Design and Analysis of Trustworthy Embedded Systems: A Model Driven Approach

Software Technology Conference
Long Beach, CA

29 March 2014

Vahid Rajabian-Schwart

William McKeever

Integrity ★ Service ★ Excellence

Overview

- **Motivation and Problem Domain**
- **Background**
  - Model-based techniques
  - Security and real-time properties
- **Methodology**
  - Domain-specific model
  - Upfront (design-stage) analysis
  - Formal verification and deployment mapping
- **Summary**

Modern-Day Embedded System

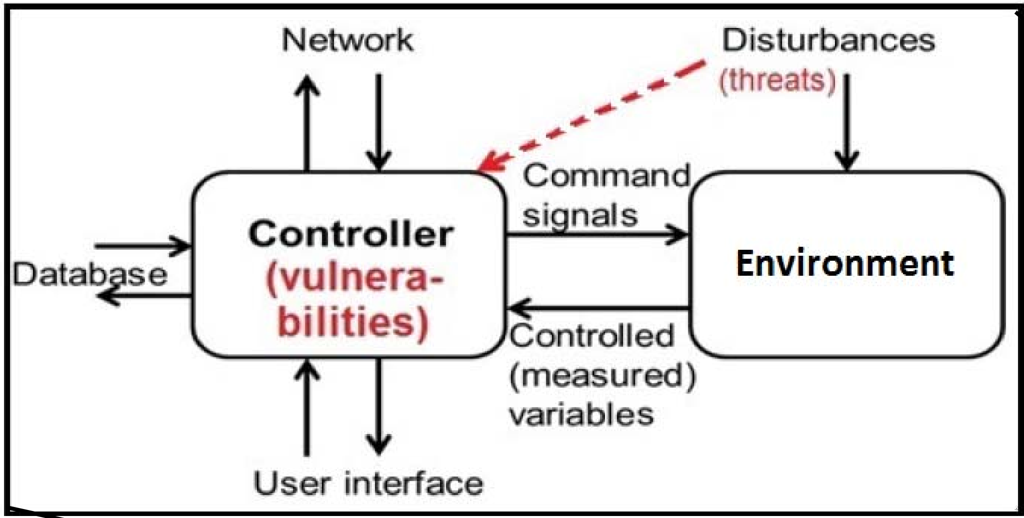


Figure derived from 1) Zalewski, Janusz. Modeling and Analysis of Trust Attributes in Software: Threat Modeling for Security Assessment in Cyberphysical Systems. In Summer 2012 VFRP Report, AFRL, August 2012. 2) Public Domain, 01/17/2014, http://commons.wikimedia.org/wiki/File:F-22_Raptor.JPG

Motivation and Problem Domain

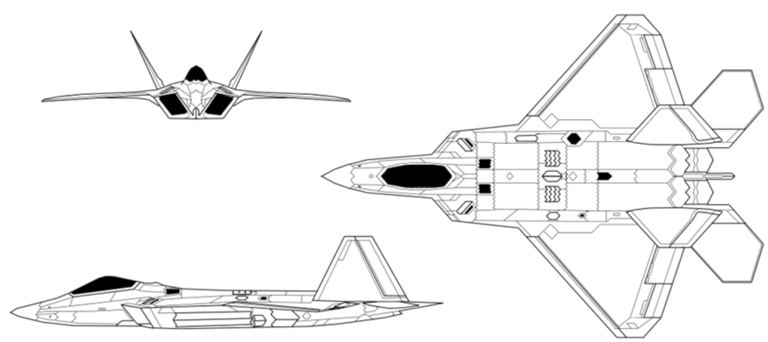
- **Rise in complexity of modern embedded systems**
  - Cross-domain (timing, power, etc.) factors influence design
  - Traditionally isolated devices have become network-enabled
  - Software implements majority of system functionality
  - System design must leverage third-party components
- **Lack of security (authentication, encryption, heterogeneity, etc.) in design**
  - Security-through-obscurity is not an option
  - Security as an afterthought leads to increased system fragility and schedule/cost overruns
  - Detection-focused security model does too little, too late
- **System real-time properties are directly affected by the addition of security mechanisms**

