WebGME-BIP: A Design Studio for Modeling Systems with BIP

Anastasia Mavridou, Joseph Sifakis, and Janos Sztipanovits

INSTITUTE FOR SOFTWARE INTEGRATED SYSTEMS

VANDERBILT UNIVERSITY

Verimag

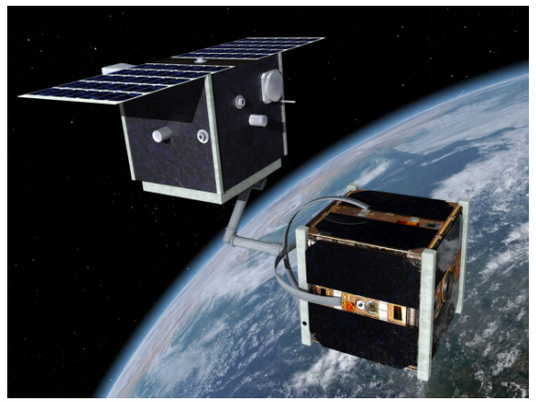
Why BIP?

- A language and tool-set for component-based system design
  - formal semantics
  - high expressiveness with small number of notions
  - code generation, simulation and verification tools
- BIP allows to **compositionally**
  - analyze existing applications
  - develop correct-by-construction applications
- Developed and maintained
  - by Verimag and RiSD, EPFL
  - directed by Joseph Sifakis

A. Basu, S. Bensalem, M. Bozga, J. Combaz, M. Jaber, T.-H. Nguyen, and J. Sifakis, "Rigorous component-based system design using the BIP framework," Software, IEEE, vol. 28, no. 3, pp. 41–48, 2011.

BIP applications

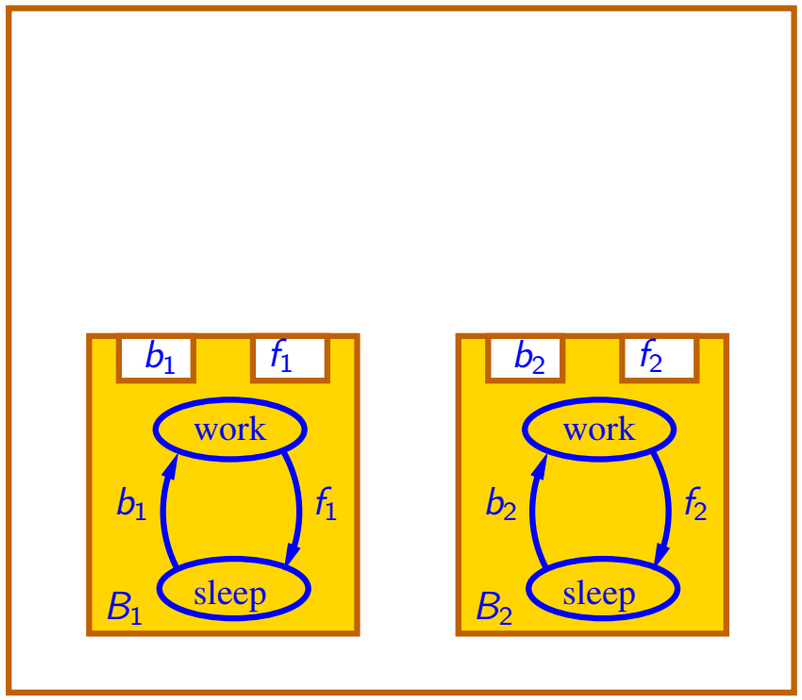
- Development of current-by-construction satellite software
  - 49 safety properties enforced by construction
  - Compositional verification of deadlock-freedom with D-Finder
    - State space size: $> 3^{10} \times 4$
    - Verification time: < 2 minutes



