Safe Design of High-Performance Embedded Systems in an MDE framework

Huafeng Yu
Abdoulaye Gamatié
Éric Rutten
Jean-Luc Dekeyser

INRIA - CNRS - LIFL - USTL

Outline

Introduction

Repetitive computation and control modeling

Formal application validation and a case study

Conclusions

Introduction

High-performance embedded systems (HPES)

- High-performance computing in embedded systems
- Examples: signal/image/video processing
  - High-definition TV, Personal Digital Assistant, digital camera, multimedia cellular phone, etc.
- **Motivations**: need of design environment to address issues of increasing system complexity and design reliability
  - Automatic code generation, performance evaluation, validation, etc.

Gaspard2 framework for the design of HPES

Characteristics

- Hardware/software co-design
- Model-Driven Engineering approach
  - High-level modeling, automatic code generation, etc.
- MARTE profile
- Core formalism for high-performance computing
  - System regularity
  - Multidimensional array
- Performance evaluation, etc.

Needs

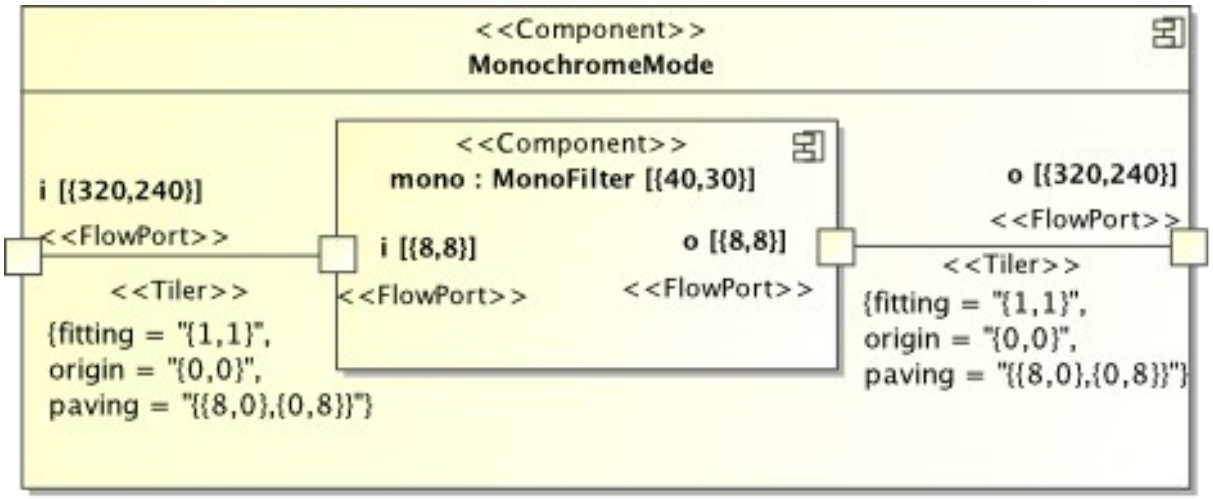
- Control mechanism and profile
- Application validation

Gaspard2 repetitive computation and control modeling

An example of Gaspard2 application

Gaspard2 basic concepts

- Elementary components: atomic functions
- Hierarchical components: task parallelism
- Repetitive components: data parallelism



