Modeling the Environment in Software-Intensive Systems

Carlo A. Furia, Matteo Rossi, and Dino Mandrioli

Dipartimento di Elettronica e Informazione
Politecnico di Milano

MISE Workshop @ ICSE 2007
May 19-20, 2007
Minneapolis, USA

Outline

- Software-Intensive Systems
- What to Model?
- What to Model Formally?
- Pushing Formalization Deep in the Environment
  - Caveats
- Example sketch: Jackson's Traffic System
- Conclusions

Software-Intensive Systems

- Software becomes pervasive
    - embedded
    - networked
    - heterogeneous
    - ...
- Software-Intensive System
    - software components
        - interact with
    - non-software components
        - from the physical world
            - e.g., mechanical, chemical, social, ...

Software-Intensive Systems

- Software interacting with Environment
- Properties of the environment
  - indicative
    - world as it is
  - optative
    - requirements
- Specification
- Software-engineering viewpoint
    - often SIS are controlled systems
    - but traditional control modeling techniques are not suited to model software **and** environment

What to Model?

- Environment
  - both world as it is
  - and requirements
- Software system
  - specification
- Their interaction
- E.g., reference (meta-)model
  - and derived meta-models

What to Model **Formally**?

- Problem:
  - how much
  - how deep
  - to formalize in a SIS model?
- In particular for the environment
  - (some) requirements intrinsically informal?
    - can't formalize much?
  - requirements are
    "deep in the environment" (M. Jackson)?
    - can formalization go deep too?
    - formalization gets very demanding easily?

What to Model Formally?

- Our view:
  - formalization
    - can be and
    - should be
  - pushed “deep in the environment”

Can Be Formalized

- Most application domains allow formalization
  - at least partially
  - even if they're considered intrinsically informal
  - even if the formalization may be complex and/or costly
- Scattered examples:
  - social organizations
  - psychology of choice
    - games, bounded rationality, etc.

