Supporting the Design of Safety Critical Systems Using AADL

T. Correa, L. B. Becker, J.-M. Farines,
J.-P. Bodeveix, **M. Filali**, F. Vernadat

IRIT LAAS UFSC

Agenda

- Introduction
- Proposed Approach
- Verification Process
- Conclusions

Introduction

- Most computer systems are embedded (95%), and many of these are **critical**
- AADL is a textual and graphical language used to **design** and **analyze** the **software and hardware architecture** of systems
  - functional interfaces to components (such as data inputs and outputs)
  - performance-critical aspects of components (such as timing)

Goal

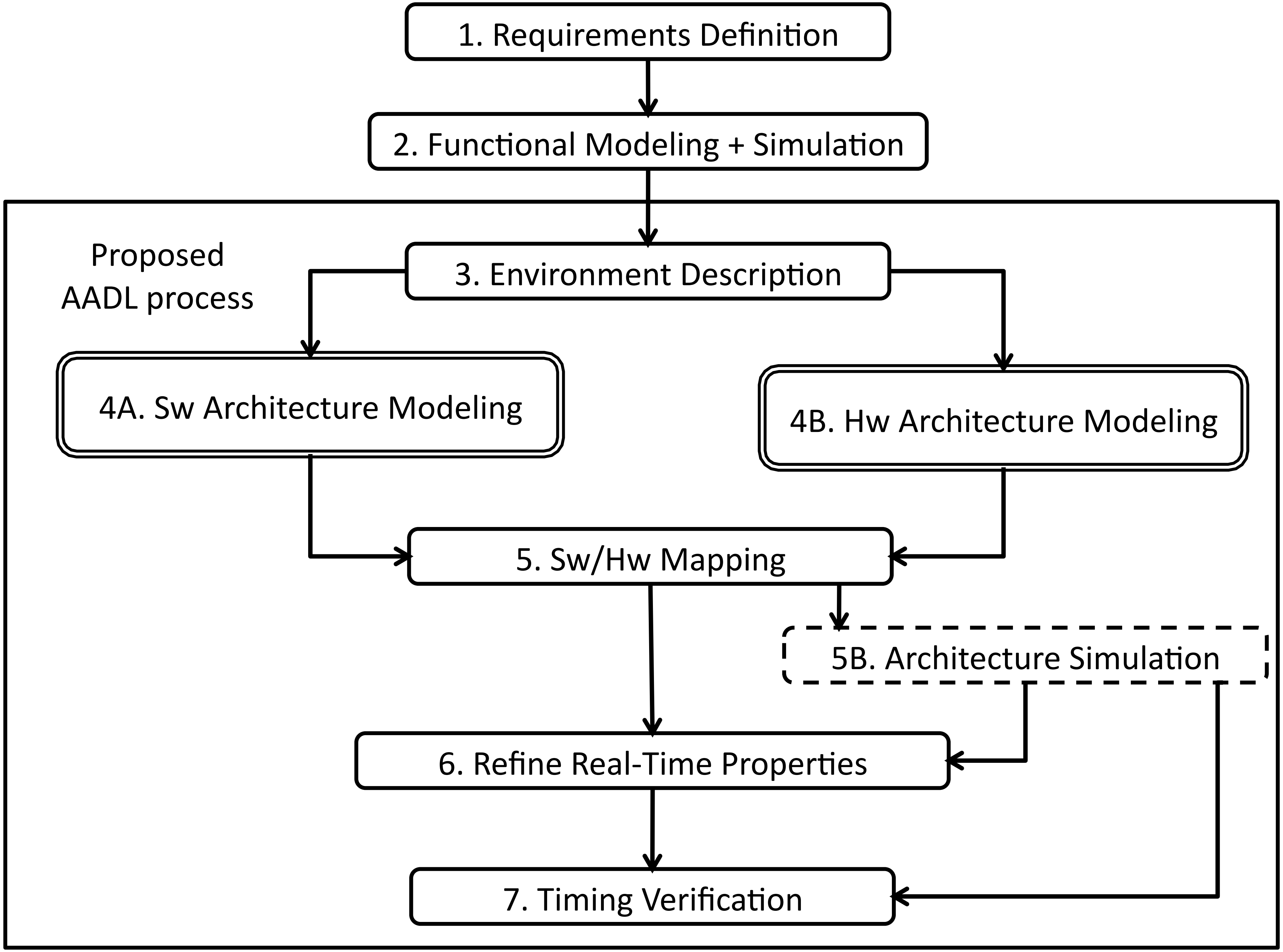
- Present a design-process for critical embedded systems to supports the safe design of the system's architecture using MDE's principles
- Propose an approach that supports model checking over AADL models
- How to deal with timing properties? (ongoing study with the hardware team TRACES: wcet analysis)