Gaspard2 control

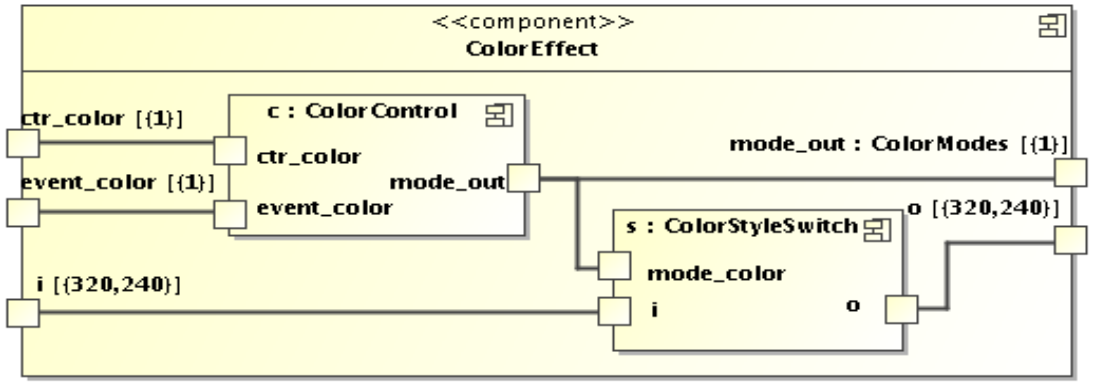
Requirements

- High-level state-based control
- Partial regularity preservation
- Verifiable and safe control
- Compatibility with MARTE

Control profile in accord with MARTE (1)

Main control concepts

- Computation mode
- Controller
  - Determine which computation is chosen according to its internal state
- Switch
  - Chose the right computation according to controller's decision



Two UML concepts for behavioral modeling:

- State machines
- Collaborations

Control profile in accord with MARTE (2)

UML State machines

- An object-oriented variant of State charts
- Explicit description of the behavior of systems

Main concepts of State machines

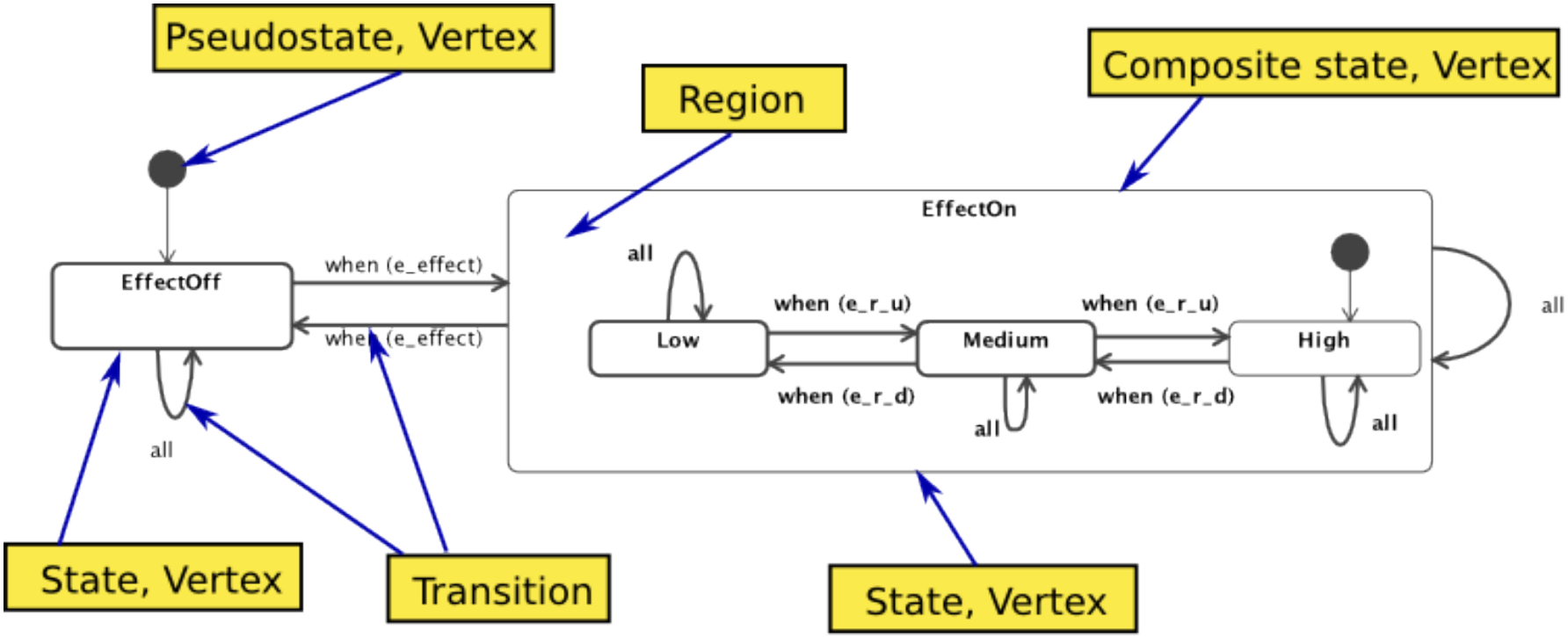
- StateMachine, Region, Vertex, State, Pseudostate, Transition, Trigger, Event, Expression, etc.