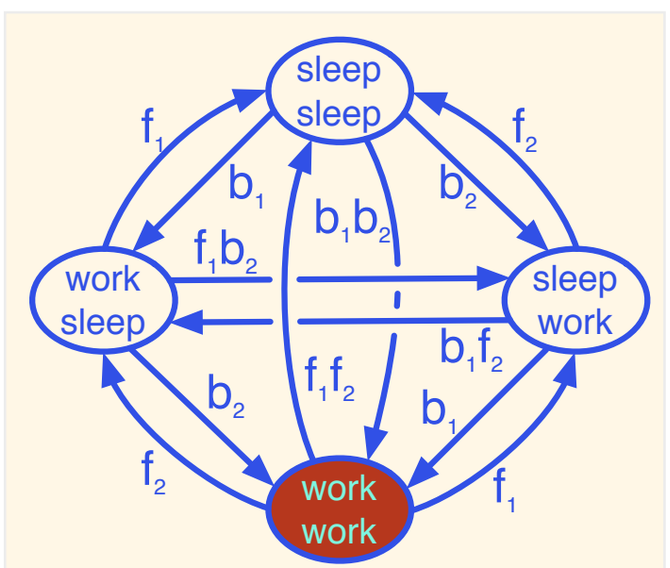
- Development of the Dala robot controller
  - > 500,000 lines of code
  - Example results of deadlock-freedom analysis with D-Finder:

Module	BIP LoC	C/C++ LoC	Estimated state space size	Verification time (minutes)
LaserRF	5,343	51,653	$2^{20} \times 3^{29} \times 34$	1:22
Rflex	8,244	57,442	$2^{34} \times 3^{35} \times 1045$	9:39
Antenna	1,645	16,501	$2^{12} \times 3^{9} \times 13$	0:14