Our Proposal

- Use of AADL as a unique formalism for:
  - Hw and Sw people,
  - synchronous and asynchronous aspects
- In the AADL model, perform a sequence of model enrichments, which finishes when the model is suitable for verification
- Experimentation on a case study: parking problem

Proposed Approach

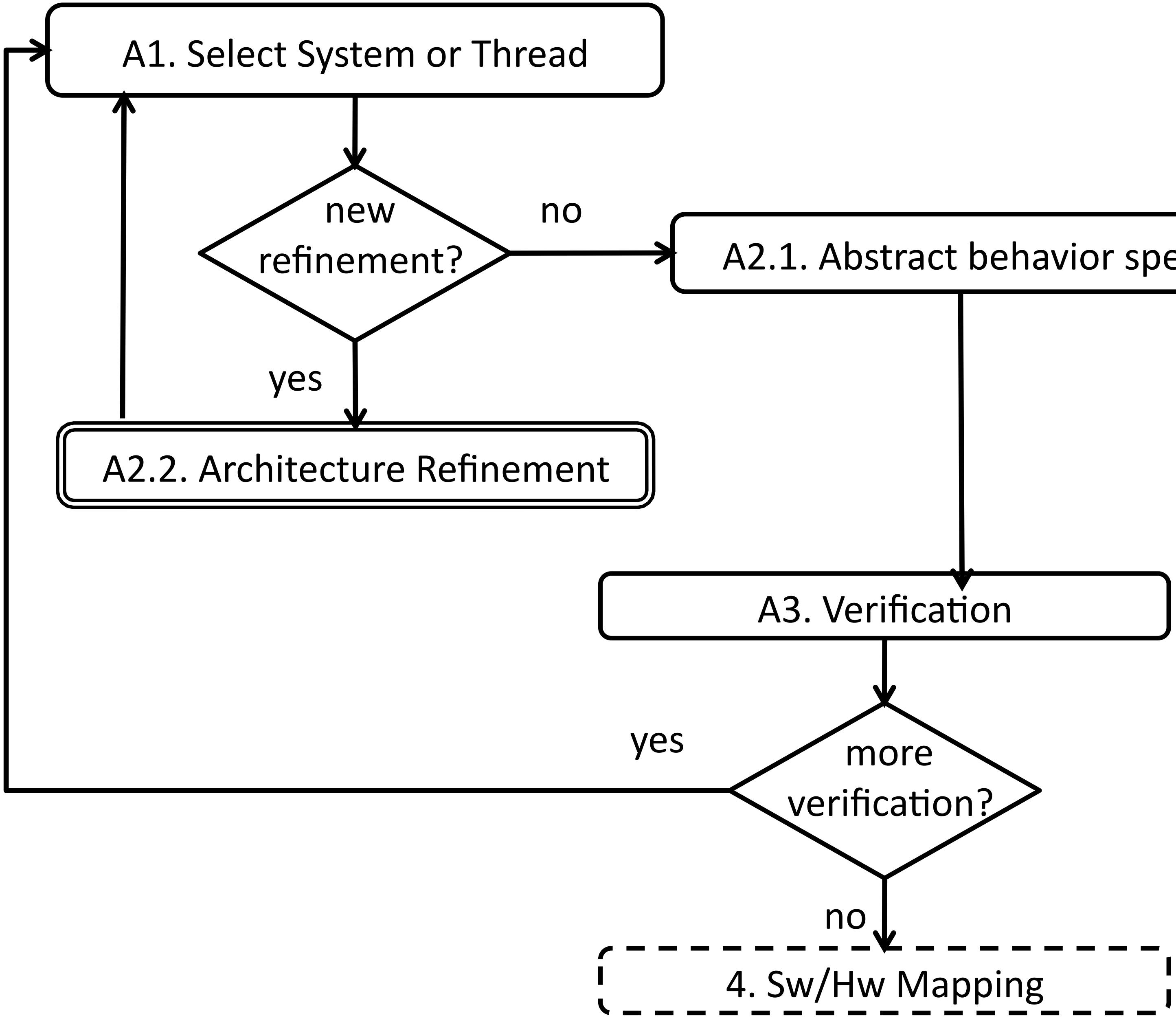
- It starts with the **definition of the functional and non-functional requirements** of the system…
- Constraint: Platform may be a priori given
- …it is concluded with the final **model verification**, which uses as input the AADL model updated with the precise timing information.