Usage specialization of state machines

- Event
  - *ChangeEvent* (prefixed by *when*) and *AnyEvent* (*all*)
- State
  - *DoActivity*
    - Expressions for the specification of an *OpaqueBehavior*

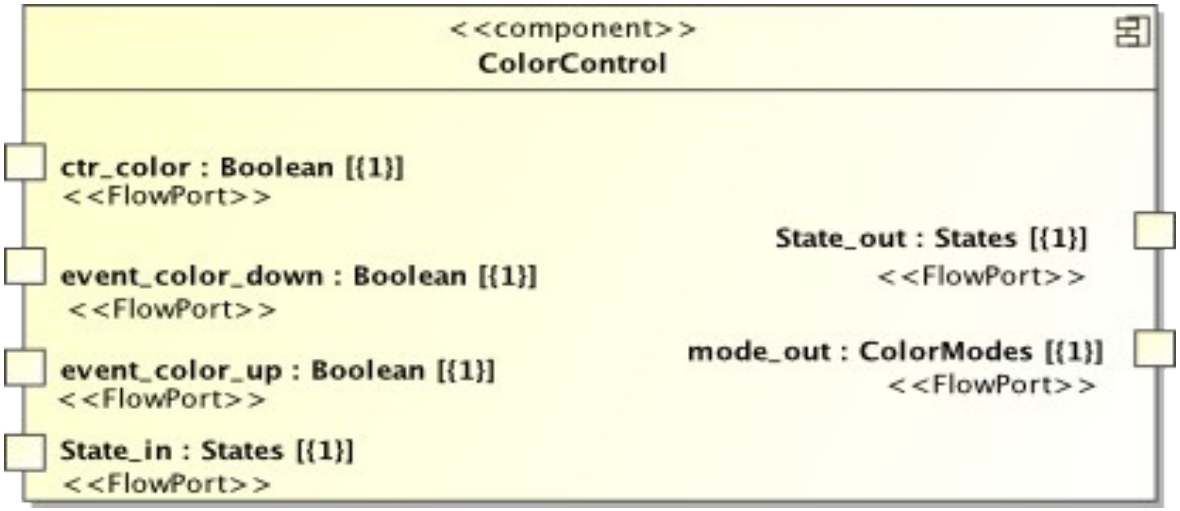
Control profile in accord with MARTE (3)

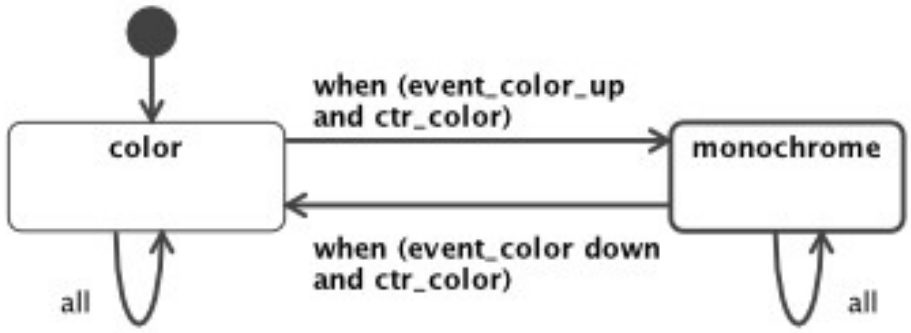
An example of State machine:



Control profile in accord with MARTE (4)

State machine component: an example





Control profile in accord with MARTE (5)

