TiMoBD

Workshop on Time Analysis and Model-Based Design, from Functional Models to Distributed Deployments

Introduction

9:00

Can existing methodologies and standards ensure the level of time predictability that is needed by modern cyberphysical systems?

M. Di Natale, Scuola S. Anna

Session 1

10:00 – 12:00

Applying MDE to real-time embedded systems: technologies, standards and experiences
Sara Tucci

Towards the Integration of EAST-ADL and UPPAAL for Formal Verification of Embedded System Architectures
Tahir Naseer Qureshi, De-Jiu Chen, Magnus Persson, Martin Törngren,

Portable Real-Time Code from PTIDES Models
Patricia Derler, John Eidson, Edward A. Lee, Slobodan Matic, Christos Stergiou, Michael Zimmer

Contract-Based Reasoning for Component Systems with Complex Interactions
Susanne Graf, Roberto Passerone, Sophie Quinton,

Compositional timing analysis.
Oded Maler

Session 2

13:15 – 15:00

Extraction of End-to-end Timing Model from Component-Based Distributed Real-Time Embedded Systems
Saad Mubeen, Jukka Mäki-Turja, and Mikael Sjödin,

Rigorous Model based Timing Analysis of Automotive Embedded Systems
S. Ramesh

Design and Evaluation of Future Ethernet-based ECU Networks
J. Teich, M. Glass, S. Graf, F. Reimann

Model-based Design of Distributed Automotive Systems
Martin Lukasiewycz, Samarjit Chakraborty, Michael Glass, Juergen Teich,

Can existing methodologies and standards ensure the level of time predictability that is needed by modern cyberphysical systems?

M. Di Natale, Scuola S. Anna

Why this workshop?

- It is a time of opportunity (emergence of complex distributed systems and industrial adoption of model-based development)
- Real world problems today (more than ever) need a merger of competencies (or a revolutionary new view of building systems!)
- The idea: bring around a table experts on: system and software models, architectures (including OS and resource managers), formal methods and model checking (everybody talks about time analysis but with different models/concepts)

