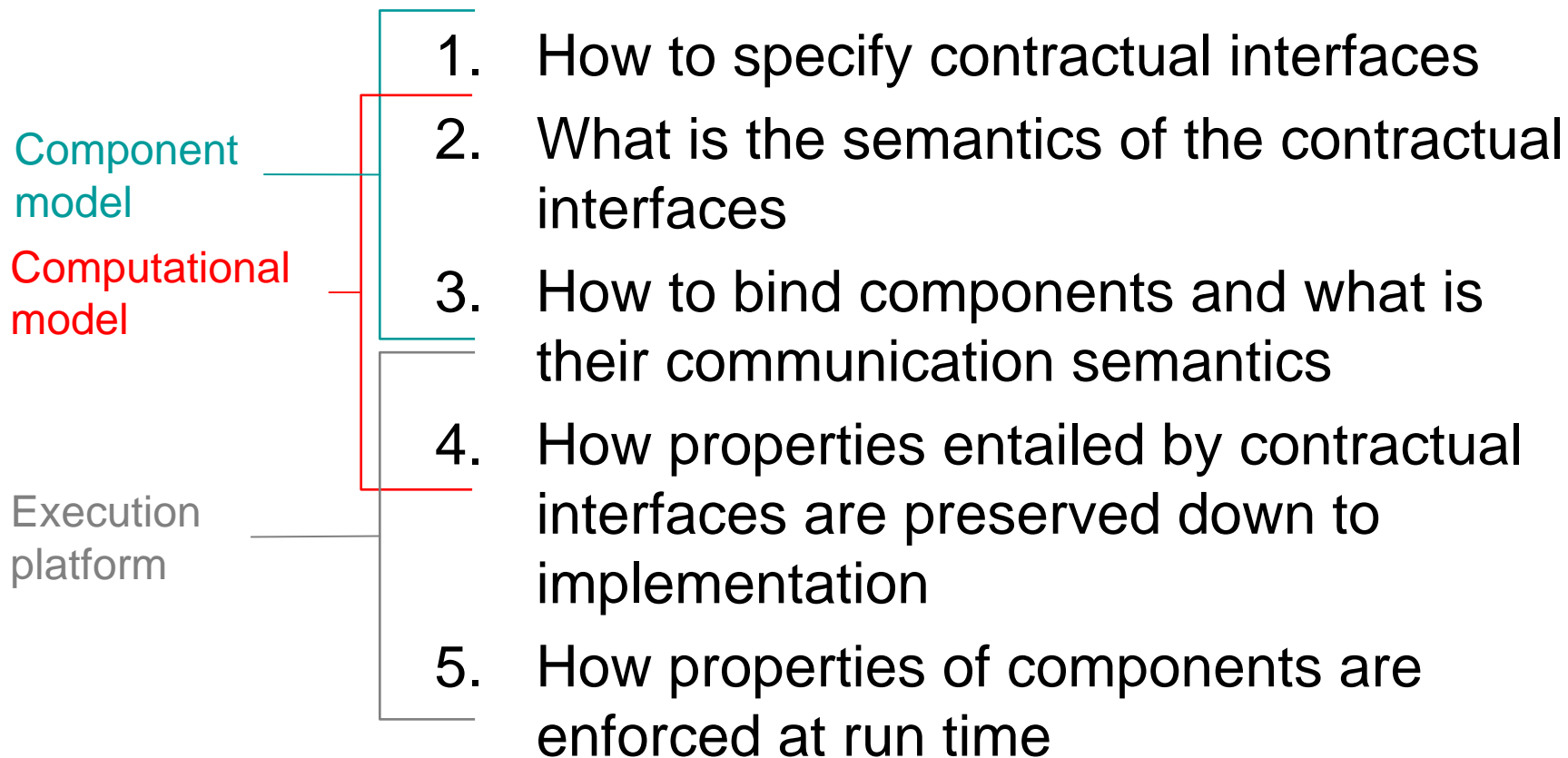


# ASSERT vision and impact on AADL

