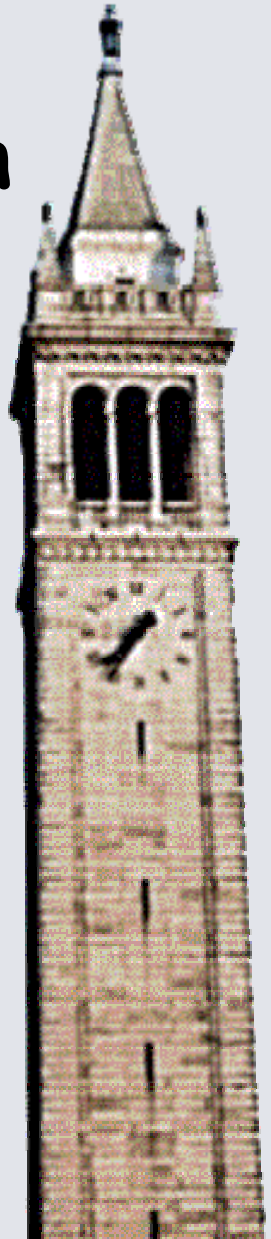
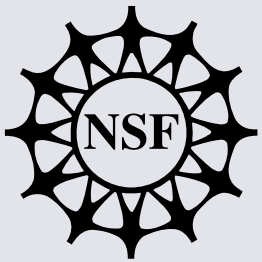
Impact: Fault Tolerance and High Confidence Embedded Systems Design