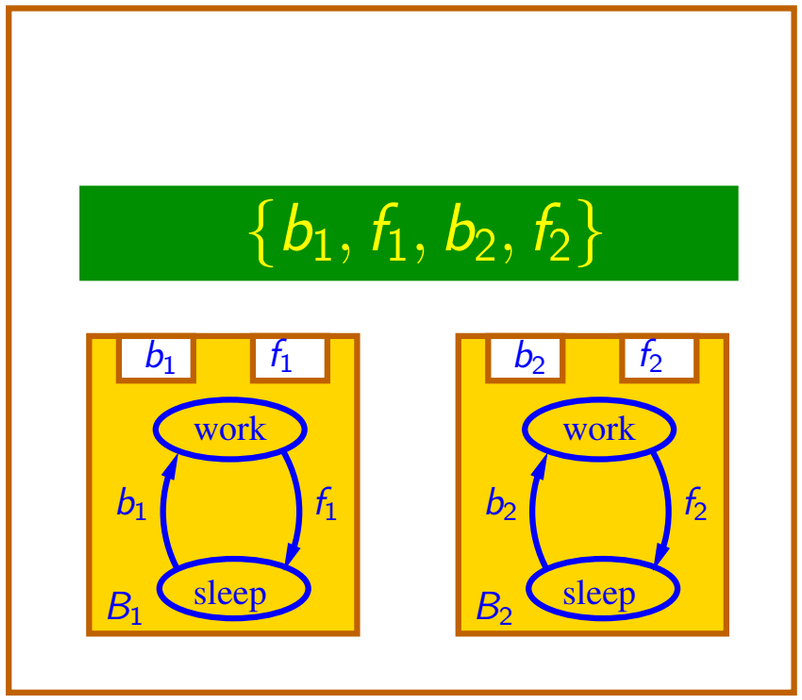
BIP by example



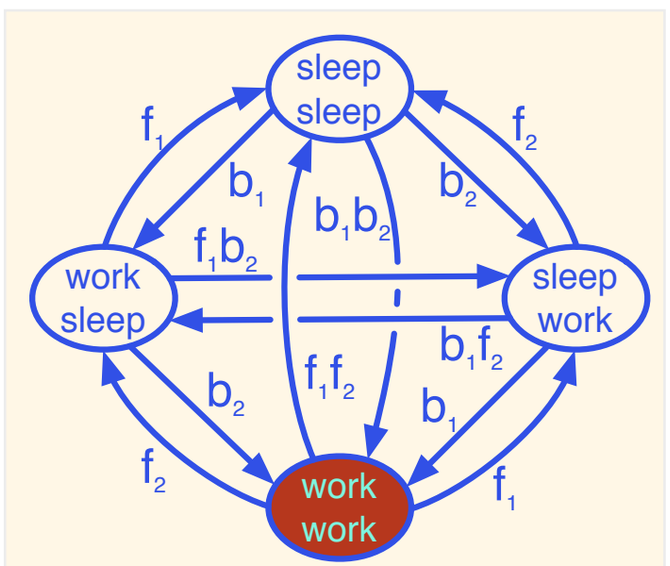
No restrictions



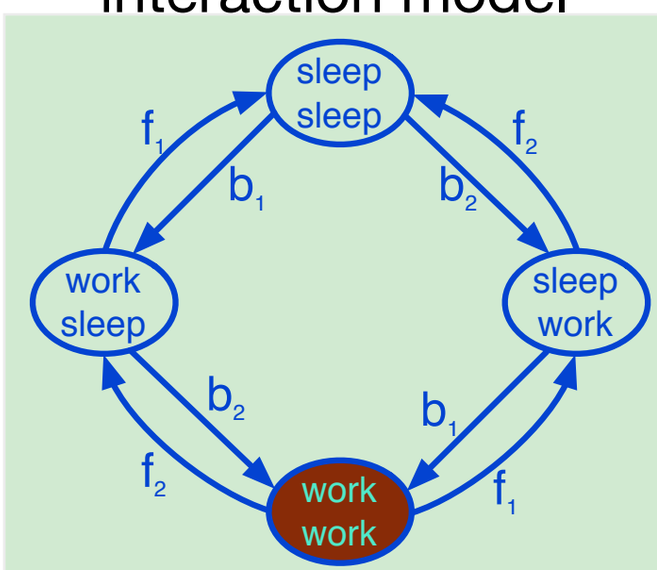
BIP by example



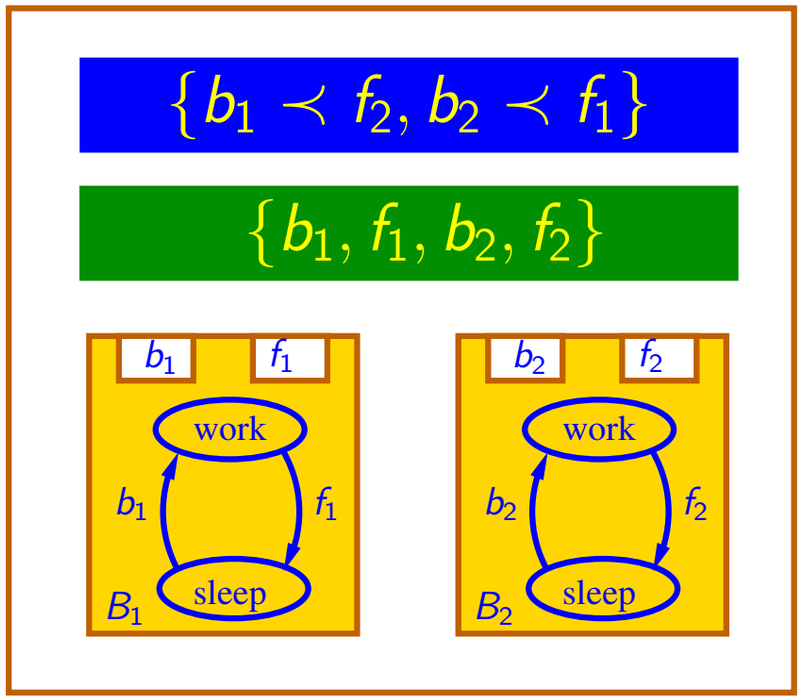
No restrictions



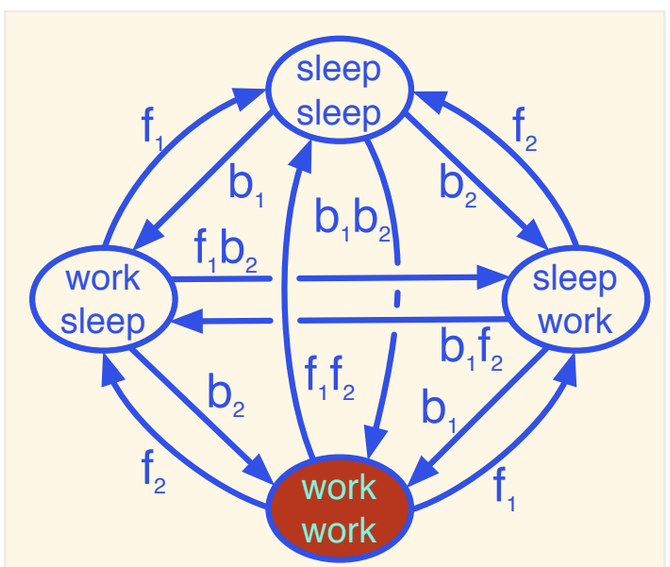
Restrictions from the interaction model



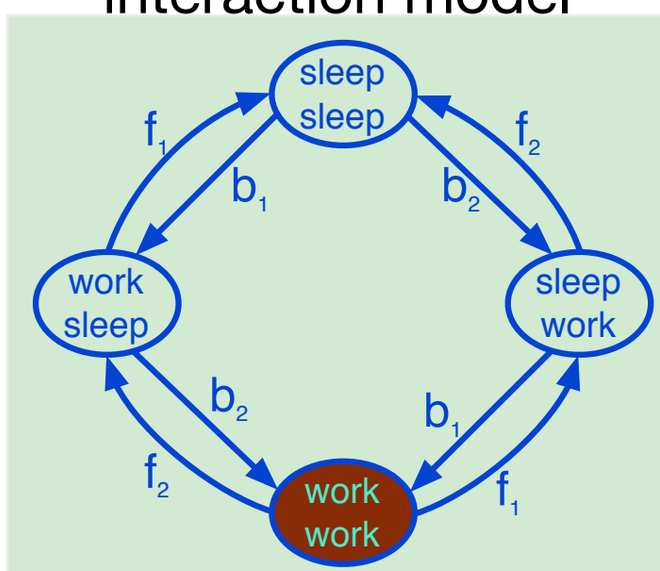
BIP by example



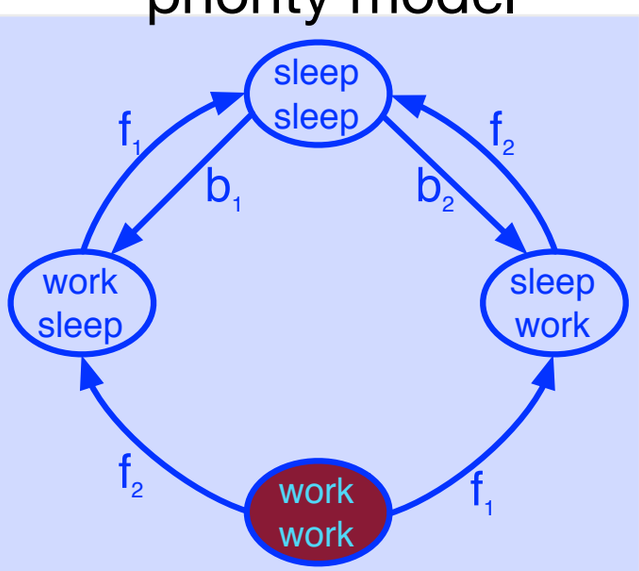
No restrictions



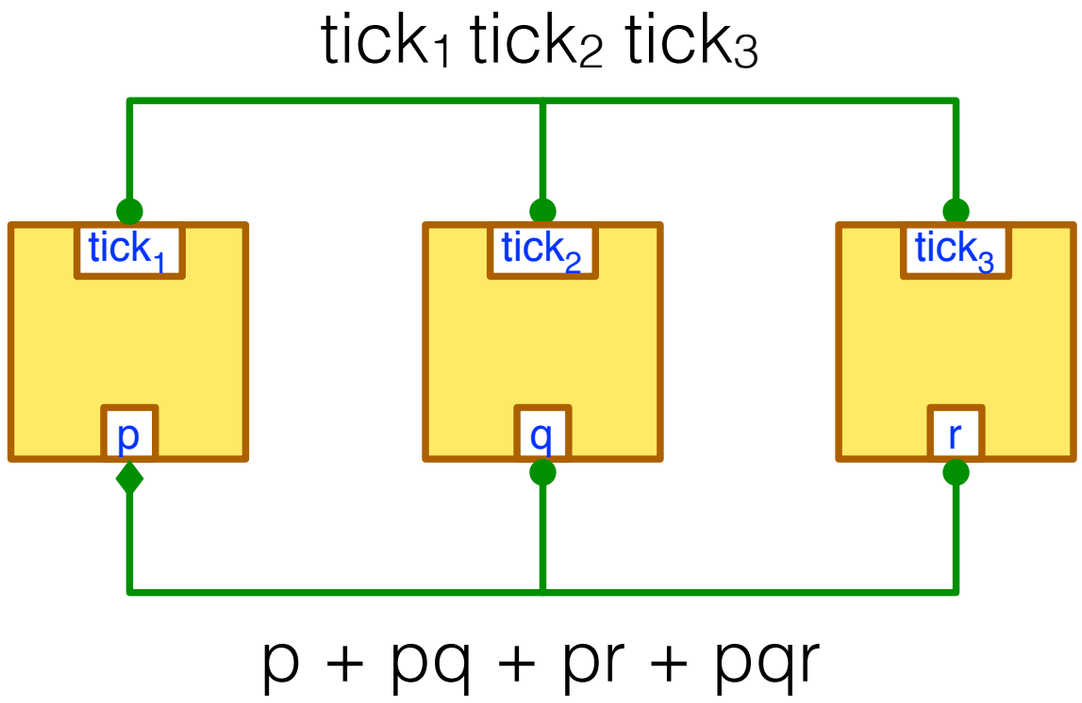
Restrictions from the interaction model



Restrictions from the priority model