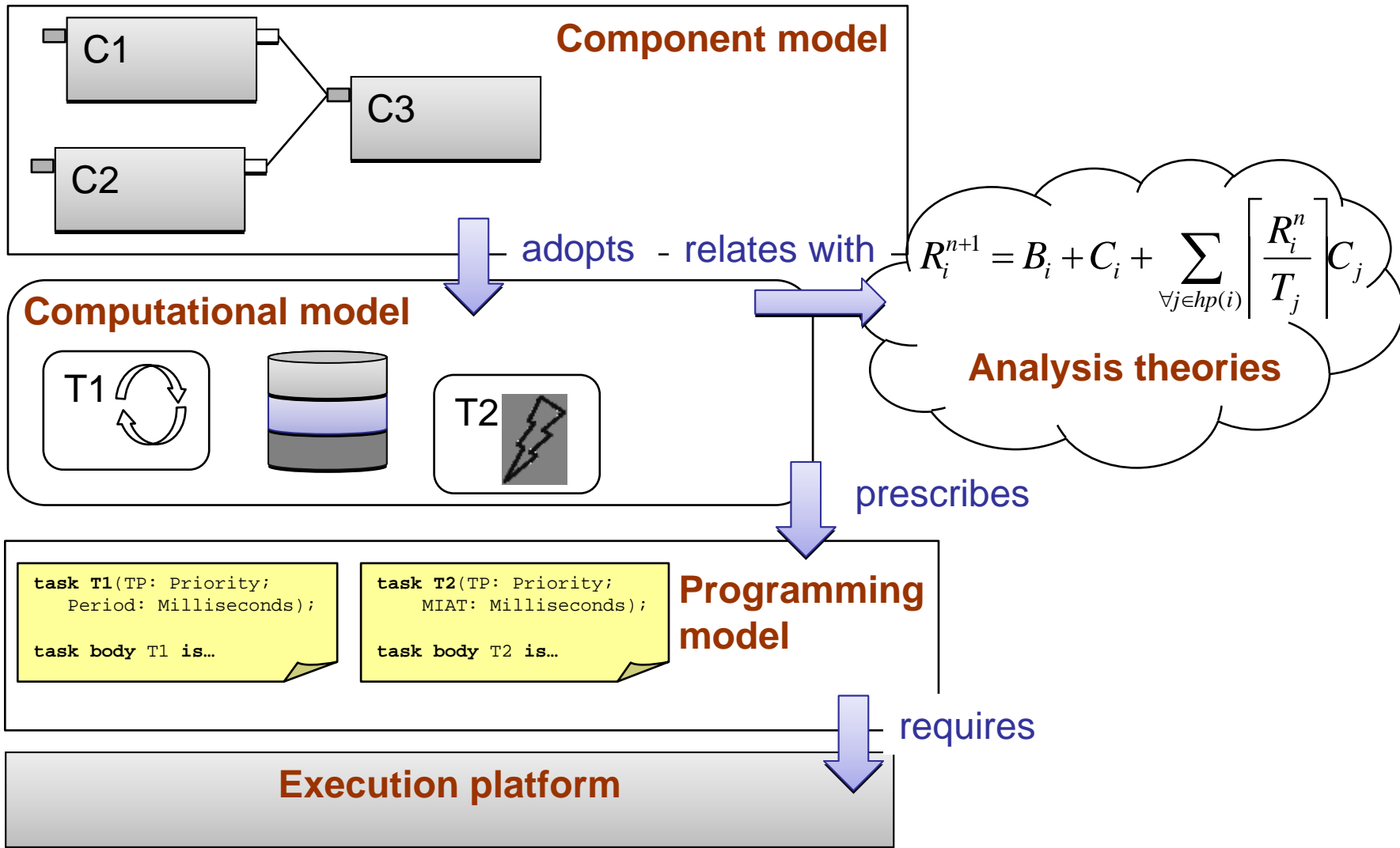
**Tullio Vardanega**  
*(University of Padua, Italy)*