Edited and presented by
Gabor Karsai
Vanderbilt University, ISIS



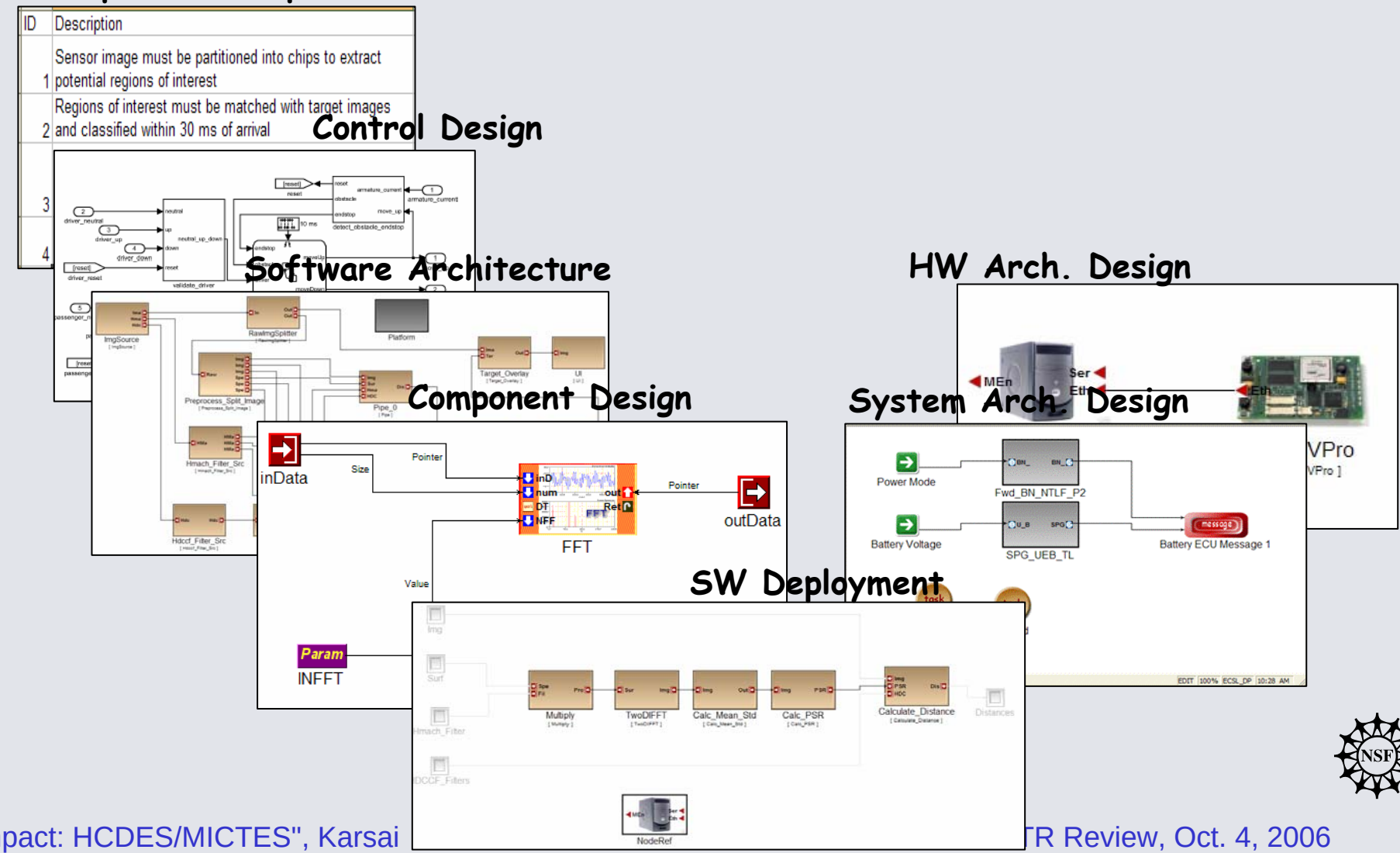


Chess Review
October 4, 2006
Alexandria, VA

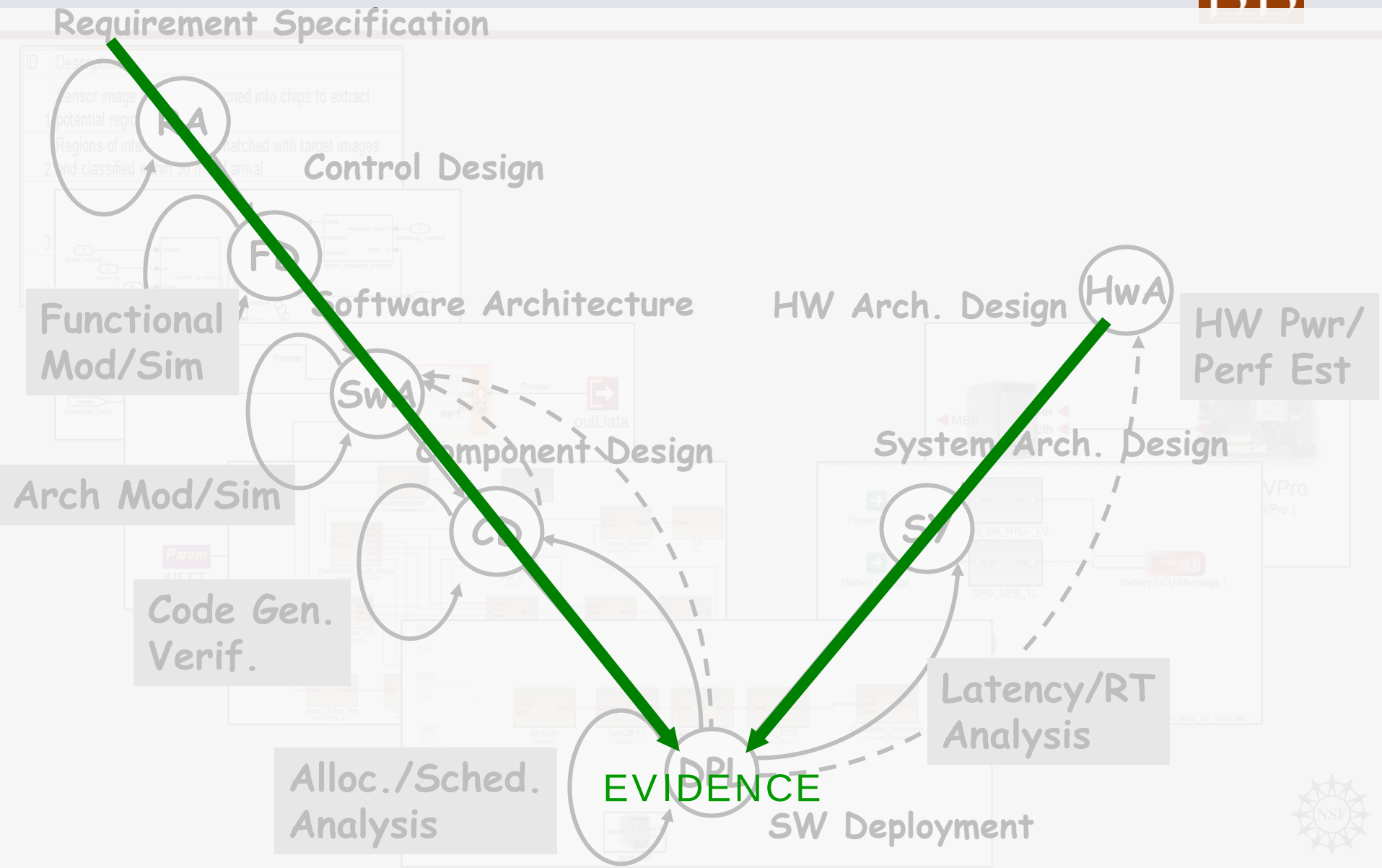
Center impact (1): New Start Research Project

- Multi-University Research Initiative:
  - Frameworks and Tools for High-Confidence Design of Adaptive, Distributed Embedded Control Systems
  - Participants: Berkeley, CMU, Stanford, VU
- Objectives:
  - Development of a theory of deep composition of hybrid control systems with attributes of computational and communication platforms
  - Development of foundations for model-based software design for high-confidence, networked embedded systems applications.
  - Support of high-level reusability of tools in domain-specific tool chains
- Web: https://wiki.isis.vanderbilt.edu/hcddes/

Embedded Control System Design Flow



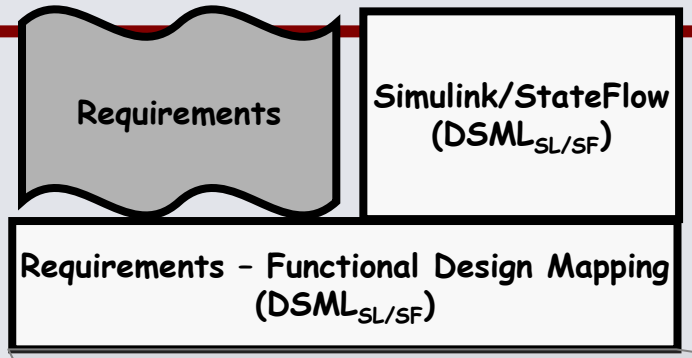
Design Flow: Tools and Analysis



Overall Undertaking

- Development of component technologies in all areas (theory/design/tools)
- Incrementally building a tool chain for a selected domain (UAV flight and mission control)
- Demonstration of control software development with the tool chain
- Experiments

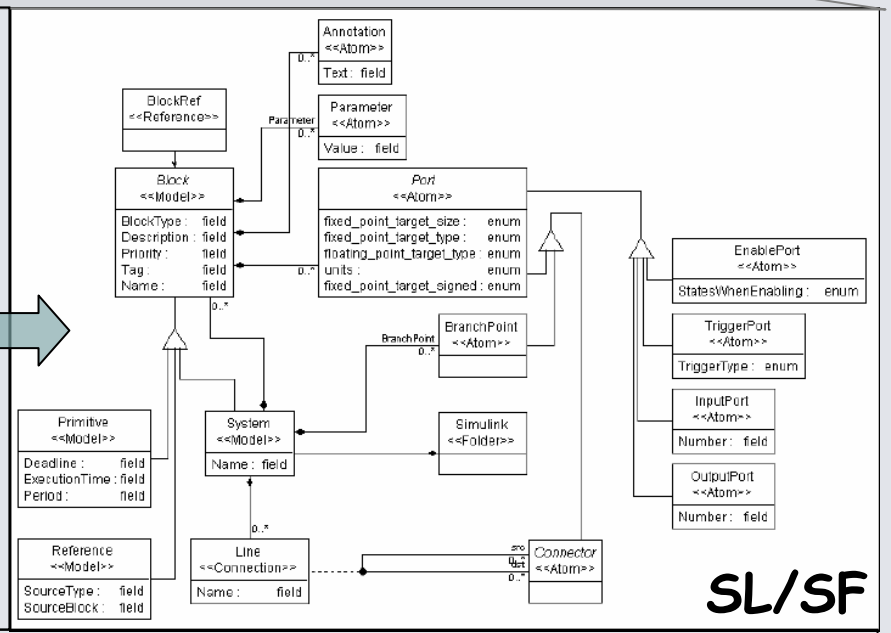
Control Design – A DSML View



Objective: Define the control laws to meet requirements
Platform: SL/SF-like modeling language, (Ptolemy 2; GME)
Tools:
SL/SF Model Builder+Simulator (Ptolemy 2)