Objectives

- **Reduce system design complexity through model-driven development**
  - Increase component reuse, enable correct-by-construction models, formal verification
- **Integrate security early-on (design stage) in the development process**
  - Identify cross-domain relationships of security mechanisms and evaluate system trade-offs

Model-Driven Development (MDD)

- **A model is an *abstract* representation of a system**
  - Abstraction helps us deal with complexity
    - Machine Code → Assembly → C/C++
    - Transistor → logic gate → integrated circuit → ASIC
  - Effective tools enable us to build complex systems
- **MDD is concerned primarily with producing models as the output of the system design process**



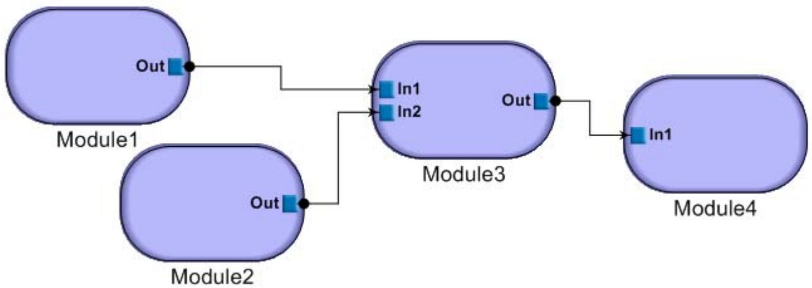
source: Zaku, http://commons.wikimedia.org/wiki/File:F22a3view.PNG, Creative Commons Attribution-Share Alike 2.0 Generic License, Accessed 01/17/2014.

Domain Specific Modeling (DSM)

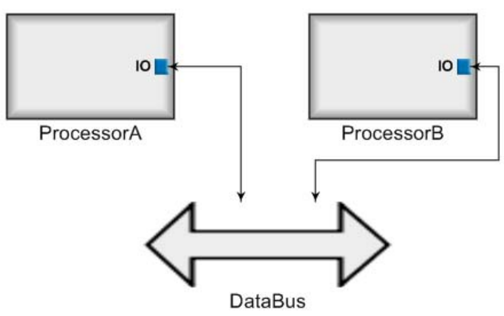
- **Raises abstraction level by narrowing application domain**
- **Uses concepts and rules from the application domain**
  - e.g. Matlab/Simulink for digital signal processing and LabVIEW for instrumentation
- **DSM for embedded systems**
  - Model how physical hardware and application software interact and are integrated to form complete systems
  - Industry standards
    - MARTE (Modeling and Analysis of Real-Time and Embedded Systems): A UML profile extension by OMG
    - AADL (Architecture Analysis and Design Language): A SAE standard

TESS Model

- **Trusted Embedded Software System (TESS)**
  - Software modules and their logical dependencies