1. Requirements Definition
2. Functional Modeling + Simulation
Proposed
AADL process
3. Environment Description
4A. Sw Architecture Modeling
4B. Hw Architecture Modeling
5. Sw/Hw Mapping
5B. Architecture Simulation
6. Refine Real-Time Properties
7. Timing Verification

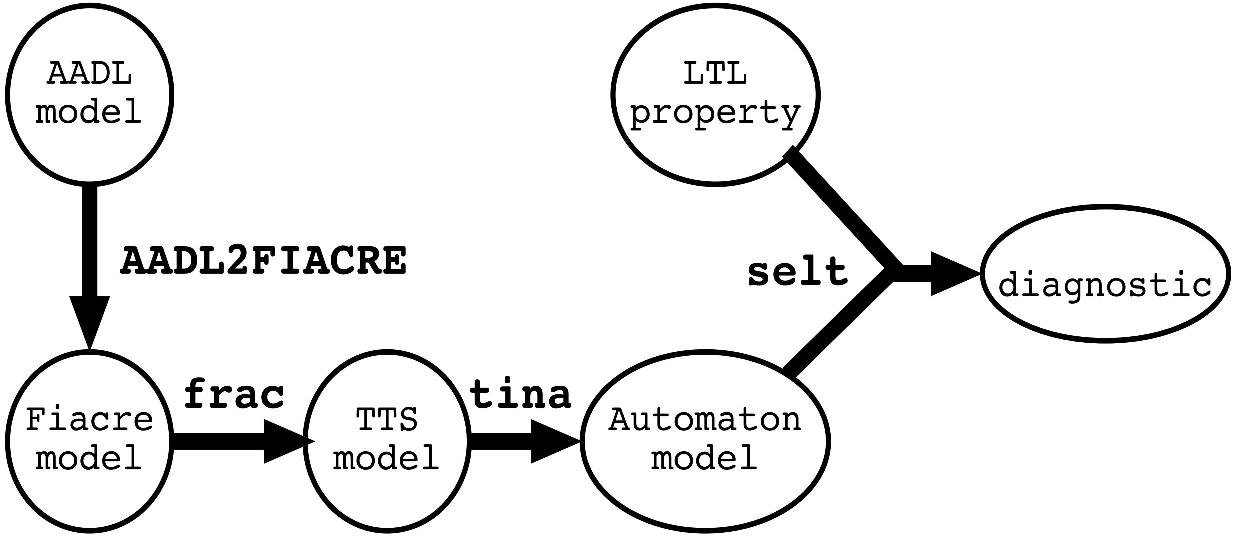
Successive Refinements

- the resulting system architecture goes through several verification steps in order to assure its correctness
- It is performed a sequence of model transformations, which starts with an AADL model and finishes with an automaton model that can be verified
- Initially the design is synchronous it ends asynchronous (physical architecture)

3A. Software Architecture Modeling


A1. Select System or Thread
new refinement?
no
A2.1. Abstract behavior spec
yes
A2.2. Architecture Refinement
A3. Verification
yes
more verification?
no
4. Sw/Hw Mapping