Requirement Specification

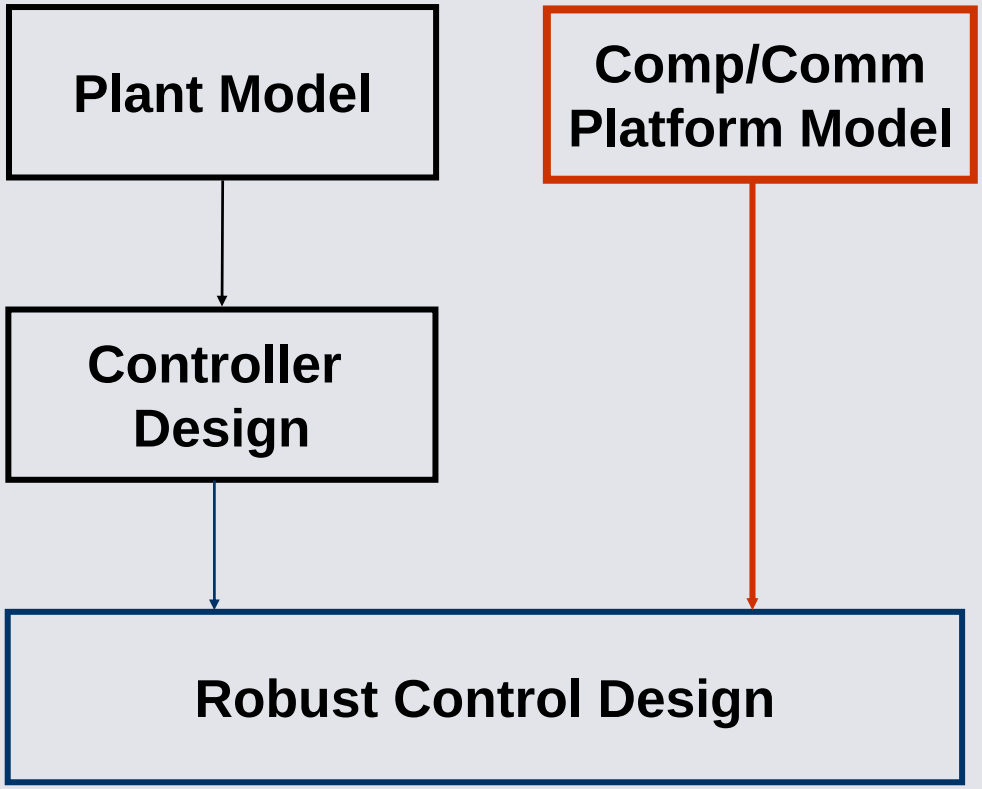
ID	Description
1	Sensor image must be partitioned into chips to extract potential regions of interest
2	Regions of interest must be matched with target images and classified within 30 ms of arrival
3	The classification must be adaptively configured according to number of sensor images arriving per second
4	The number of target classes must be adaptively configurable to support rapid deployment in theater



Requirements - Functional Design Mapping

Control Design: Approaches

Goal: Design controller behavior satisfying all requirements



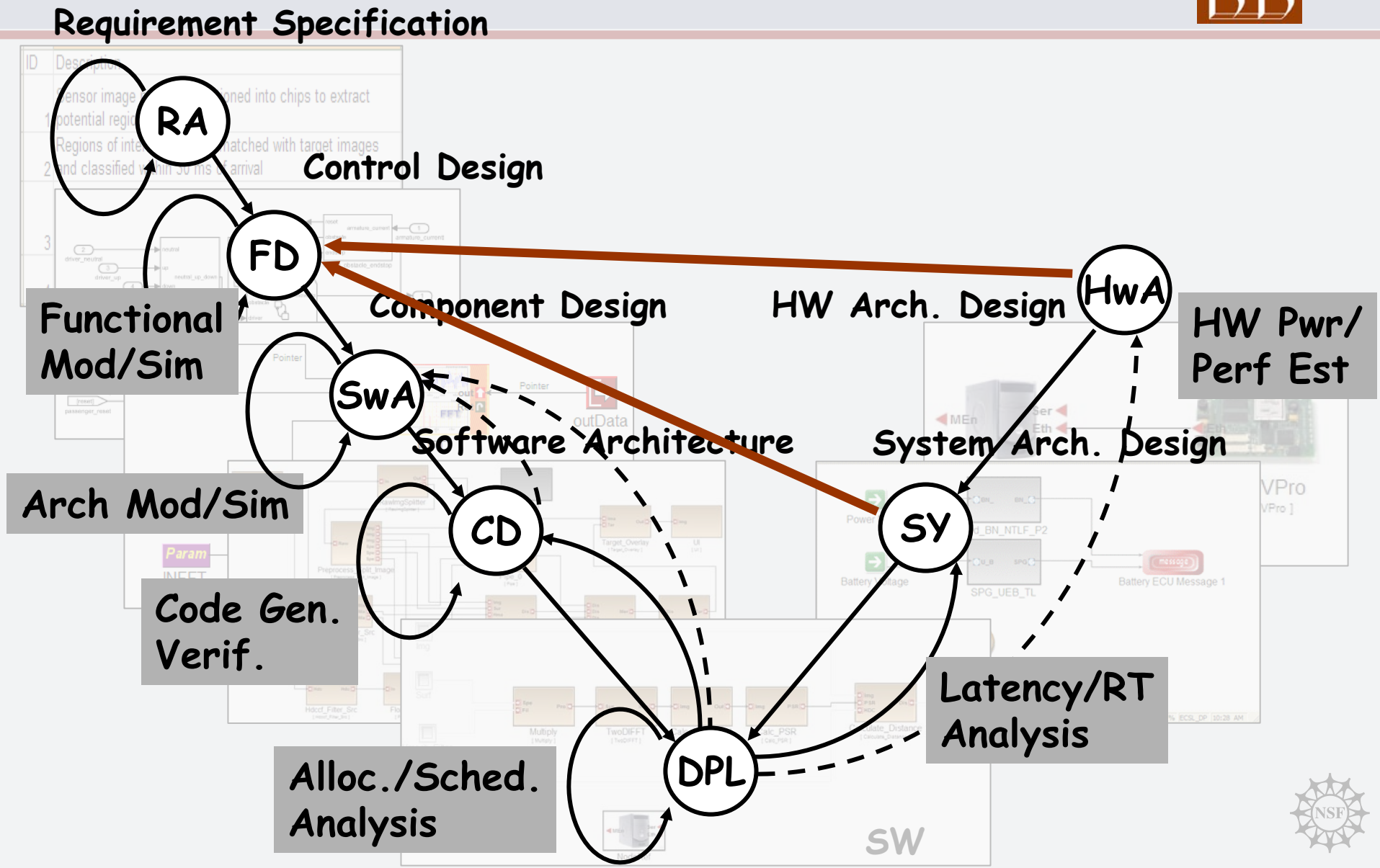
- **Embedded Systems Modeling and Deep Compositionality**
- **Hierarchies of Robust Hybrid and Embedded Systems**
- **Verification and Validation of Conservative Approximations**
- **Adaptive Control Architectures for Uncertainty Handling**

- Quantization, finite word length, round-off errors
- Modality
- Limited resources, resource sharing
- Concurrency models, scheduling
- Limited communication bandwidth, networking