  - Hardware components and their physical configuration

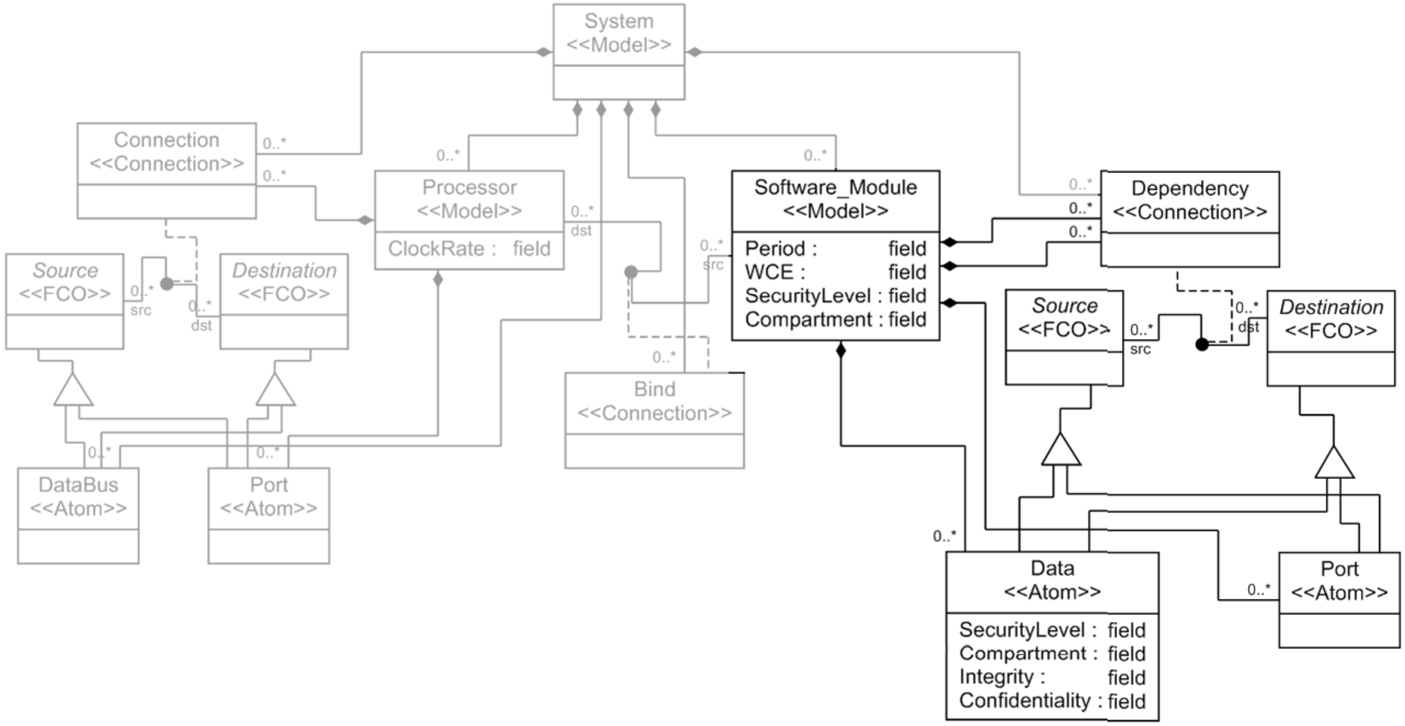


Meta-models

- **Used to define the modeling language of the domain application**
- **Make use of generic concepts that are abstract enough but common to most domains**
  - Provide fundamental objects and relationships in order to model a modeling language
  - A constraint language, such as OCL, is used to refine composability rules
- **Tools enable model definition and interpretation**
  - Generic Modeling Environment (GME)[4]
  - Eclipse Modeling Framework (EMF)[5]
  - Metaedit+[6]