UML collaborations

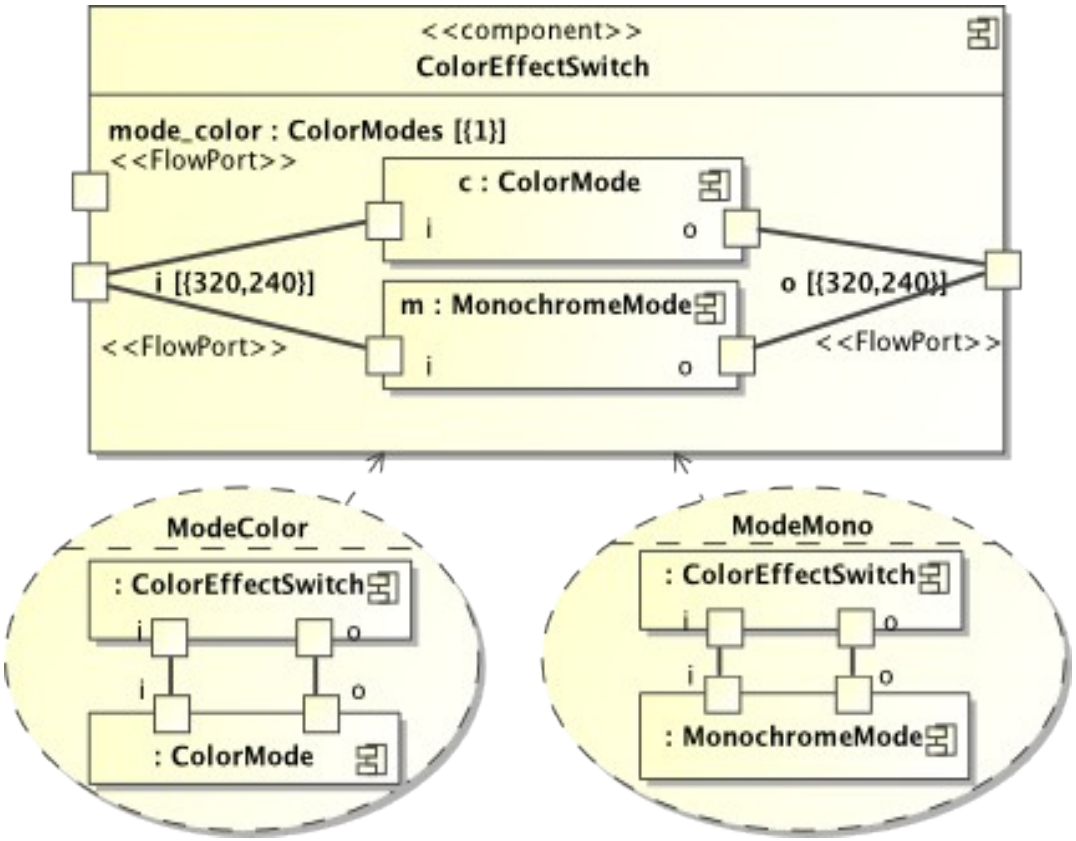
- A collaboration specifies the relationship between collaborating elements from a certain point of view