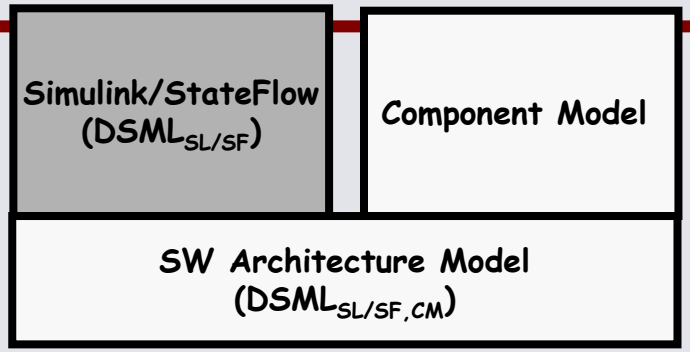
- Mathematical model of the Plant
- Design of a lin. or non-lin. controller satisfying stability/performance requirements
- Simulations/refinement

- Uncertain dynamics, unknown non-linearities
- Fault effects, sensing errors
- Fault adaptive control
- Robust analysis, (SDP, LMI),
- Simulations

Addition to the Design Flow



SW Architecture Design

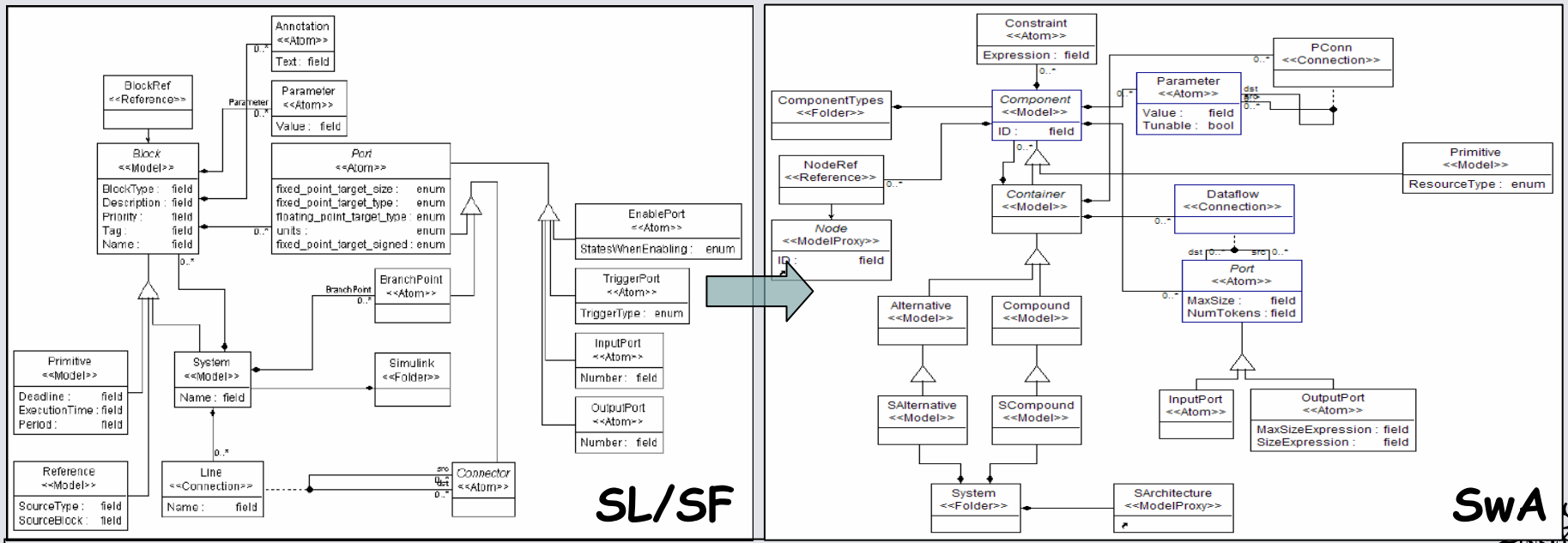


Objective: Optimize the SW architecture by selecting a component model and by allocating functions to components.

Platform: MoC-s

Tools:

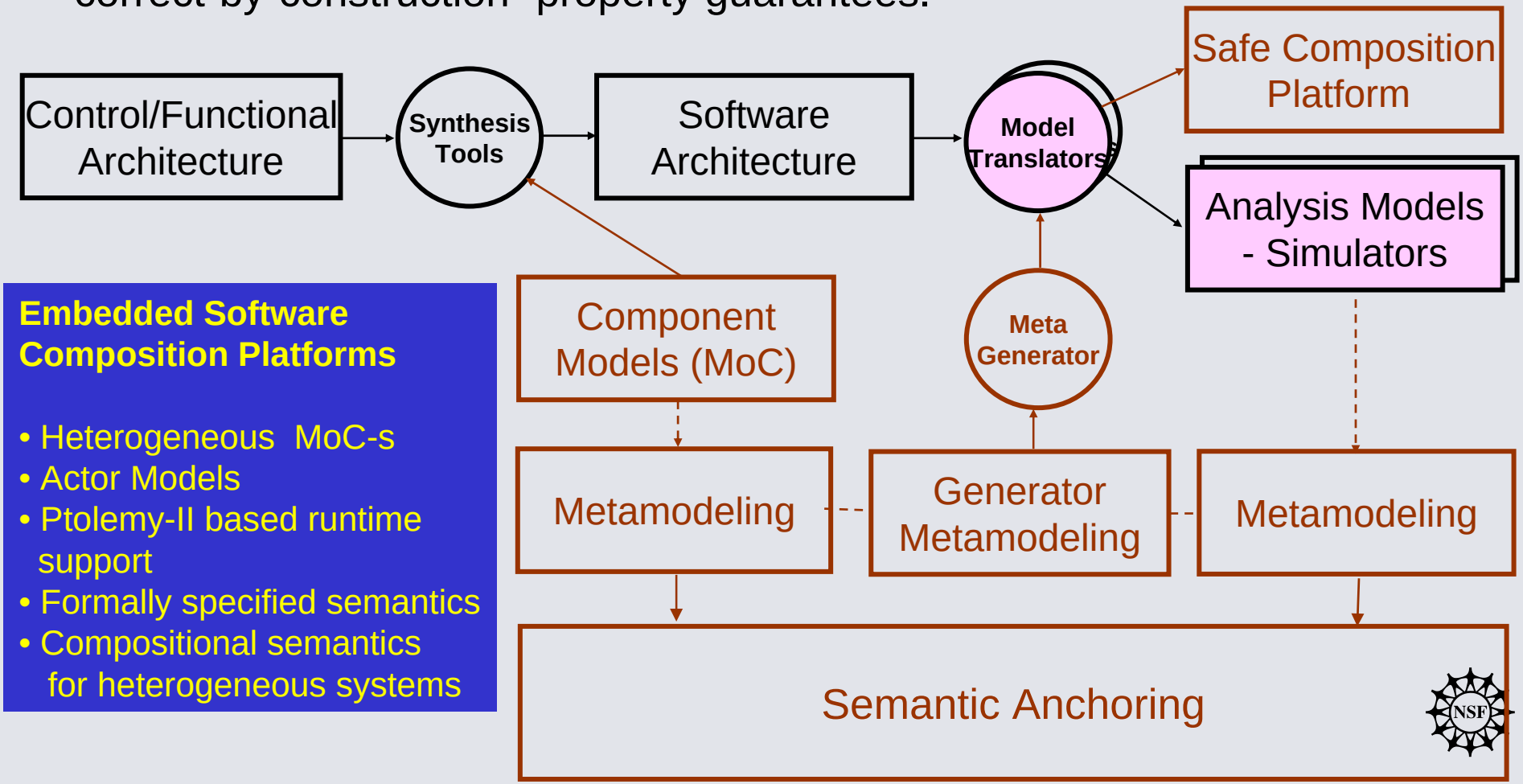
GME, GReAT, DESERT, Ptolemy-2,…



Functional Architecture – SW Architecture Mapping

Software Architecture Verification

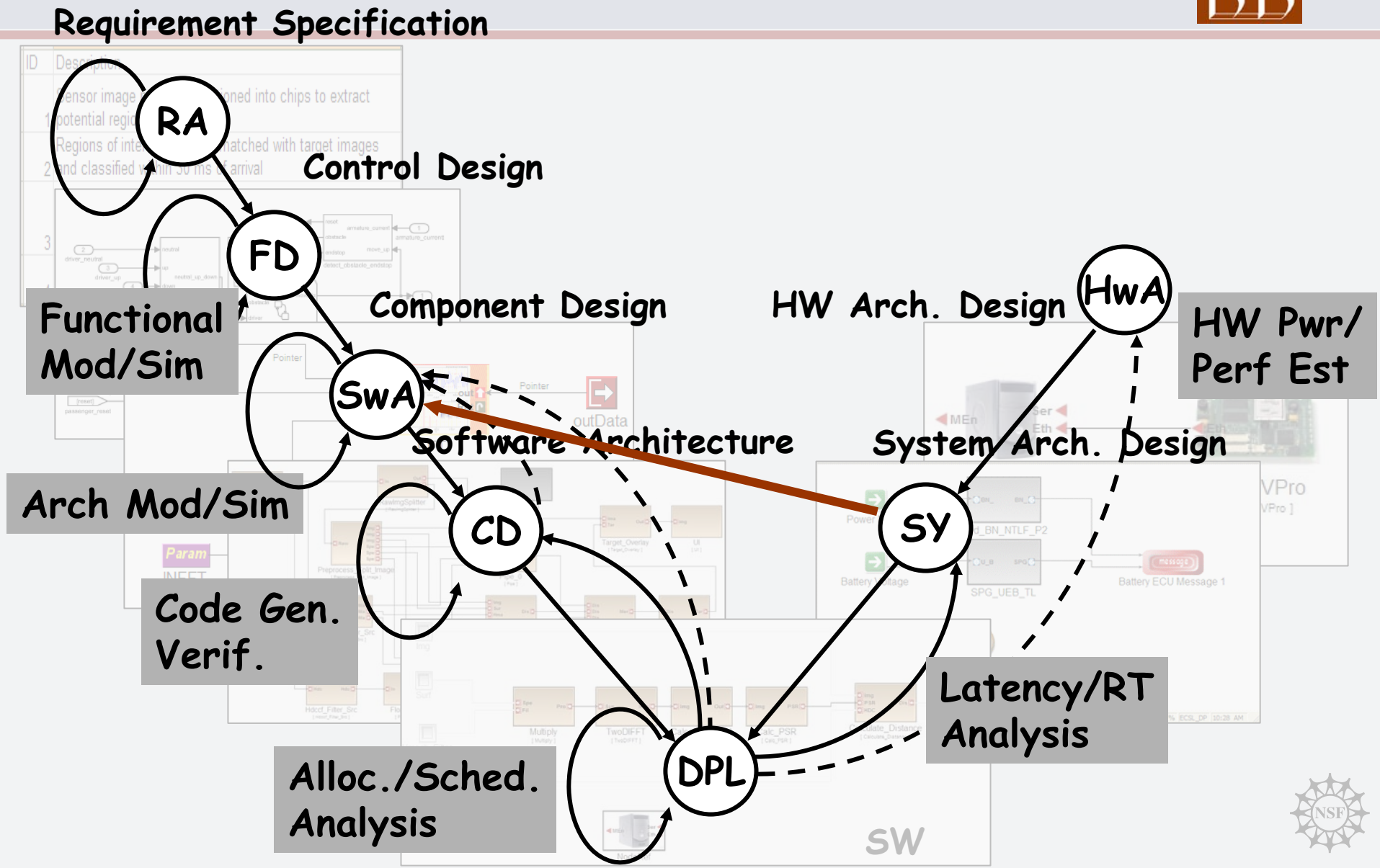
Goal: design software architecture using well understood composition platforms that allow verification of properties using analysis or "correct-by-construction" property guarantees.



Embedded Software Composition Platforms

- Heterogeneous MoC-s
- Actor Models
- Ptolemy-II based runtime support
- Formally specified semantics
- Compositional semantics for heterogeneous systems