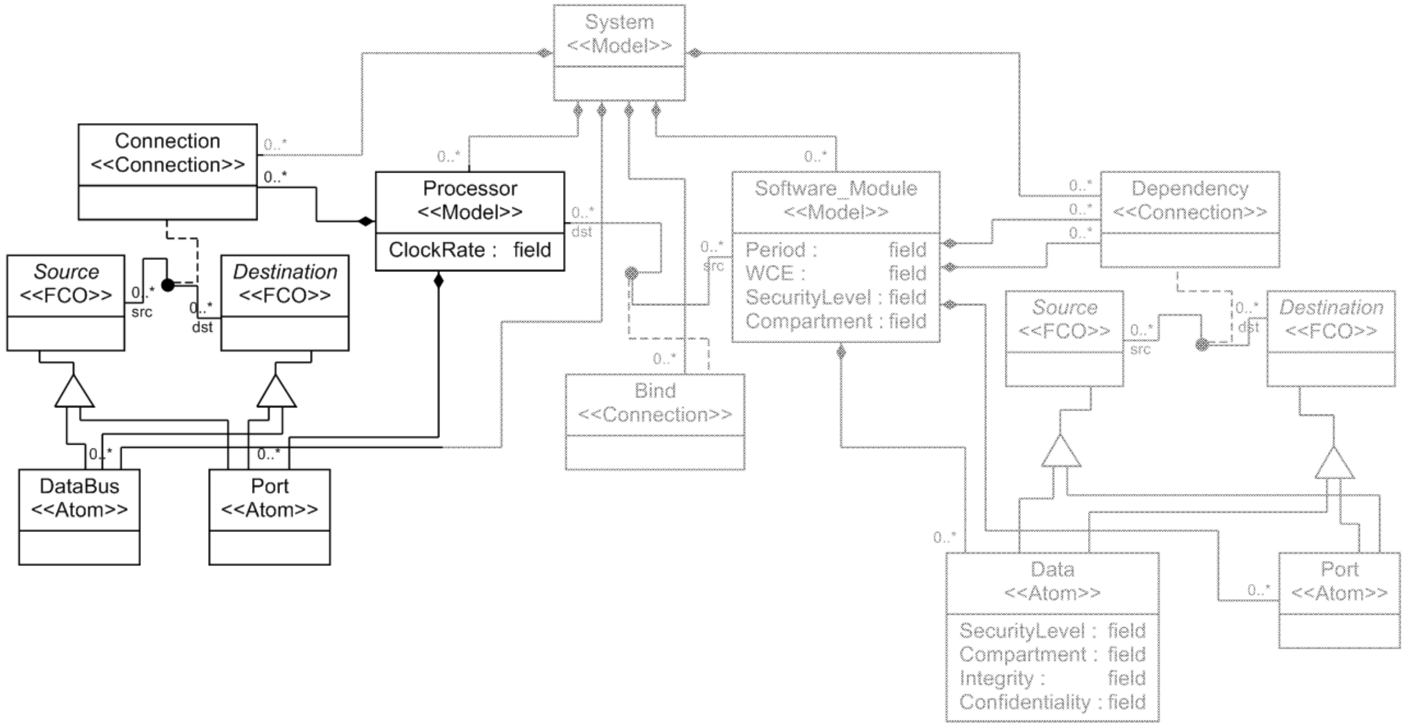
TESS Meta-model

- **Software modules**



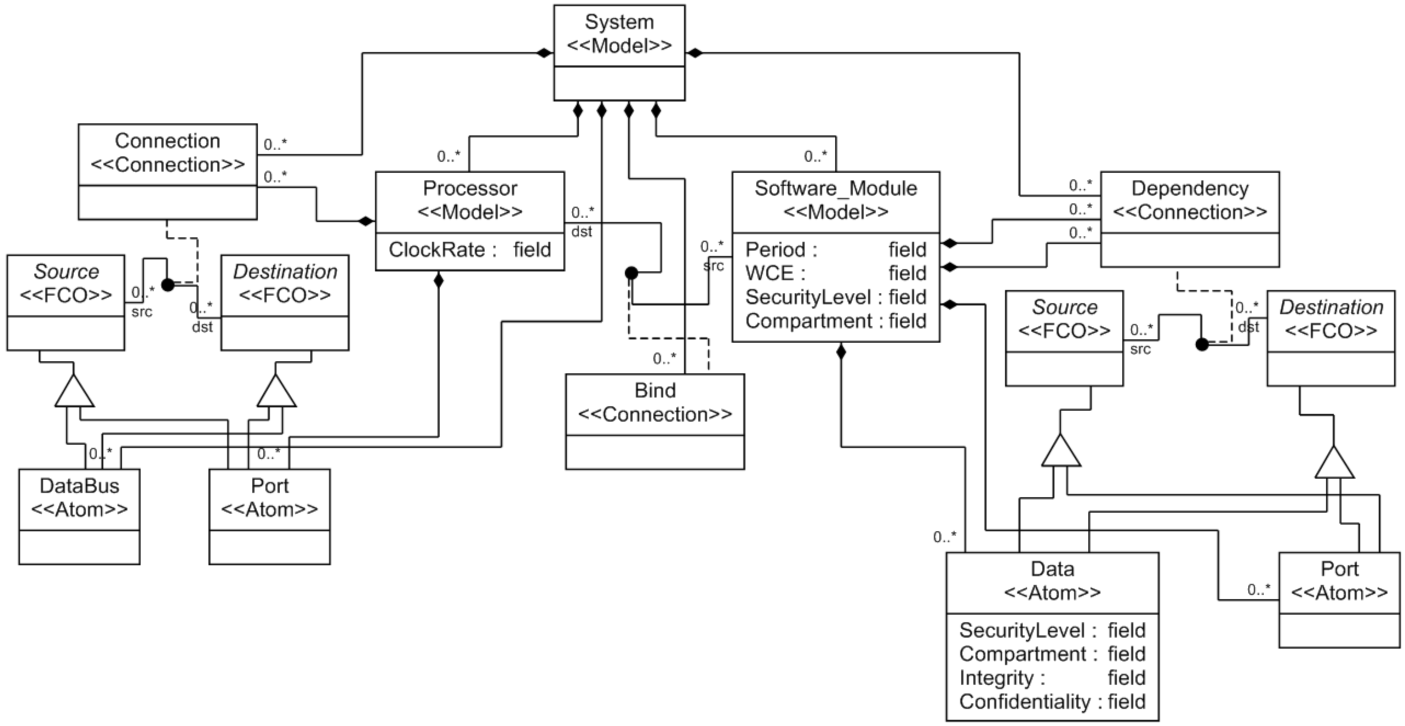
TESS Meta-model

- **Hardware components**



TESS Meta-model

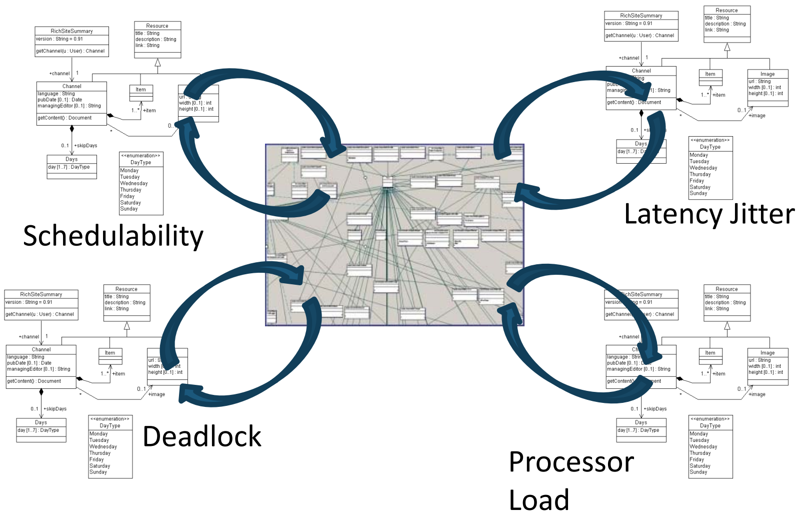
- **System-level view**



DISTRIBUTION A. Approved for public release; Distribution unlimited. (Approval AFRL PA # 88ABW-2014-0460, 11 February 2014)

Upfront Analysis

- **Upfront analysis is done during the system design stage and is used to**
  - Reduce uncertainty in architectural decisions
  - Evaluate system trade-offs
- **Models are annotated with implementation specific attributes (e.g. thread periodicity, WCET)**
- **Ex:**
  - Information flow
  - Schedulability



Evaluating Security

- **Information flow analysis**
  - Formal application of Bell-LaPadula[1] and Biba[2] information flow theory to a component-based dataflow model
  - Enables us to verify the confidentiality and integrity of information flow

A **data object** $D = (I, O, l)$ is a tuple:

- $I = \{i_1, i_2, ..., i_n\}$ is a set of **inputs**
- $O = \{o_1, o_2, ..., o_n\}$ is a set of **outputs**
- l is a **security level**, where $l \in L$ and $L \subset \mathbb{N}$

A **component** $C = (Do, I, O, Fo)$ is a tuple:

- Do is a set of data objects s.t. $Do \in C$
- A **dataflow connection** $f = (o_i, i_j)$ is an ordered pair
- Fo is a set of dataflow connections s.t. $Fo \in C$

Confidentiality property (i.e. no downflow): $\forall (o_i, i_j) \in Fo$ such that $o_i, l_i \in D_i$ and $i_j, l_j \in D_j$ then $l_i \leq l_j$.

Integrity property (i.e. no upflow): $\forall (o_i, i_j) \in Fo$ such that $o_i, l_i \in D_i$ and $i_j, l_j \in D_j$ then $l_i \geq l_j$.

Schedulability

- **Rate Monotonic Analysis (RMA)[3]**
  - Each task is periodic and has a Worst-Case Execution Time (WCET) based on the processor clock rate
  - Enables us to determine if a set of tasks are schedulable (i.e. do not miss any deadlines) on a processor

The **processor utilization factor** U is the fraction of processor time spent in the execution of a task set.

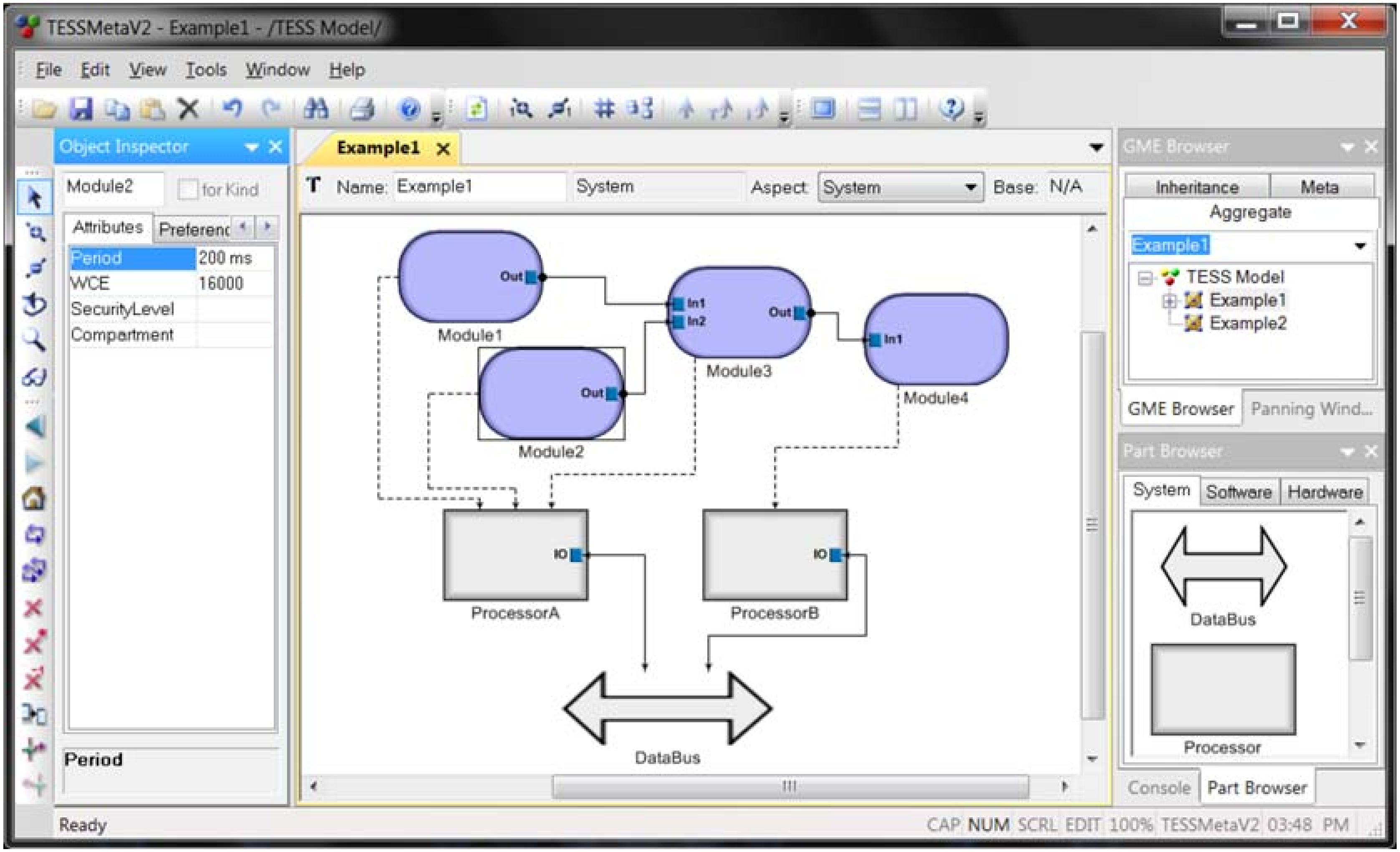
For a set of n tasks, $U = \sum_{i=1}^{n}(C_i/T_i)$ where C_i is the run-time and T_i is the request period of task i.