Mode switch component

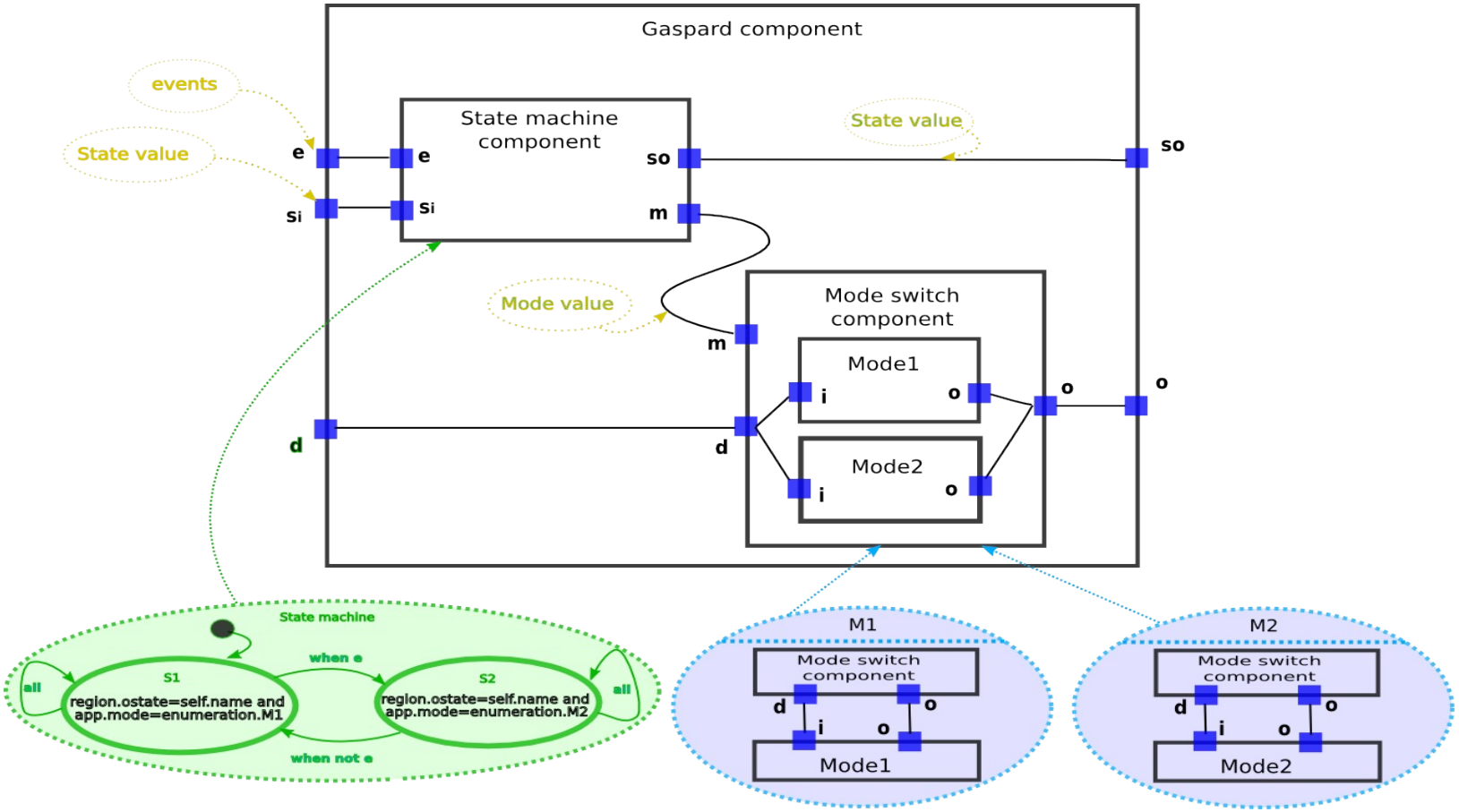
- Two types of ports
  - Mode port (receive mode values, a UML behavior port)
  - Data port (receive or send data)
- Change of its internal computations according to received mode values
- Correspondence between mode, mode value and collaboration
  - A collaboration specifies a mode, which is identified by a mode value
  - A collaboration is named according to a mode value

Control profile in accord with MARTE (6)

Mode switch component: an example



A typical composition



Validation requirements

UML-related verifications

Gaspard2-related verifications

- Safe array assignment, data dependency analysis, etc.

Application-related verifications

- Properties to be verified
  - Functional properties
  - Non-functional properties

Formal validation and a case study

Synchronous languages for validation

Synchronous languages

- Strong mathematical foundations
- Unambiguous specifications
- Languages: Esterel, Lustre, Signal, etc.
- Large number of associated validation tools
- Platforms: Esterel Studio, Scade, RT-Builder, etc.