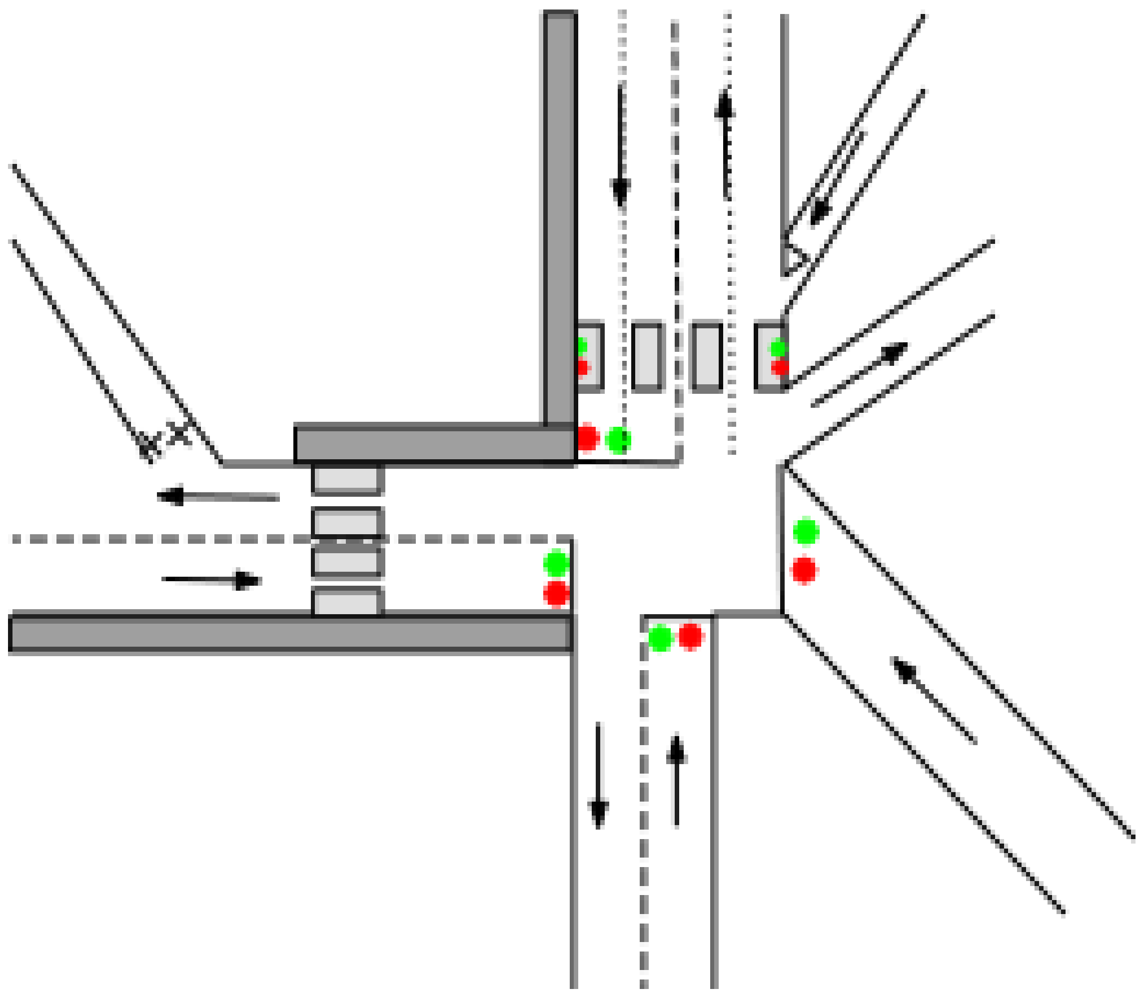
Should Be Formalized

- Formalization brings conspicuous benefits
  - early detection of errors and misunderstanding
  - better understanding of application domain
  - shows that more things can be formalized
  - ...
- The great cost/complexity is usually traded-off favorably against benefits

Caveats

- Formalization ameliorate several aspects, but it's no silver bullet
  - it doesn't replace completely non-formal approaches
  - better: incremental application of formalization
- Advantages and efforts depend on several factors:
    - context / application domain
    - goals
    - ...
  - Don't have to formalize **always** and **everything**

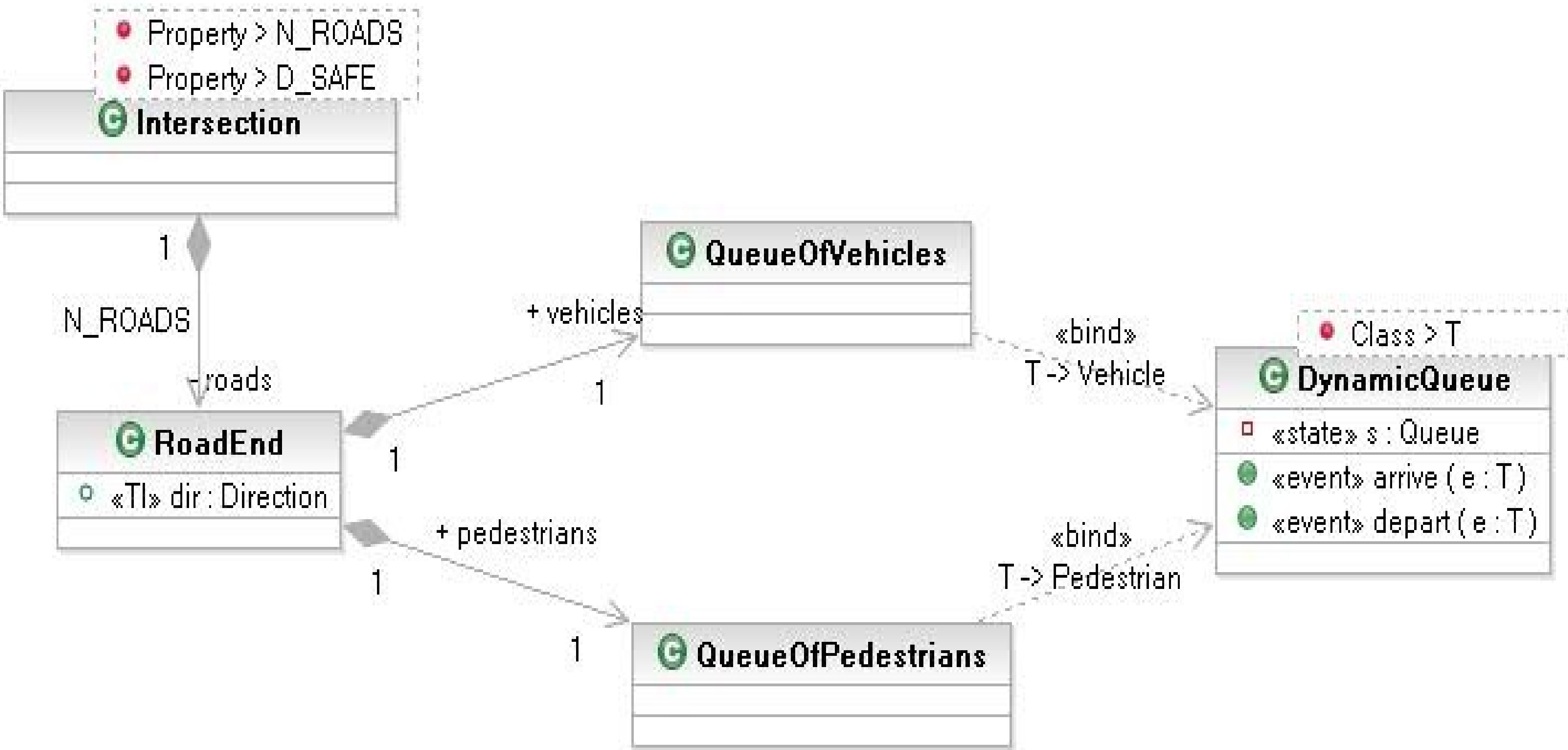
Example sketch:
Jackson's Traffic System