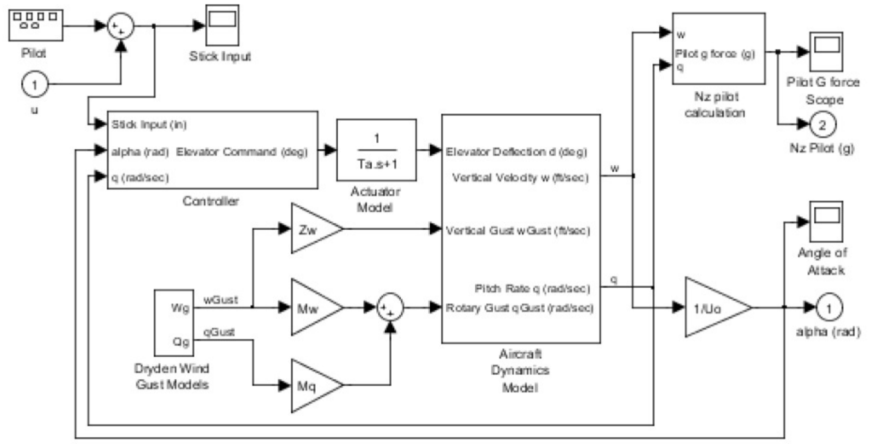
My perspective …

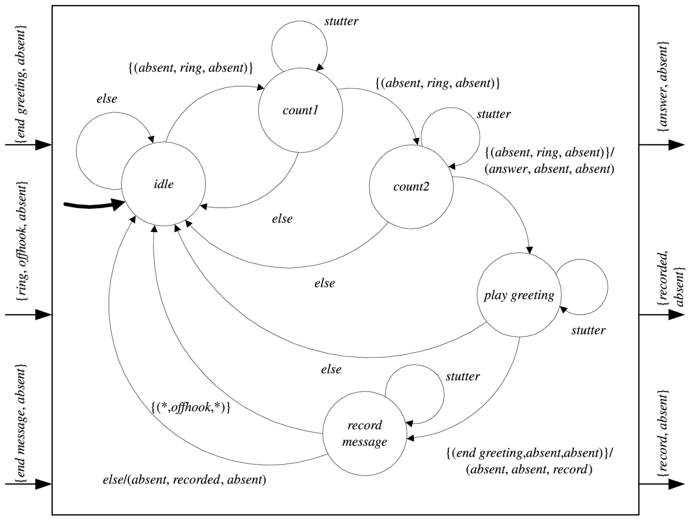
- I am from the schedulability analysis/time analysis community (almost an outsider… more on this later)
- Most of us are Operating Systems people, some with programming languages (Ada) background
- Our world consists of program functions, called in the context of threads (tasks) executed under the control of an OS
  - Non surprisingly, close to the AUTOSAR model
- A possible system model is:

  Given a set of tasks $T = \{\tau_1, \tau_2, \dots \tau_n\}$ each characterized by a model $\tau_i = \{C_i, T_i, D_i, p_i, E_i\}$, where C_i is the worst-case execution time, T_i its period, D_i its deadline, p_i its priority, and E_i the CPU on which it is allocated for execution …

Unfortunately, tasks are hardly the starting point

Where are the tasks?





NOTE: *stutter* = {(*absent*, *absent*, *absent*)}

- When we asked the industry why they did not apply our (worst-case) time analysis the common response was: “we have functional (correctness) problems and maintenance problems well before deadlines problems”
- However, tasks are definitely there …
  - Should the designer “see” them and control their creation? How?
  - Should they be the product of synthesis and optimization tools? What tools?
- And the same should be said for a complex (CPS) execution platform

Schedulability (real-time) analysis

- Predictability typically means that it is possible to compute the *worst-case response time* of a task without excessive pessimism
- Other communities have different goals/objectives/definitions when it comes to time constraints (the previous one is quite weak when modeling controls or safety-critical functions that cannot tolerate jitter)
- In the end the risk is to have separate communities, each of us with our hammer looking at a world consisting of (our type of) nails
  - Assuming our models this is what we can deliver …
  - How many models are needed to capture a modern complex CPS?
  - How good/realistic/capable of dealing with the required complexity are our models?

What happened to our timing analysis?

(from the real-time community)

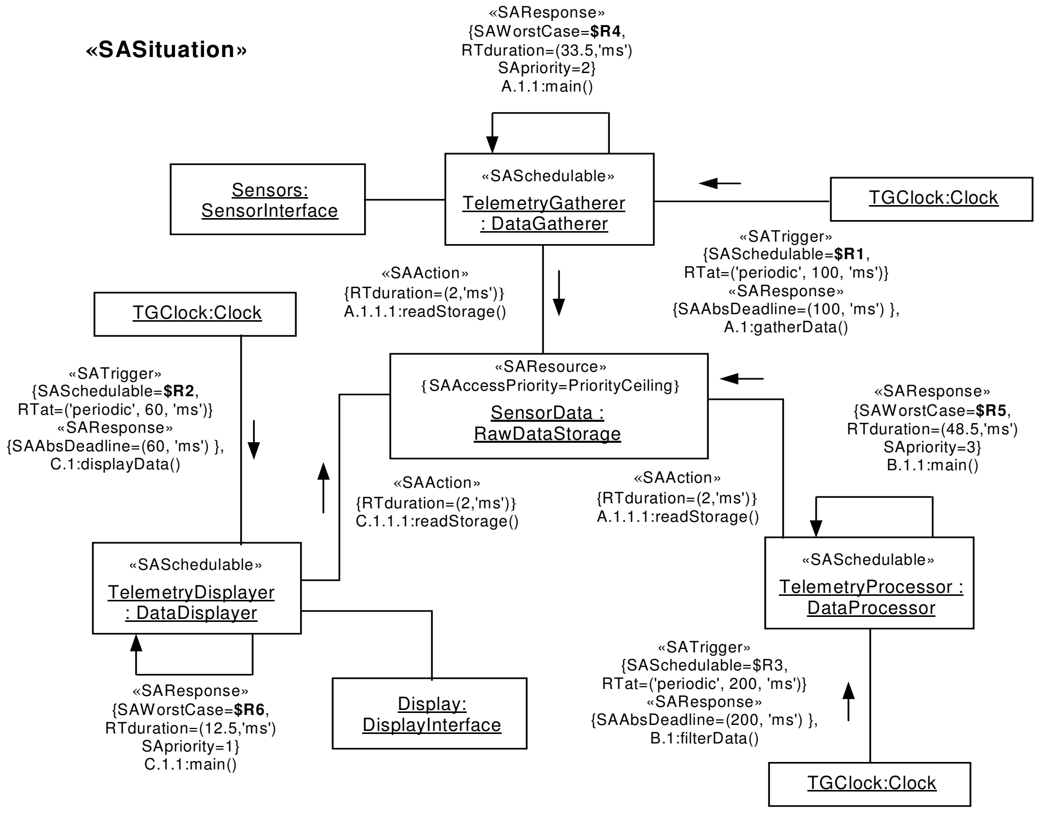
- We have been fairly successful in developing runtime algorithms for resource management that made into operating systems and communication protocols and improved their predictability (as in the previous definition)
- Examples:
  - Priority Inheritance (Mars Pathfinder)
  - The automotive OSEK standard
  - Influenced several other standards (CAN bus)

