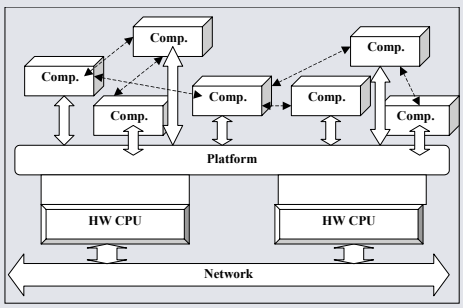
Platform Modeling and Analysis

Presented by
Tivadar Szemethy
ISIS, Vanderbilt University

Chess Review
May 11, 2005
Berkeley, CA

Model-Based Design with Components



Component-Based System

Design Model

components interacting according to a "Model of Computation"

Runtime Platform

abstraction of HW/SW/MW services

System Synthesis:

1. Map the design-time components into platform objects
2. Enforce interaction rules using platform services

Analysis model for verification

Behavior := System Model + Components
+ MoC semantics + Platform properties

- Verification:
  - does it satisfy requirements specification ?
- Need design requirements, in terms of
  - observable MoC events (mapped to Platform)
  - Platform quantities (resources)

Analysis necessitates Platform-level model

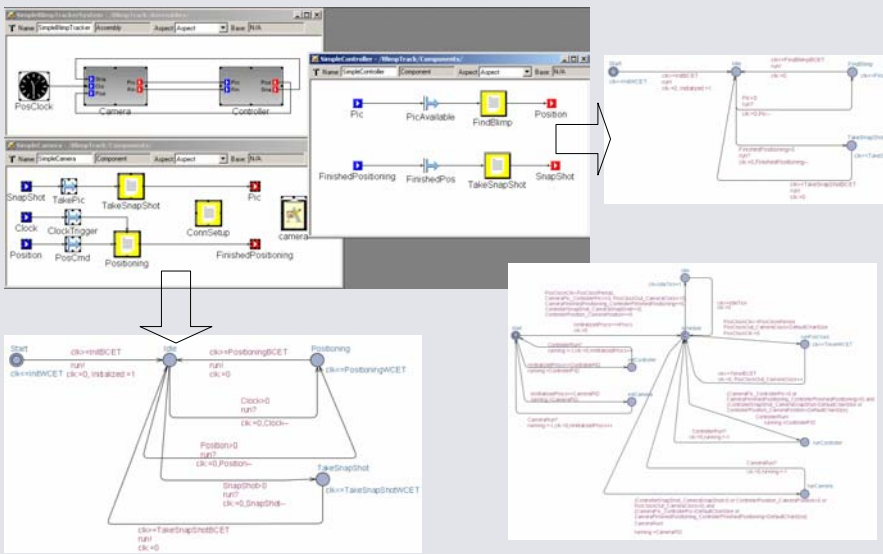
Platform-level analysis model

- Purpose
  - formal verification (ideal)
  - simulation (at least)
- Language for analysis model:
  - SMV/SPIN model, Timed/Hybrid Automata…
  - Simulink…
- To be automatically constructed based on
  - MoC semantics (formal, well-defined)
  - Platform properties (?)

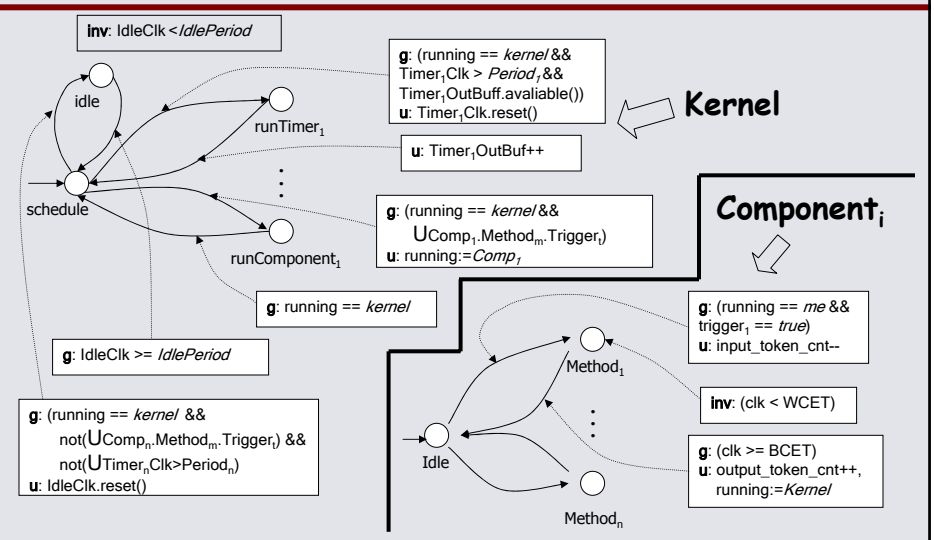
Example: SMOLES -> UPPAAL 1.

- Application: Camera tracking an object
- Simple MOdeling Lang. for Embedded Sys.
  - Dataflow-oriented DSML in GME
  - Components, Ports, Triggers, Timers, Methods
- Platform: DataFlow Kernel
  - simple OO asynch. dataflow engine in C++/Java
  - SMOLES model interpreter generates code
- Analysis model: UPPAAL Timed Automata

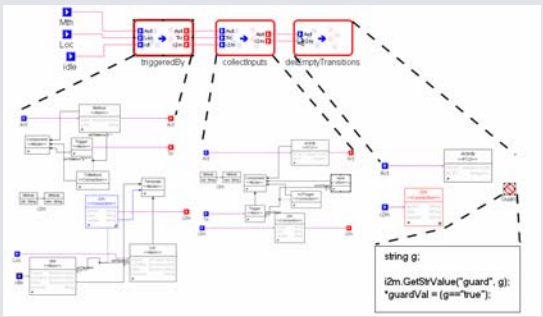
Example: SMOLES -> UPPAAL 2.