# Games in town ...

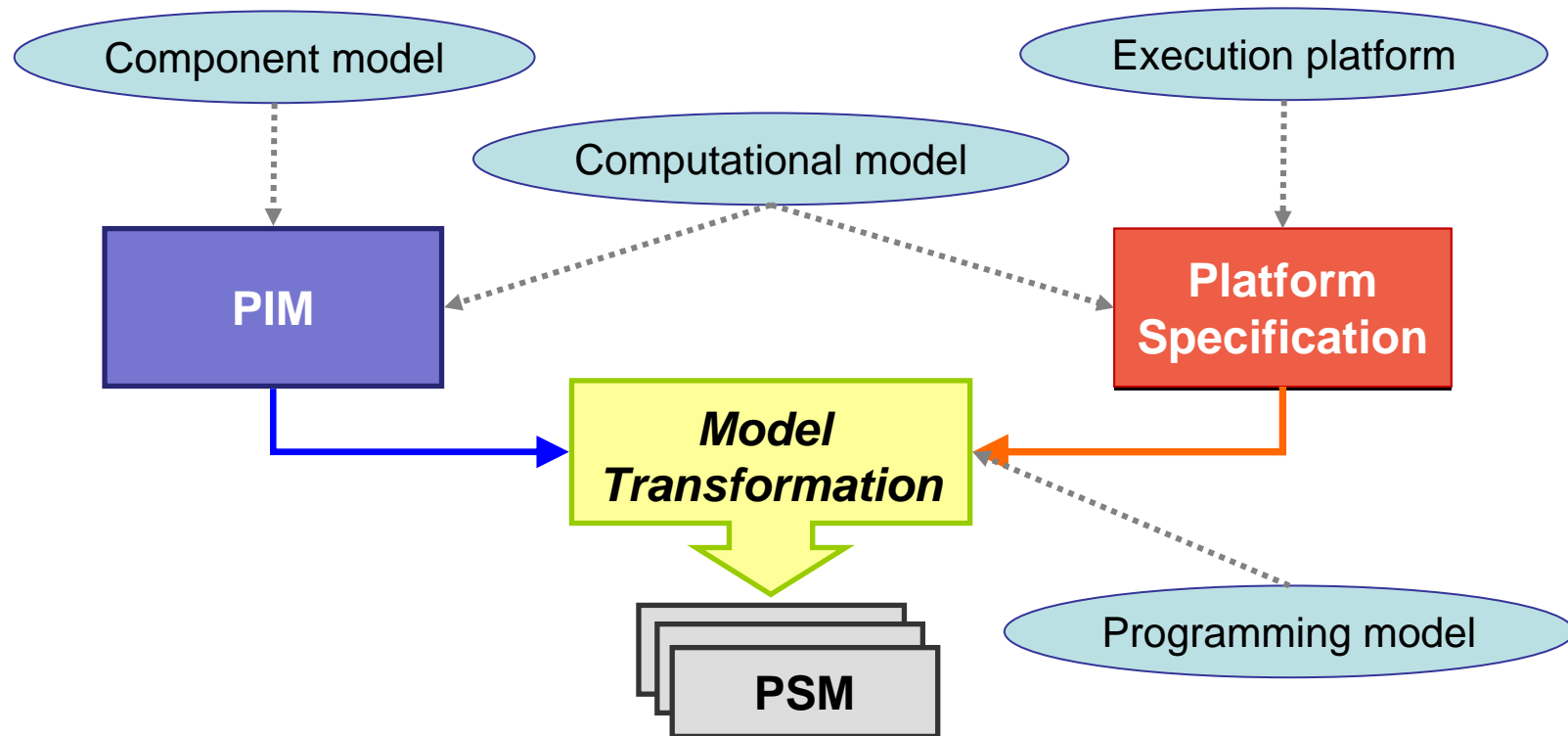


*A comprehensive, unified framework is much required*

# ... And their interplay



# MDE in context



*The computational model is key: we plant it in the core metamodel so that it informs all that is modeled off it*

# Foreground goals

- Separation of concerns
  - Impacts on development process and infrastructure
- Property preservation
  - Requires infrastructure with provable guarantees
- High integrity
  - Timing predictability
  - Isolation in time, space and communication
- Distribution transparency

# Background goals

- **Composability**
  - The properties of individual components should hold regardless of the presence of other components in the system
  - We want property preservation across model transformations as well as at run time!
    - Often overlooked
- **Compositionality**
  - The overall system behavior should be determinable as a (simple) equation combining the behavior of individual components
  - The equation is the system's and not the components'!
    - Often misunderstood

# Property preservation @ run time

- Ravenscar Computational Model as a concurrent computational model provably amenable to the chosen body of feasibility and sensitivity analysis
- A compiler that produces executable code for “legal” entities only
  - With as many off-line conformance checks as possible
- A run-time environment that only accepts and supports “legal” entities
- Run-time services that actively police that entities preserve their non-functional properties
  - Arrival model
    - Periodic arrivals are kept time accurate
    - Aperiodic arrivals are prohibited
    - Sporadic arrivals are kept apart by the stipulated minimum separation
  - Execution time and communication budget monitoring
  - Fault containment in time, space and communication

# Enforcement policies and issues

- **Arrival model**
  - **Easy**: the OBCS stores any activation events pushed in too frequently
  - **Downside**: latency commensurate to stipulated MIAT
- **Execution time monitoring**
  - **Easy**: built-in RCM support via execution-time timers and timing events
    - Single timer per task: modest, predictable and fully determinable overhead
    - Very little use for group budgeting: good replenishment policy unknown
      - Replenish on consumption or on budget?
  - **Difficult**: what to do on the notification of violations
    - A three-stepped escalation
      - Log event and integrate its frequency over time
      - Second chance using slack allowances confirmed by sensitivity analysis
      - Partition safe stop or system-wide mode change
- **Communication budget monitoring**
  - **Easy**: static allocation model, static communication paths
  - **Difficult**: as for execution time

# Impact on AADL – I

- ASSERT baseline set on AADL 1.0 as of November 2004
  - ESA, ENST, Ellidiss, Axlog behind the effort
- What was to be built?
  - Almost everything: modelers, code generators, tools for validation & verification

# Impact on AADL – II

- From AADL 1.0 to AADL v2
  - AADL v2 effort started in 2005 and finished in 2008
  - Effort lead by Ellidiss, ENST and Axlog
  - Mostly a clarification of some constructs and better semantics coverage for modeling

# Impact on AADL – III

- Revision to support abstract component types, prototypes, refinement-based modeling
- Two annexes (guidelines for specific modeling activities)
  - Programming language annex:  
how to interconnect AADL modeling entities, runtime and C or Ada code
  - Data modeling annex:  
how to model data types at model level

# Impact on AADL – IV

- New capabilities
  - Better integration of V&V in one conceptual simplified framework
    - AADL-based modeling patterns tailored to ASSERT-style modeling
  - Seamless code generation patterns for high-integrity systems targeted to C/RT-POSIX, C/RTEMS, Ada 2005/Ravenscar
    - No dynamicity at all
    - Generated source is amenable to analysis for WCET computation (Bound-T), code coverage (COUVERTURE) and code correctness (SPARK)