What happened to our timing analysis?

- But really, almost nobody really tries to predict the worst-case response time using our formulas
  - Not as much as you would expect
- Little use of design time analysis (until now) despite possible needs and several tools
- *Maybe the task model is not the right starting point …*
- *Maybe we needed a better integration between the analysis and mainstream design/modeling methodologie/languages*
- Starting from the late 90's there has been an attempt to bring the concepts of schedulability analysis into UML
- A neighboring domain … UML originated from the Object-oriented programming language/SW modeling communities
  - Possibly one branch of software engineering

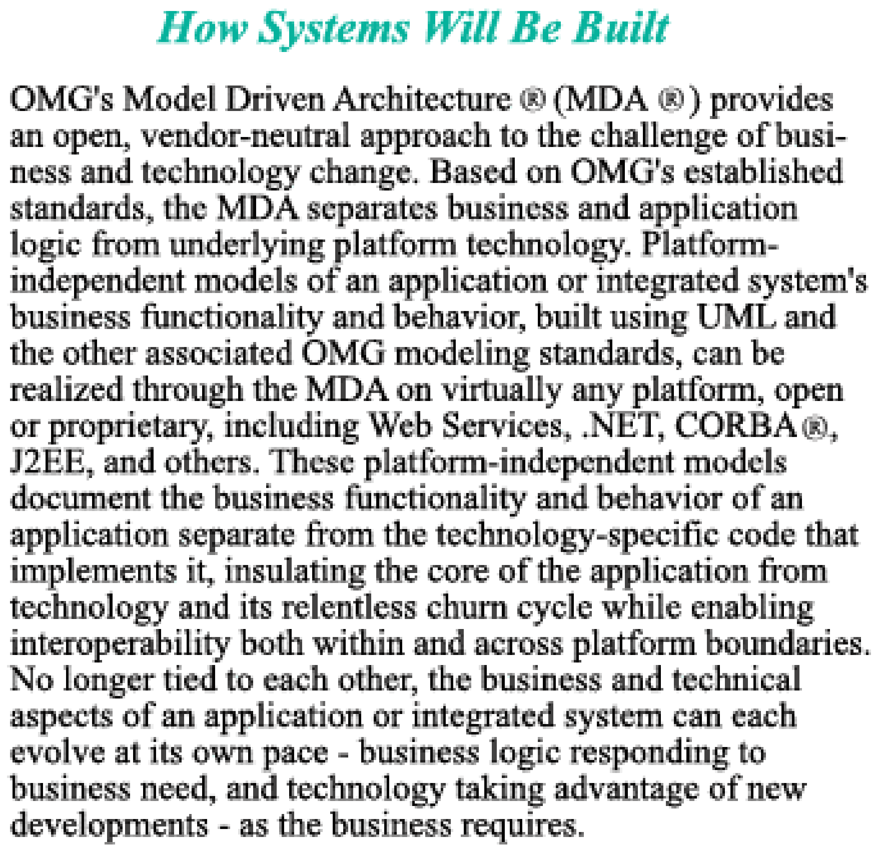
UML and then SysML and then Profiles and tools