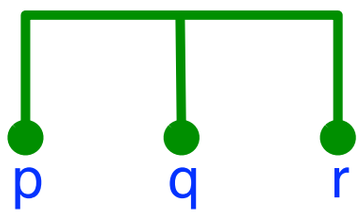


BIP Connectors

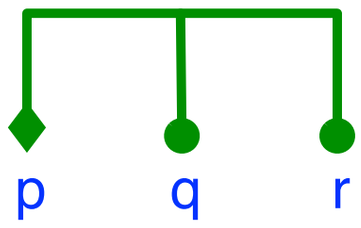


- *Connectors* are tree-like structures
  - ports as leaves and nodes of two types
    - *Triggers* (diamonds) — nodes that can “initiate” an interaction
    - *Synchrons* (bullets) — nodes that can only “join” an interaction initiated by others

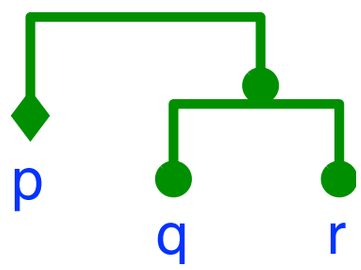
Connector examples



Strong synchronization: pqr



Broadcast: p + pq + pr + pqr



Atomic broadcast: p + pqr

Enumerative BIP specification