Model transformation

- From the Gaspard2 model towards the synchronous equational model
- Prototype tool as an Eclipse plugin

Model checking

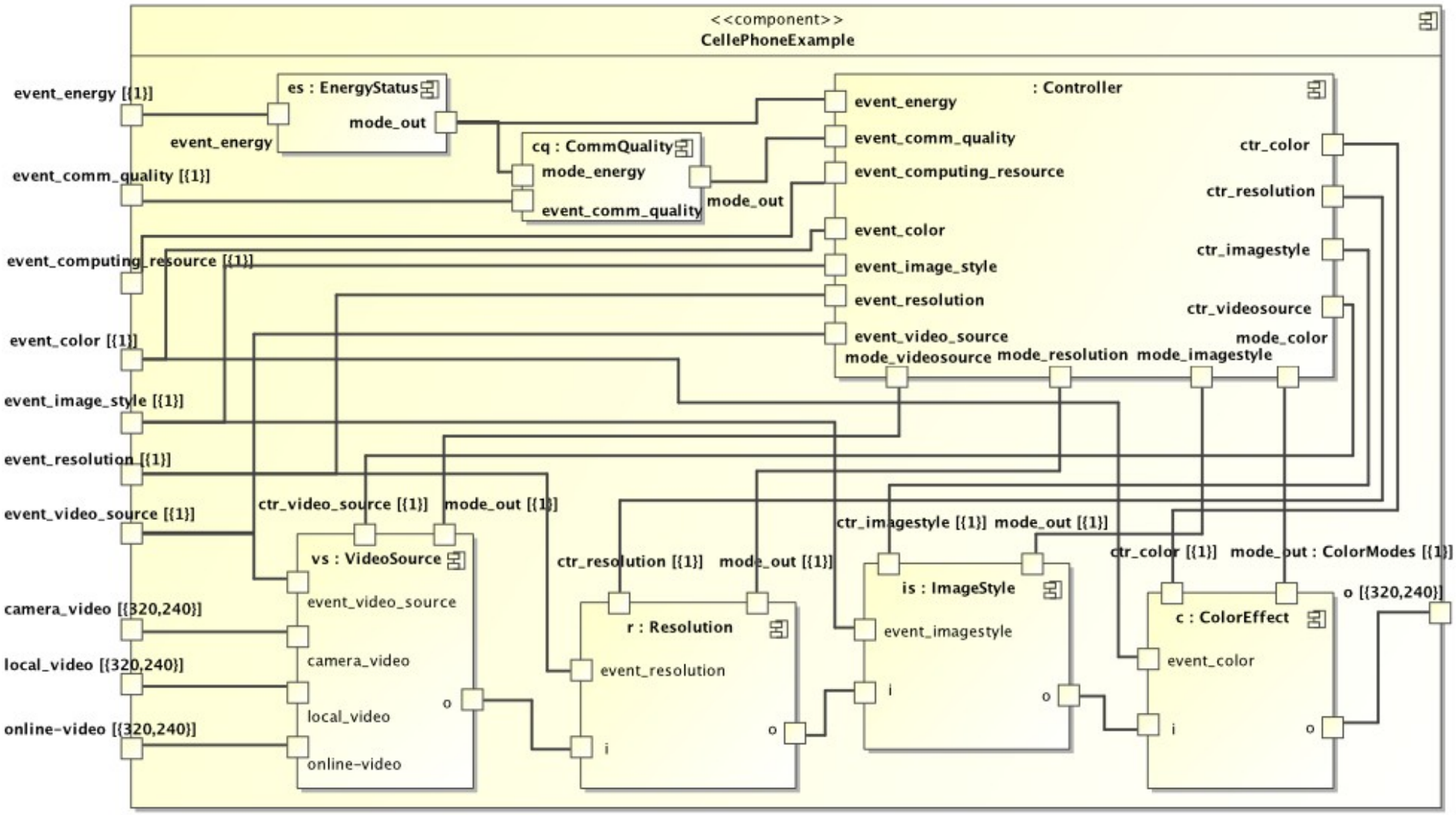
Model checking of functional properties

- Properties to be verified
  - Reachability, safety, ...
- Reachability verification example
  - Black & white state and color state