Addition to the Design Flow



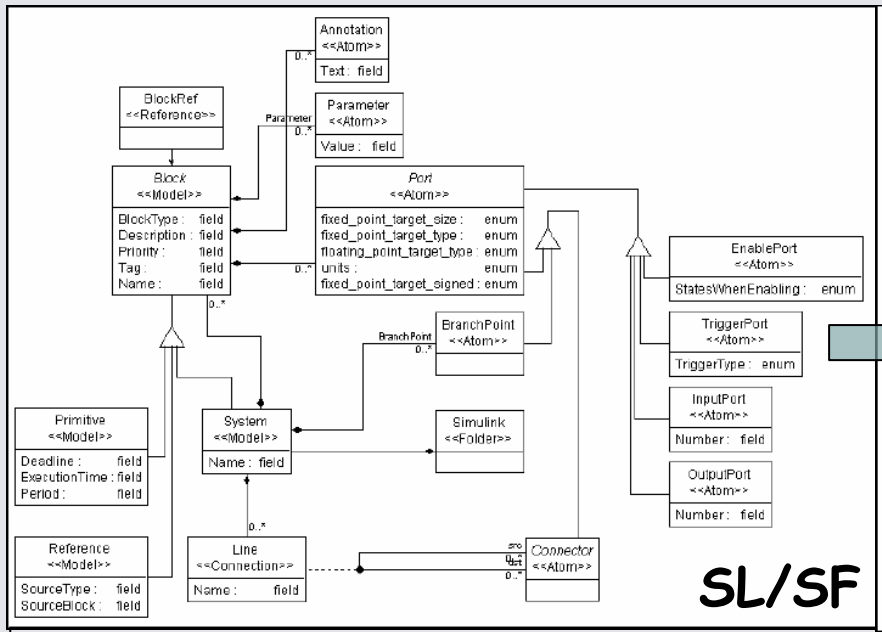
SW Component Design

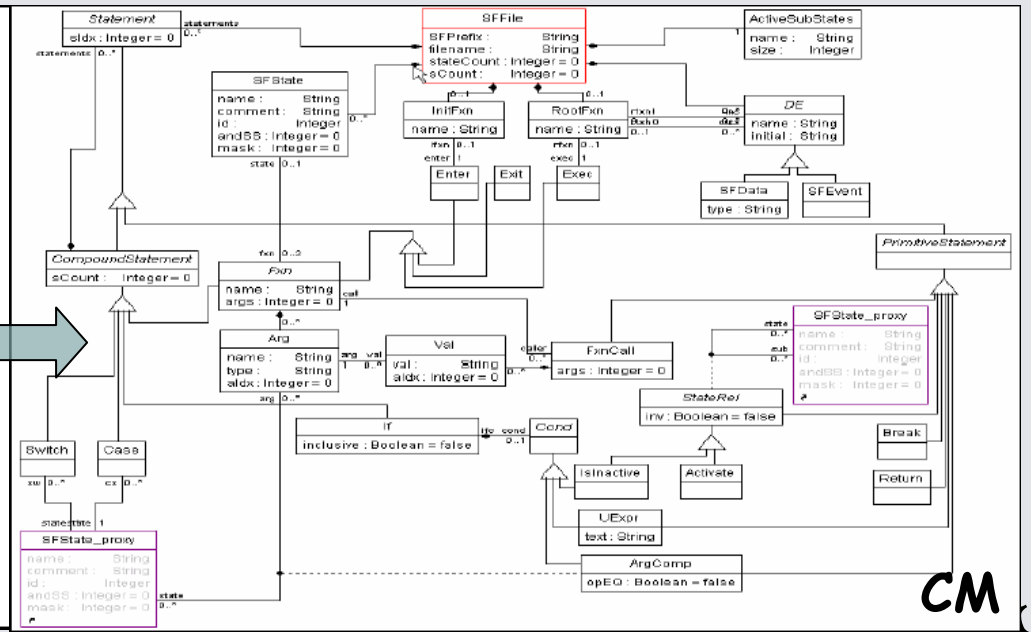
Simulink/StateFlow ($DSML_{SL/SF}$)	Component Implementation Lng.
SW Components	

Objective: Design and implement SW for components satisfying behavior defined by control laws.

Platform: Component Implementation Languages (Java, C++, Other..)

Tools: Generators (RT-Workshop; GReAT), Compilers, WCET Analyzers

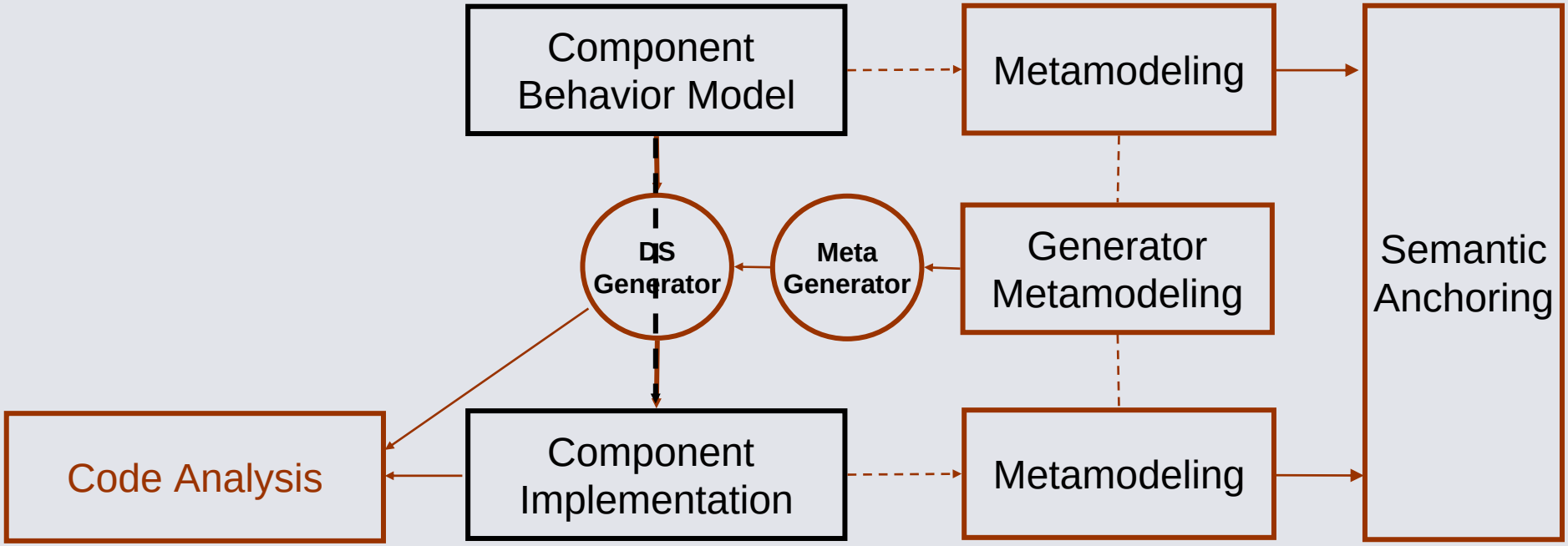




Functional blocks – SW Component Mapping

Software Component Verification

Goal: prove that the component software behaves as intended under all foreseeable operating conditions.