- The Schedulability-Performance and Time profile (an extension/refinemement of UML) was defined in the late 90s by the OMG. It mostly failed …
- Later, it has been superseded by the MARTE profile
- Both are quite large (more than 600 pages)


«SASituation»
«SAResponse» {SAWorstCase=$R4, RTduration=(33.5,'ms') SApriority=2} A.1.1:main()
Sensors: SensorInterface
«SASchedulable» TelemetryGatherer : DataGatherer
TGClock:Clock
«SATrigger» {SASchedulable=$R1, RTat=('periodic', 100, 'ms')} «SAResponse» {SAAbsDeadline=(100, 'ms') }, A.1:gatherData()
«SAAction» {RTduration=(2,'ms')} A.1.1.1:readStorage()
TGClock:Clock
«SAResource» {SAAccessPriority=PriorityCeiling} SensorData : RawDataStorage
«SATrigger» {SASchedulable=$R2, RTat=('periodic', 60, 'ms')} «SAResponse» {SAAbsDeadline=(60, 'ms') }, C.1:displayData()
«SAResponse» {SAWorstCase=$R5, RTduration=(48.5,'ms') SApriority=3} B.1.1:main()
«SAAction» {RTduration=(2,'ms')} C.1.1.1:readStorage()
«SAAction» {RTduration=(2,'ms')} A.1.1.1:readStorage()
«SASchedulable» TelemetryDisplayer : DataDisplayer
«SASchedulable» TelemetryProcessor : DataProcessor
«SATrigger» {SASchedulable=$R3, RTat=('periodic', 200, 'ms')} «SAResponse» {SAAbsDeadline=(200, 'ms') }, B.1:filterData()
«SAResponse» {SAWorstCase=$R6, RTduration=(12.5,'ms') SApriority=1} C.1.1:main()
Display: DisplayInterface
TGClock:Clock

UML and then SysML and then Profiles and tools

- UML, SysML and possibly the MARTE profile are part of the MDE (Model-Driven Architecture) initiative by the OMG (http://www.omg.org/mda/)



