  - Integration of the models is of **paramount importance**

- **Current integration challenges:**
  - Build cost-effectively tools to integrate the models at the **semantics-level** (not only syntax-level) for a “virtual integration” to also support analysis of emergent properties
  - **Multidirectional transformation and synchronization** may establish **syntax-level consistency** throughout the development to enable automated **semantics-level integration checks**

---

For Future MPM4CPS with self-adaptation we get:
- Runtime model sync.
- Executable Runtime Mega Models organizing the sync. and other model operations


Bibliography (2/3)


Bibliography (3/3)