```
connector type Broadcast(HelloPort_t
p, HelloPort_t q, HelloPort_t r)
    define p' q r
    on p q r
    on p q
    on p    r
    on p
```

Symbolic BIP specification

$$(q \implies p) \bigwedge (r \implies q) \bigwedge (p \implies true)$$

p Requires $-$

p Accepts q, r

q Requires p

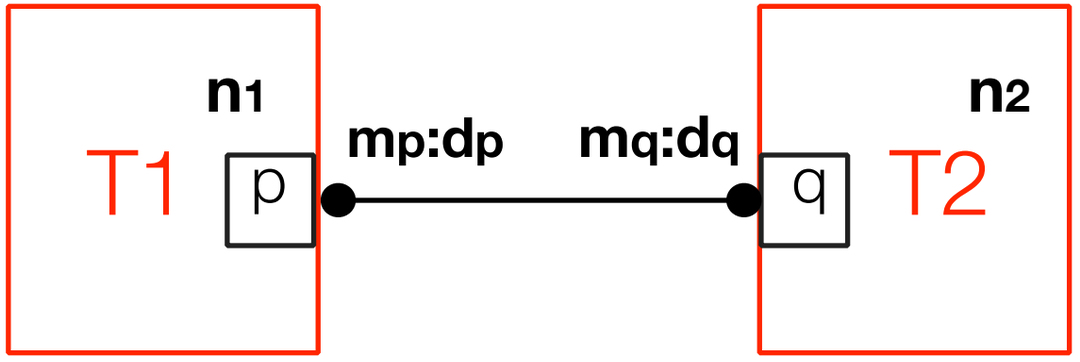
q Accepts p, r

r Requires p

r Accepts p, q

Architecture diagrams

- An architecture diagram consists of:
  - component types
    - port types
  - cardinality
  - connector motifs
    - degree
    - multiplicity