Example: SMOLES -> UPPAAL 3.



Example: SMOLES -> UPPAAL 4.



Translation using
graph transformations

GReAT graph rewriting tool (GME add-on)

Translation ruleset
- input: SMOLES model
- output: UPPAAL TA
- implicit: DFK "internals"

Platform-level model
- one TA per component
- one TA for Kernel

Lessons learned

results published in:

"Platform Modeling and Model Transformations for Analysis"

in Journal of Universal Computer Science vol. 10, pp. 1383-1407, 2004

UPPAAL analysis model too restrictive
- no preemptive scheduling
- only for timing analysis
- no higher-level structures

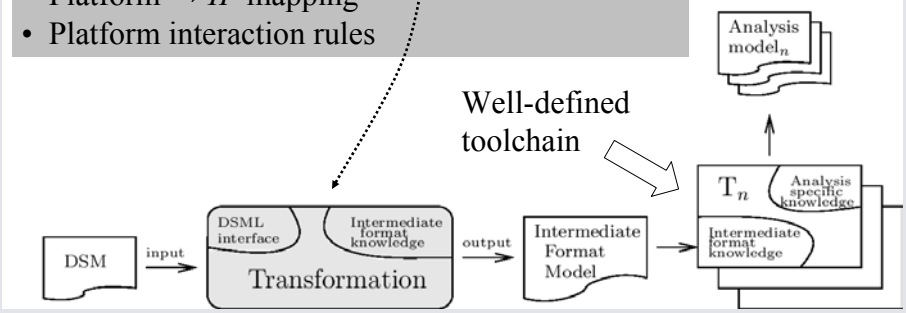
"Intermediate format"
- e.g. IF or Metropolis
- higher-level language
- provides mapping to multiple analysis tools

DFK model was implicit in transformation:
- complex, monolithic transformation spec.

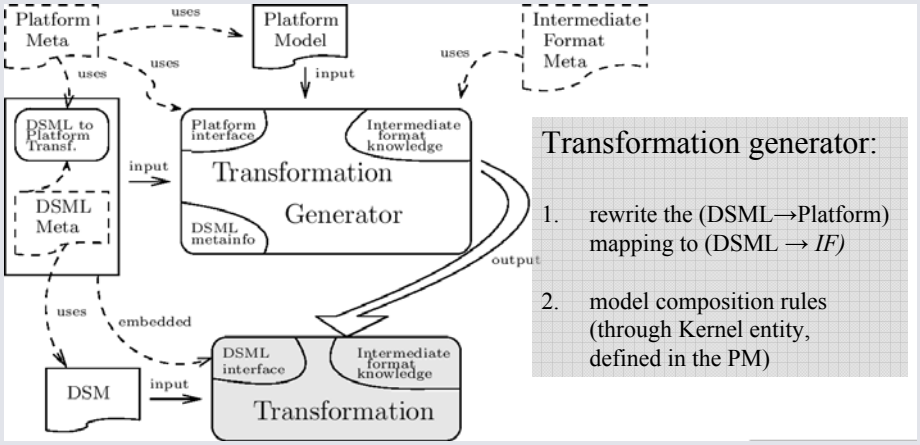
Platform model as
1) "Skeleton" for
   1) Components
   2) Kernel
2) Rules to construct synchronizers/guards
3) Composition

Generating the analysis model

The DSML → IF transformation needs to "know":
- DSML → Platform mapping
- Platform → *IF* mapping
- Platform interaction rules



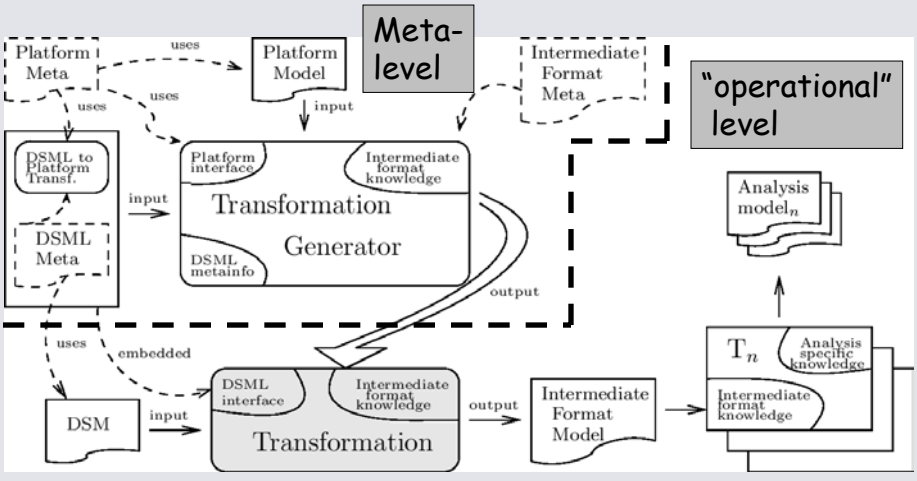
Generating the transformation



Transformation generator:

1. rewrite the (DSML→Platform) mapping to (DSML → *IF)*
2. model composition rules (through Kernel entity, defined in the PM)

The "Big Picture"



Preliminary results

- Formalizing the Platform
  - GME metamodel for DFK
  - SMOLES → DFK transformation specified as graph transformation in GReAT
- In search of an Intermediate Format
  - Evaluating VERIMAG's IF framework (metamodel, simple translators and examples)
- Started working on modeling
  - Giotto with E-machine