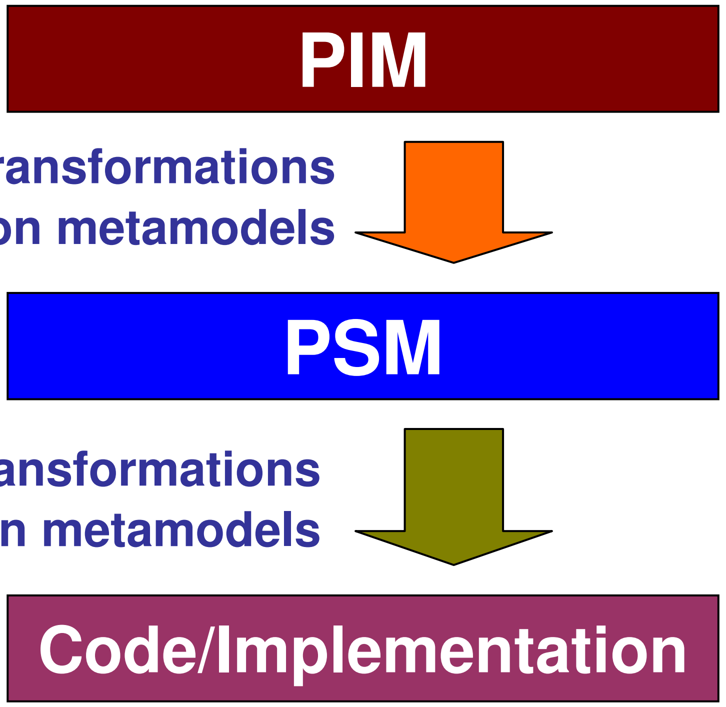
How Systems Will Be Built

OMG's Model Driven Architecture ® (MDA ®) provides an open, vendor-neutral approach to the challenge of business and technology change. Based on OMG's established standards, the MDA separates business and application logic from underlying platform technology. Platform-independent models of an application or integrated system's business functionality and behavior, built using UML and the other associated OMG modeling standards, can be realized through the MDA on virtually any platform, open or proprietary, including Web Services, .NET, CORBA ®, J2EE, and others. These platform-independent models document the business functionality and behavior of an application separate from the technology-specific code that implements it, insulating the core of the application from technology and its relentless churn cycle while enabling interoperability both within and across platform boundaries. No longer tied to each other, the business and technical aspects of an application or integrated system can each evolve at its own pace - business logic responding to business need, and technology taking advantage of new developments - as the business requires.

The good of MDA

- Recognize that the process is a 3-tier

Platform-Independent Model
Analysis of functional properties



Model-to-model (automatic) transformations
Based on metamodels

Platform-Specific Model
Analysis of architecture properties

Model-to-model (automatic) transformations
Based on metamodels

Availability of tools

- Metamodel-based approach (even if no MoC)
- Open source Eclipse/EMF
- Standard transformation languages and standards, open source transformation tools

(Some) issues with UML and MDA

Problems/Issues

- Lack of a formal semantics (or existence of a tool-dependent semantics only)
- Asynchronous models make verification of properties much harded
- OCL not widely used as a language for property specification
- Stereotypes for architecture modeling can easily be cumbersome (MARTE?), standardization of architecture models is difficult
- Not enough case studies (yet) for embedded systems
- Mapping of (PIM) functional models into (PSM) platform-dependent models representing tasks, messages, cores, networks …
  - What are the constraints on this mapping, how do you represent them ?

SysML

- UML was strongly oriented to the modeling of SW
- Insufficient to represent data-oriented communication (as in dataflow models)
- Insufficient for system-level modeling (modeling of signals, physical components, functional dependencies and constraints)
- SysML was defined to
  - How fit/mature is SysML for the representation of embedded systems?
  - Are there examples of application of SysML w. MDA on embedded systems with sufficient complexity?
  - Can MDA “borrow” from or itegrate with methodologies with a more formal MoC?
  - How about integration with other architecture languages or standards for component modeling ADL, AUTOSAR?

In the meantime, model-based design made it into the automotive/aeronautics practice

Tenets of MBD

(MBD vs MDA, that becomes a problem even when you write a workshop proposal!)

- Executable models (for simulation purposes)
- Models based on formal MoC (for analysis)
- Continuous V&V
- Automatic generation of implementation
- Most commercial solutions are based on synchronous reactive MoC: SCADE, Simulink

Model-based design came into practice

Advantages

- Available path to implementation
- Availability of tools for model verification of properties

Issues

- .. the available paths to implementation are de-facto for single-core platforms only
- Or, at most time-triggered platforms
- *Static scheduling of code or simple task models*
- How about large, complex CPS?
- Typically there is no way (in commercial tools) to represent the execution platform and the task and message models
- Large-scale CPS might require the platform modeling and the modeling of the "mapping" of functionality to platform where the platform can be LTTA (not necessarily TT)
- What are the limitations for the model verification?

… and model verification is becoming a need

- Certification of critical systems needs extensive coverage of decisions/conditions, as required by DO-178B
- Verification by testing will soon be impractical because of combinatorial complexity explosion
- Automated verification by theorem proving/model checking seems the only option
- Certified model verification requires demonstrated semantics preservation in the generation of refinements/implementations.
- What complexity is affordable today?
- Are asynchronous models out of the picture?