A set of n tasks is **schedulable** if $U \leq n(2^{1/n} - 1)$

Correct-by-Construction

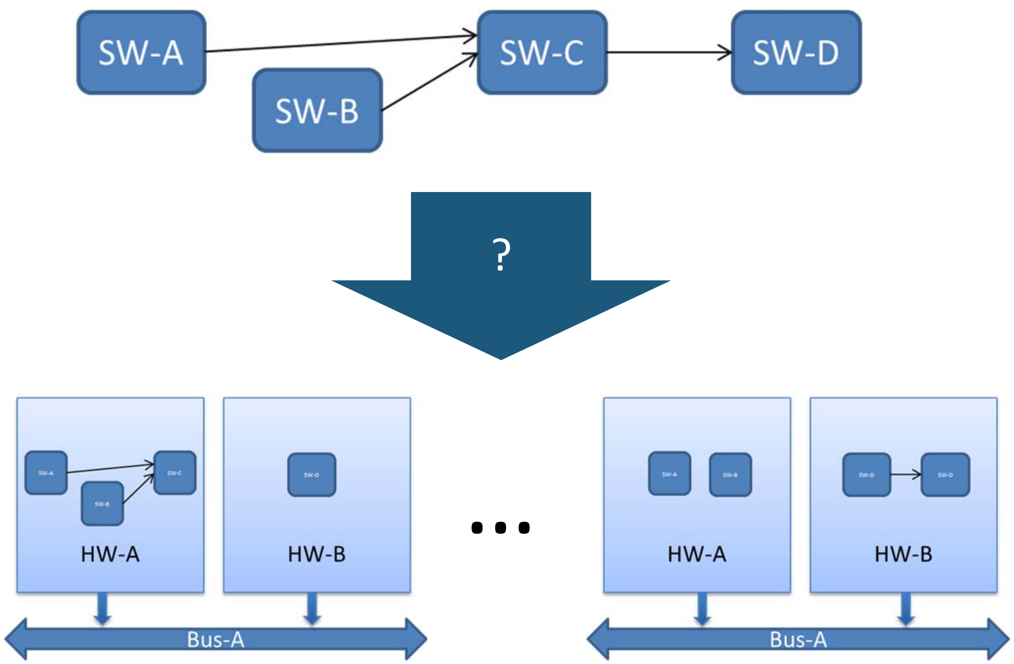
- **Design rules may be defined in the meta-model in order to enforce correct construction of model instances**
  - Object Constraint Language (OCL)
- **Example**
  - Design rule
    - *"All data items must have a security level equal or lesser than the security level of the software module in which they are processed."*
  - OCL Declaration
    - `self.parts( Data ) -> forAll( x:Data | x.SecurityLevel <= self.SecurityLevel );`