Verification Process



FIACRE is the pivot language of the TOPCASED project
FIACRE is a process algebra: message and shared memory.
TINA: verification engine (Petri net based)
AADL execution model « helps » in fighting combinatorial explosion

Need of property patterns
Need of better support for communication abstraction

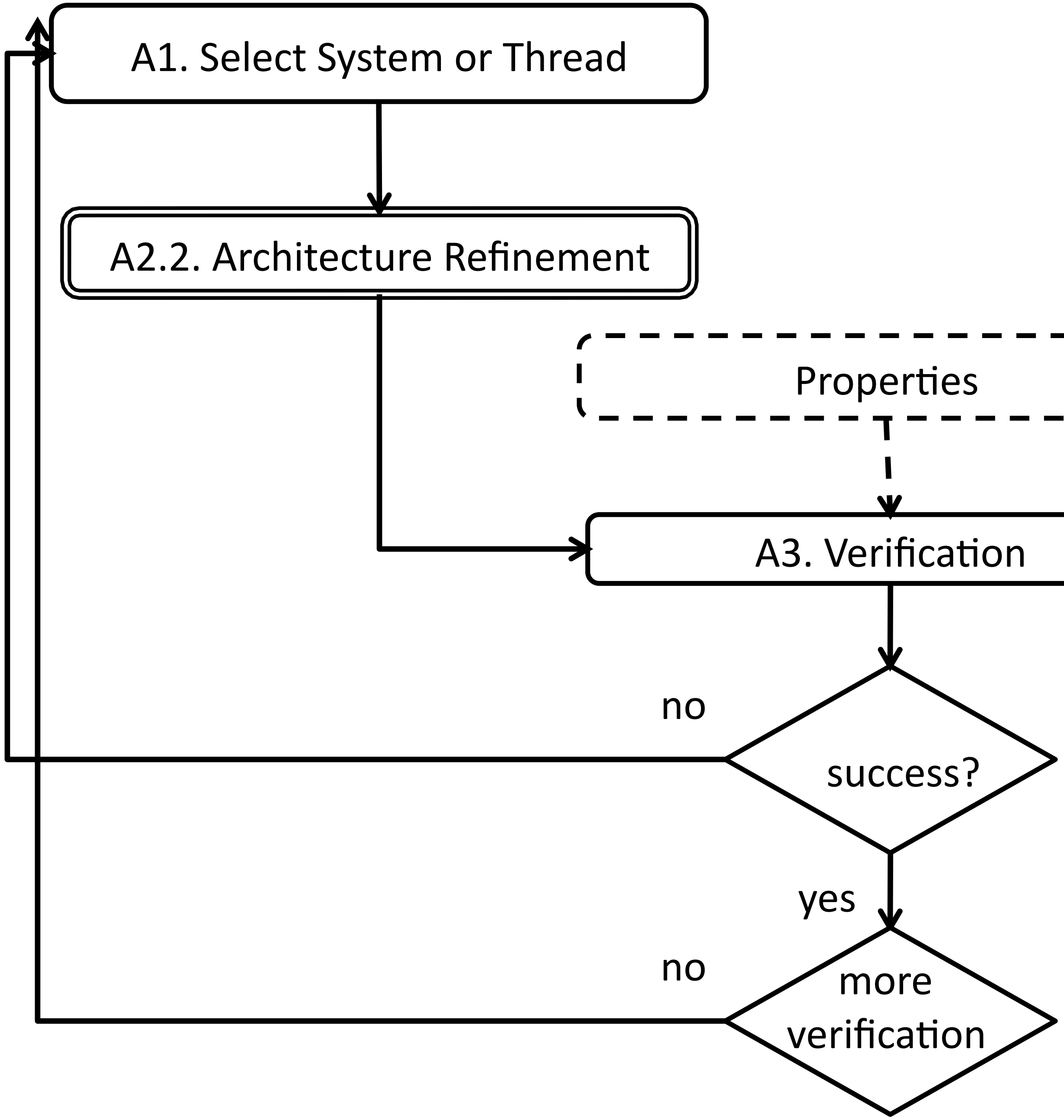
Properties

- Use of temporal logic: LTL enriched with events: SE-LTL. In fact, LTL and CTL are not enough: use of Modal Mu calculus: reason over atemporal properties (not temporised)
- Need of an intuitive logic to reason over the system and its environment. Requests are state and or event based.
- Need of patterns to avoid new (usually complex) formulas and reuse existing ones.