Model checking of non-functional properties

- Non-functional properties
  - Energy, communication quality, processor load, memory usage, etc.
- Properties to be verified
  - Reachability, ...
- Verification example
  - Reachability verification under energy constraints

A case study

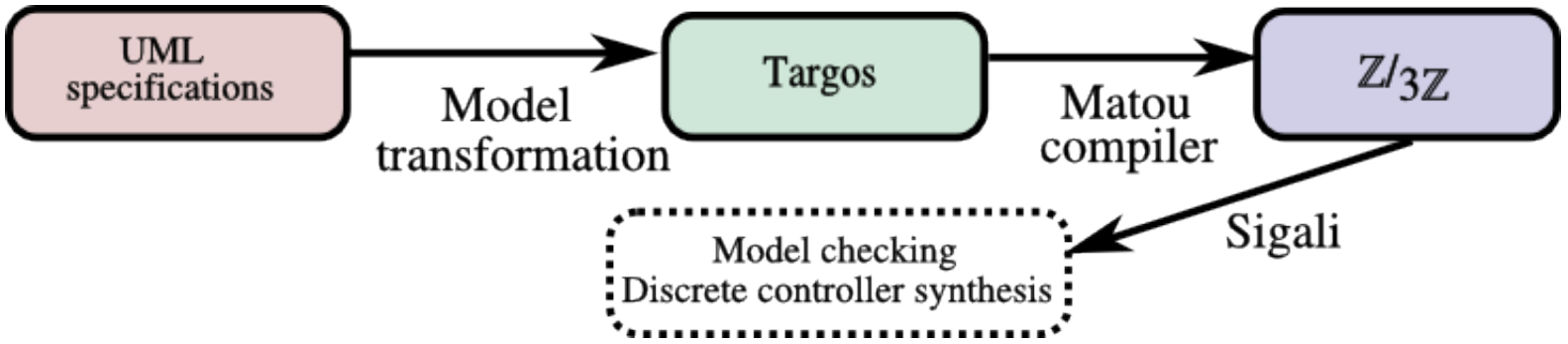

<<component>>
CellePhoneExample
event_energy [{1}]
es : EnergyStatus
event_energy
mode_out
event_comm_quality [{1}]
cq : CommQuality
mode_energy
event_comm_quality
mode_out
event_computing_resource [{1}]
event_color [{1}]
event_image_style [{1}]
event_resolution [{1}]
event_video_source [{1}]
: Controller
event_energy
event_comm_quality
event_computing_resource
event_color
event_image_style
event_resolution
event_video_source
ctr_color
ctr_resolution
ctr_imagestyle
ctr_videosource
mode_color
mode_videosource
mode_resolution
mode_imagestyle
ctr_video_source [{1}]
mode_out [{1}]
vs : VideoSource
event_video_source
camera_video
local_video
online-video
o
camera_video [{320,240}]
local_video [{320,240}]
online-video [{320,240}]
ctr_resolution [{1}]
mode_out [{1}]
r : Resolution
event_resolution
i
o
ctr_imagestyle [{1}]
mode_out [{1}]
is : ImageStyle
event_imagestyle
i
o
ctr_color [{1}]
mode_out : ColorModes [{1}]
c : ColorEffect
event_color
i
o
o [{320,240}]

Model checking

An example of model checking

```
CE_Color: a_var(Color, 0, 30, 0);
CE_Monochrome: a_var(Monochrome, 0, 20, 0);
CE_ColorEffect: CE_Color + CE_Monochrome;
CE_GL: CE_ColorEffect + CE_ImageStyle + ...;
MAX_En: 110;
CE_Limitation: a_sup(CE_GL, CE_MAX);
Reachable(S , CE_Limitation);
```

Tools used in the model checking



Conclusions

Conclusions

A MARTE-compatible control profile

- On the basis of UML state machines and collaborations

Model transformation into synchronous languages

- Manual transformation of the control into synchronous languages
- Extension of the synchronous model
- Extension in consideration of the presented control mechanism

Formal validation

- In consideration of non-functional properties
- Illustration through a case study

Thank you for your attention!