Secure Physical Deployment of Software System



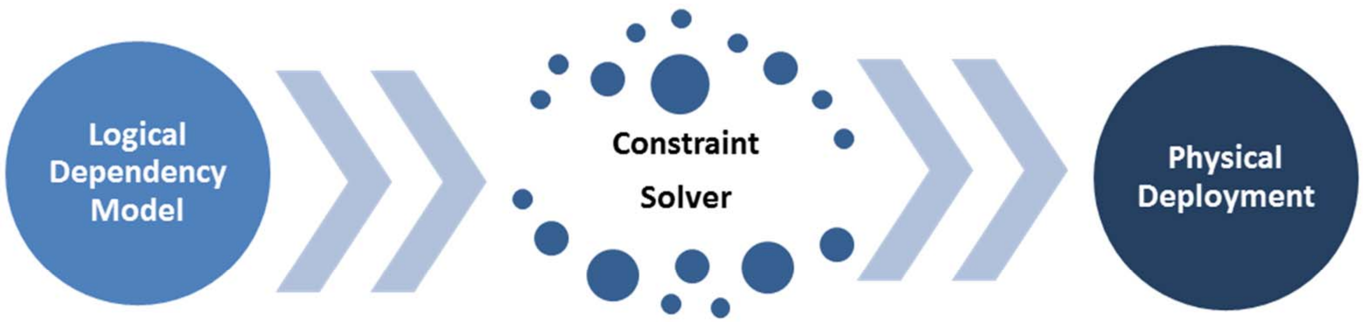
Secure Physical Deployment of Software System

- **Given the logical dependencies between K software modules, and a configuration of N processors, there exist N^K possible deployment mappings**
  – Computationally costly, manually infeasible



Secure Physical Deployment of Software System

1. **Define logical dependencies of software modules and configuration of hardware components**
2. **Add real-time and security attributes (security levels, clock rates, WCET, etc)**
3. **Manually specify required software to hardware bindings**
4. **Generate remaining software to hardware mapping that meets security and real-time constraints**



Summary

- **Model-driven development as an approach to deal with rising complexity of software systems**
  – Abstraction, formal verification, component reuse
- **Security considerations early-on in the development process**
  – Cross-domain relationships of security mechanisms influence design
  – System trade-off evaluation
- **A domain-specific model for secure embedded systems development**
  – Information flow and schedulability analysis
  – Deployment mapping

References

1. D. E. Bell and L. LaPadula, “Secure computer systems: Mathematical foundations.” Technical Report mtr-2547, MITRE Corporation, Bedford, MA, 1973.
2. K. Biba, “Integrity considerations for secure computer systems.” Technical Report mtr-3153, MITRE Corporation, Bedford, MA, 1975.
3. C. L. Liu and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard-real-time environment," J. ACM, vol. 20, no. 1, pp. 46-61, 1973.
4. Generic Modeling Environment (GME), Institute for Software Integrated Systems, Vanderbilt University, 01/17/2014, http://www.isis.vanderbilt.edu/Projects/gme/
5. Eclipse Modeling Framework (EMF), The Eclipse Foundation, 01/17/2014, http://www.eclipse.org/modeling/emf/
6. Metaedit+, MetaCase, 01/17/2014, http://www.metacase.com

Thank You

Questions?

Acronyms

- **ASIC: Application-Specific Integrated Circuit**
- **DSM: Domain-Specific Modeling**
- **DSP: Digital Signal Processing**
- **EMF: Eclipse Modeling Framework**
- **GME: Generic Modeling Environment**
- **MDD: Model-Driven Development**
- **OCL: Object Constraint Language**
- **OMG: Object Management Group**
- **RMA: Rate Monotonic Analysis**
- **SAE: Society of Automotive Engineers**
- **TESS: Trusted Embedded Software System**
- **WCET: Worst-Case Execution Time**