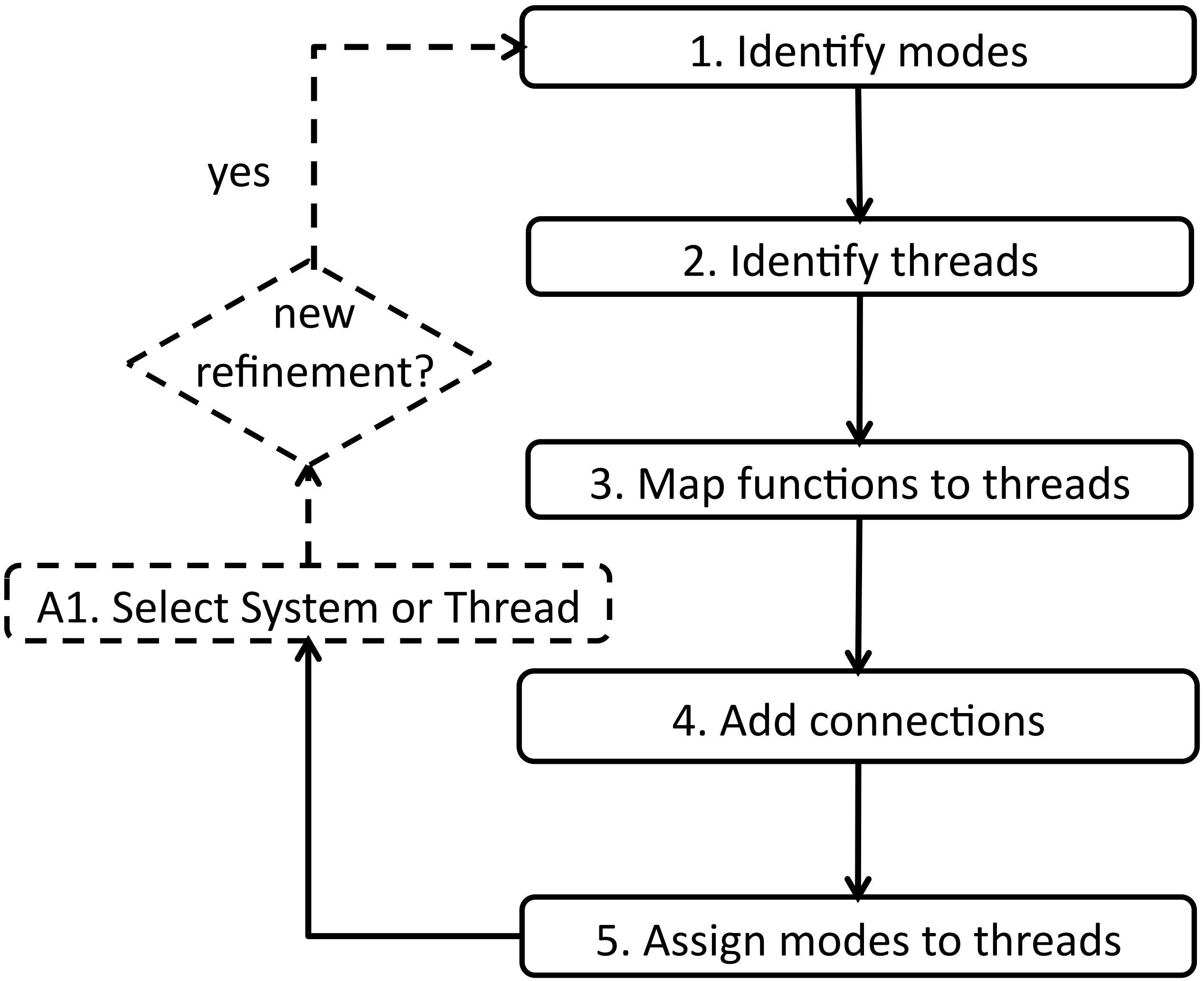
Conclusions

- Design methodology for software-hardware systems.
- It is not a top down or bottom up approach.
- Use of AADL as a unique language to address software and hardware issues.
- AADL execution model helps for fighting against combinatorial explosion.
- Use of logics to express the properties. Need for a logic to express the interaction between the system and its environment. Need of patterns.