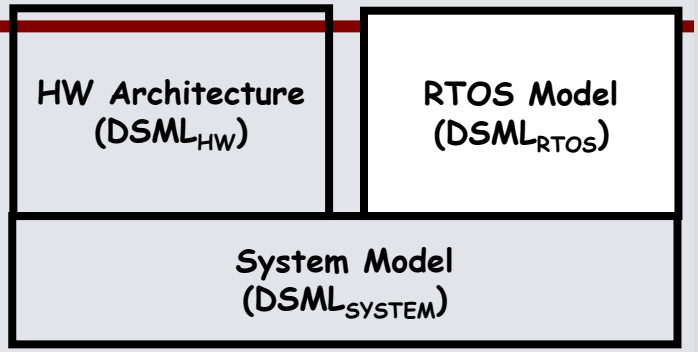


Automated Source Code Verification and Testing
- Model-based test generation
- Advanced static analysis

- Model refinement
- Model verification
- Model compilation or hand coding
- Static analysis
- Test-based verification

Model Integrated Computing
- Metamodeling
- Model-based code generation
- Meta-model-based tesing of code generators

System Configuration Design

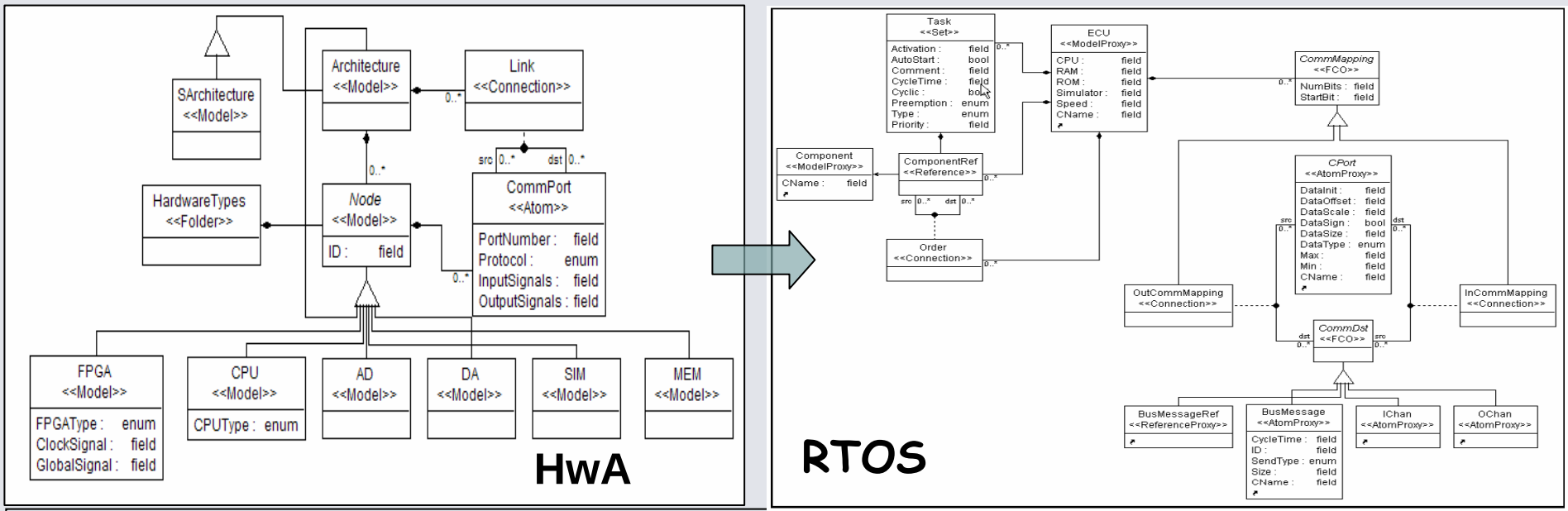


Objective: Design System configuration that meets cost/reliability/power requirements.

Platform: Comm-links; RTOS, Comp. Middleware

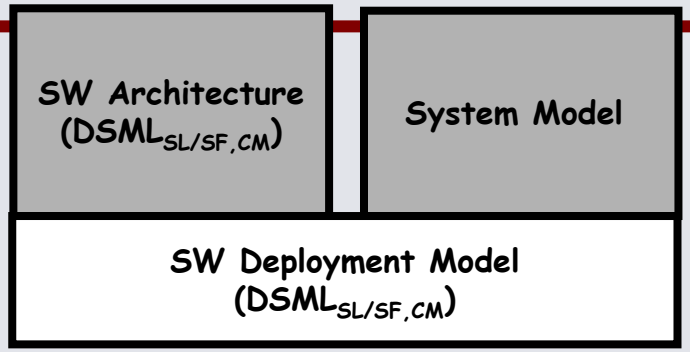
Tools:

GME, RTOS, Comp. Middleware tools

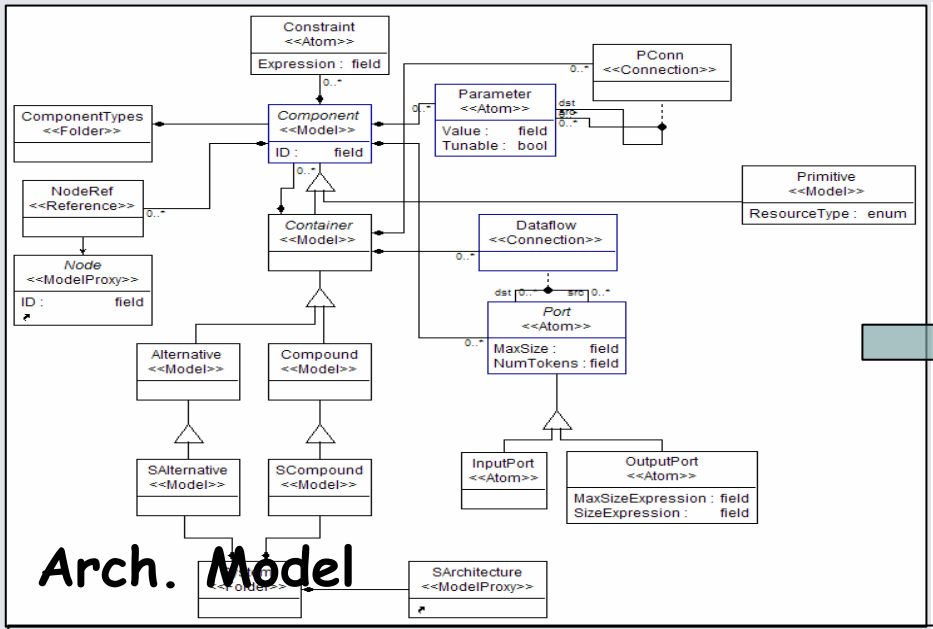


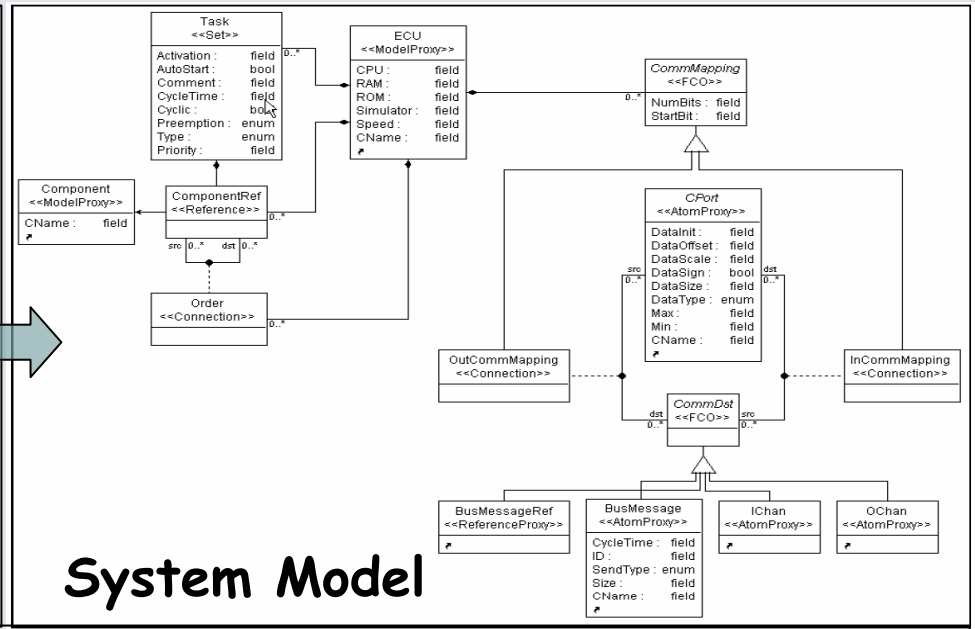
System Modeling: HW-Comm-RTOS mapping

SW Deployment



Objective: Optimize System architecture by allocating SW components to RTOS Tasks and Communication Channels.
Platform: Composition Model
Tools: GME, DESERT, Timing Analysis,…





SW Deployment: SW Components – System Mapping

Approach & Technical Challenges

Guaranteed behavior of distributed control software using the following approaches:

(1) extension of robust controller design to selected implementation error categories

(2) providing "certificate of correctness" for the controller implementation

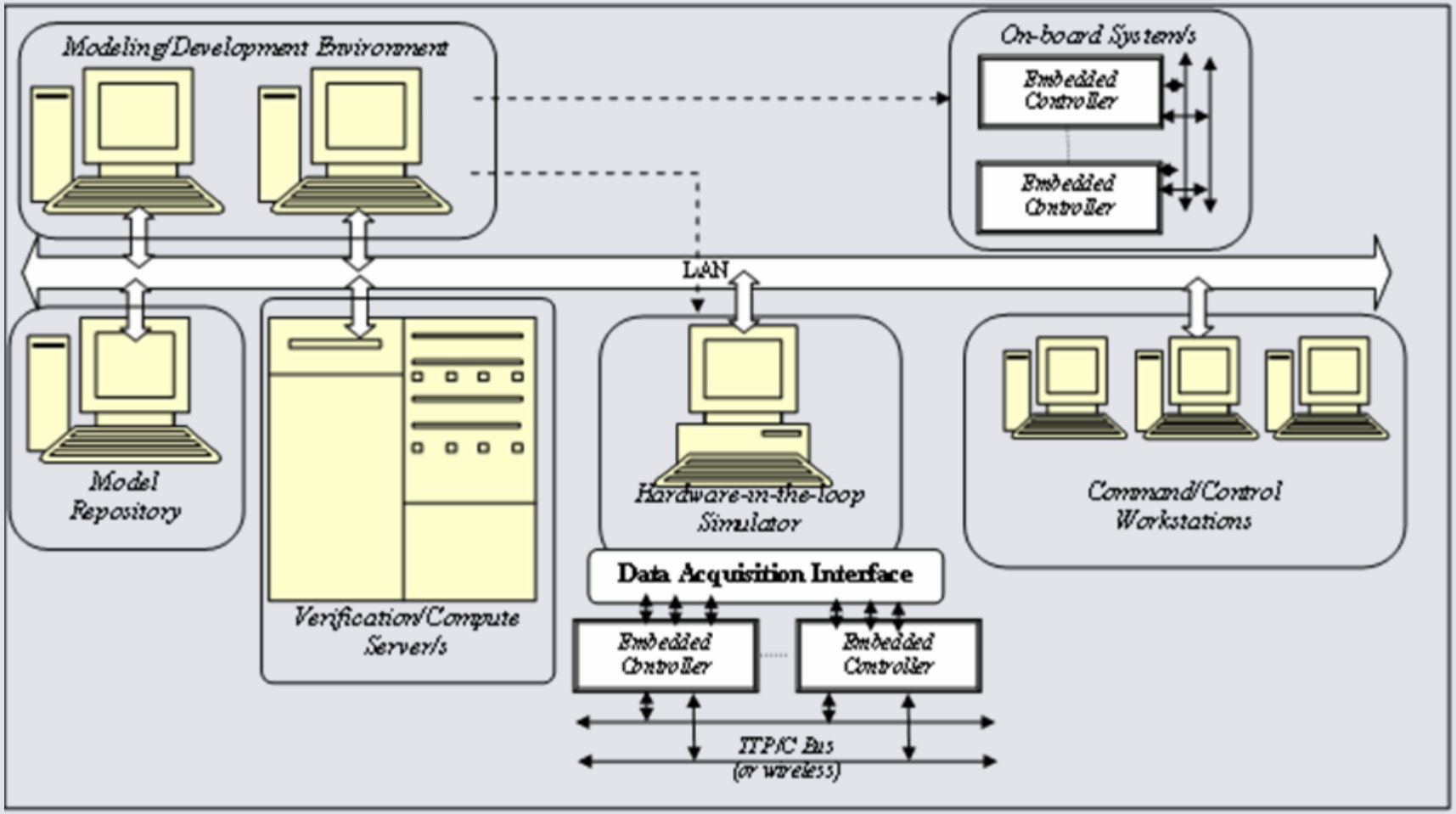
(3) development of semantic foundation for tool chain composition

(4) introducing safe computation models that provide behavior guarantees

Expected Deliverables

- Composable tool architecture
  - New generation of Open Tool Integration Framework
  - Prototype Tool Chain
- Testing and Experimental Validation
  - Software fully built and validated by tools
    - Avionics for small UAVs
    - Mission Management
    - COP for C2

Computing Platforms:
Design/Verification Tools, Embedded Devices



Transition Approach

- Tools are disseminated through the ESCHER Repository (Open Source)
- Government: AFRL connections
  - AFRL/IF, AFRL/VACC, AFRL/VACA, AFRL/CSD
- CerTA Project (Boeing/UCB)
- Future Combat Systems Program (Boeing/VU)

Center Impact (2): Collaborative Research with NASA/ARC

Model-Integrated Computing Tools for Exploration Systems (MICTES)

Goal: Assured Development of Flight Control Software for Spacecraft Applications

- **Front-end Modeling**: Simulink/Stateflow
- Code generation using a GReAT-based model transformation tool
- CG output includes annotated code (verification conditions)
- Model checker/theorem prover is used to prove code properties (ARC)
- **Expected result**: Integrated code generator/code verifier tool for Simulink/Stateflow-based Embedded Software Development