Example sketch: ArchiTRIO Formal Notation

- UML diagrams
  - system's components and structure
    - class diagrams
    - composite structure diagrams
- TRIO logic formulas
  - real-time temporal logic

Example sketch: Deep in the Environment



Example sketch: World Formalization

- Formalization of DynamicQueue
  - according to the need of our domain
  - Vehicles in queue cannot change their relative positions (i.e., no overtakes or U-turns when in queue)

$$\forall e: T \ (\ depart(e) \Rightarrow e = s.head \ )$$

Example sketch: World Formalization

- Formalization of DynamicQueue
  - according to the need of our domain
  - Behavior of elements in queue over time

$$\forall q: Queue[T]: (\neg q.isEmpty \Rightarrow$$
$$(s = q \;\Leftrightarrow$$
$$Since(\neg\exists e1: T\ (arrive(e1) \vee depart(e1)),$$
$$\exists e2: T, q': Queue[T]\ ($$
$$(arrive(e2) \wedge s = q' \wedge q = q'.enqueue(e2))$$
$$\vee$$
$$(depart(e2) \wedge s = q' \wedge q = q'.dequeue)))))$$

Example sketch: Requirements

- Formalization of "Orderly Safe Traffic"
    - based on world formalization
  - no two items coming from conflicting roads can flow into the intersection within a short timespan

$$\forall r1, r2: \text{RoadEnd} \; (\text{conflicting}(r1, r2)$$
$$\wedge \; \exists v1: \text{Vehicle} \; (r1.\text{vehicles}.\text{depart}(v1))$$
$$\Rightarrow$$
$$\neg \exists v2: \text{Vehicle} \; (\text{Within}(r2.\text{vehicles}.\text{depart}(v2), \text{TSAFE})) \wedge$$
$$\neg \exists p: \text{Pedestrian} \; (\text{Within}(r2.\text{pedests}.\text{depart}(p), \text{TSAFE})))$$

Example sketch: Possible Developments

- Other formalizations of “Orderly Safe Traffic”
  - more detailed
    - requiring the formalization of more elements
  - for more complex intersections
  - ...

Example sketch: Possible Developments

- How to meet the requirements?
  - e.g., traffic lights
    - with previous formalization of OST
    - add them to the formal model
    - early design decisions
  - new formalized elements may in turn prompt us to reconsider some previous assumptions
    - e.g., vehicles in a queue can change relative posititions when light is green
  - iterative process
- Formal argument that requirements are met

Conclusions

- Environment in Software-Intensive Systems
    - interacting with software components
- We can formalize significant portions
    - push formalization deep in the environment
- We should formalize significant portions
    - large effort, but usually pays off
    - even if it seems “intrinsically informal” at first
- Formalization improves development quality
    - not replacement but enhancement and complement