3A. Software Architecture Modeling

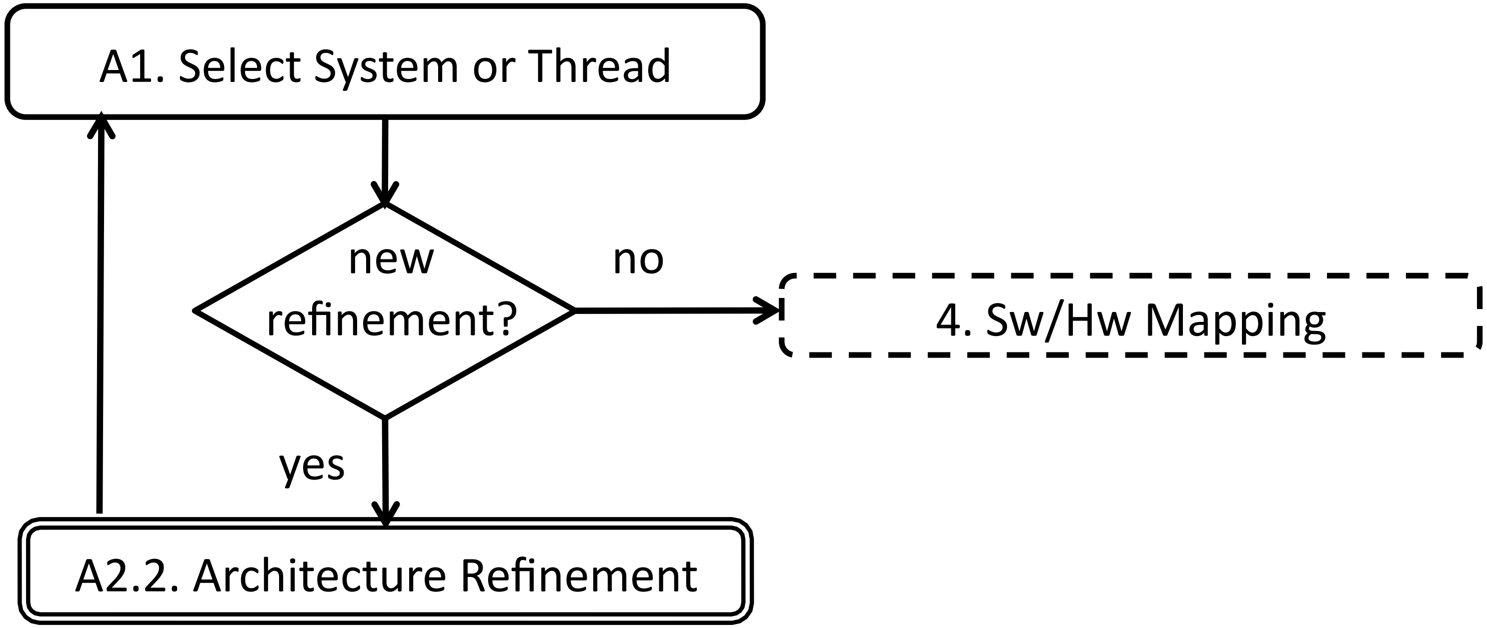


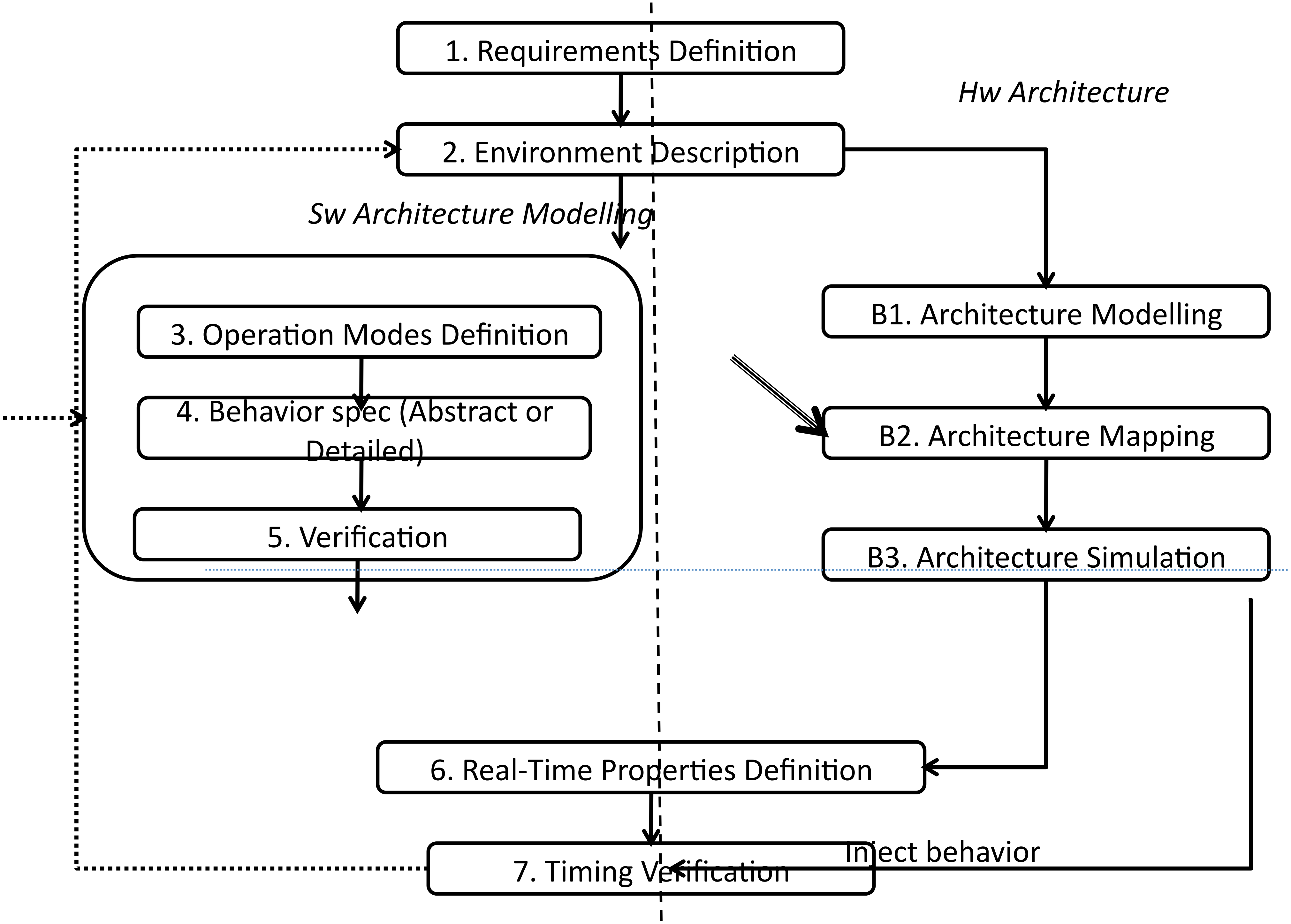
A2.2. Architecture Refinement



1. Identify modes
2. Identify threads
3. Map functions to threads
4. Add connections
5. Assign modes to threads
A1. Select System or Thread
new refinement?
yes

3A. Software Architecture Modeling




1. Requirements Definition
Hw Architecture
2. Environment Description
Sw Architecture Modelling
3. Operation Modes Definition
4. Behavior spec (Abstract or Detailed)
5. Verification
B1. Architecture Modelling
B2. Architecture Mapping
B3. Architecture Simulation
6. Real-Time Properties Definition
Inject behavior
7. Timing Verification