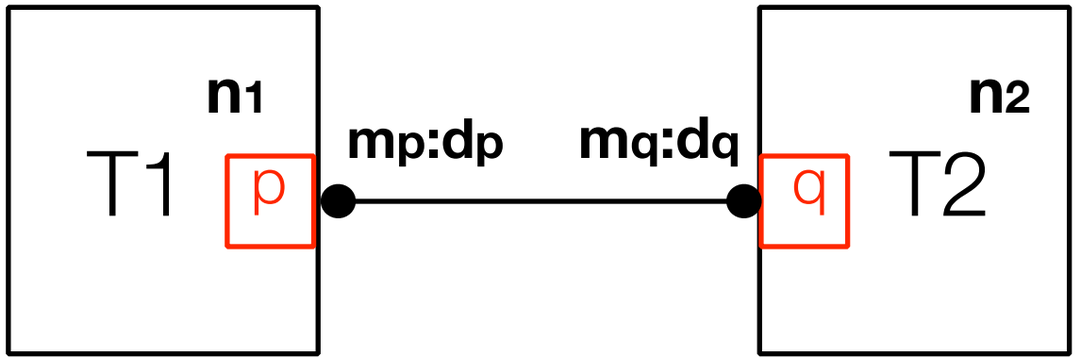


A. Mavridou, E. Baranov, S. Bliudze, and J. Sifakis, “Architecture diagrams: A graphical language for architecture style specification,” 9th Interaction and Concurrency Experience, 2016.

A. Mavridou, E. Stachtiari, S. Bliudze, A. Ivanov, P. Katsaros, and J. Sifakis. "Architecture-Based Design: A Satellite On-Board Software Case Study." 13th Formal Aspects of Component Software, 2016.

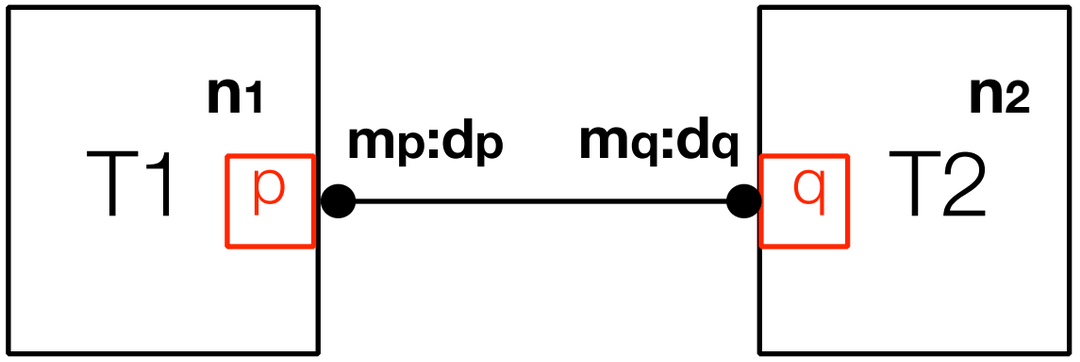
Architecture diagrams

- An architecture diagram consists of:
  - component types
    - port types
  - cardinality
  - connector motifs
    - degree
    - multiplicity



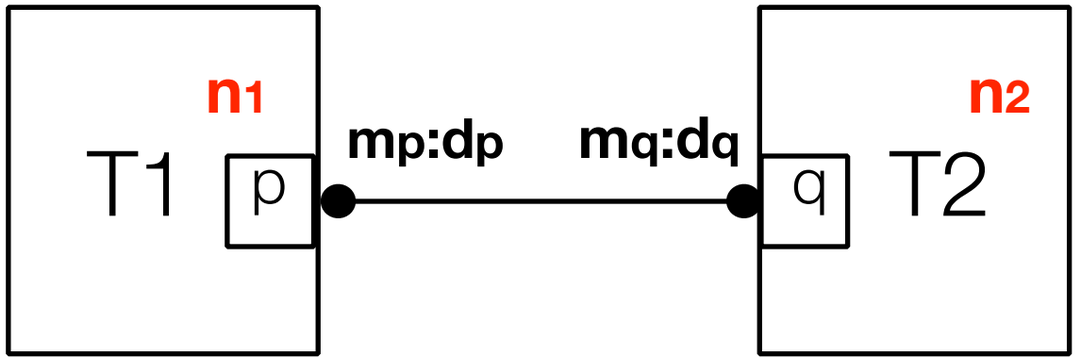
Architecture diagrams

- An architecture diagram consists of:
  - component types
    - port types
  - cardinality
  - connector motifs
    - degree
    - multiplicity



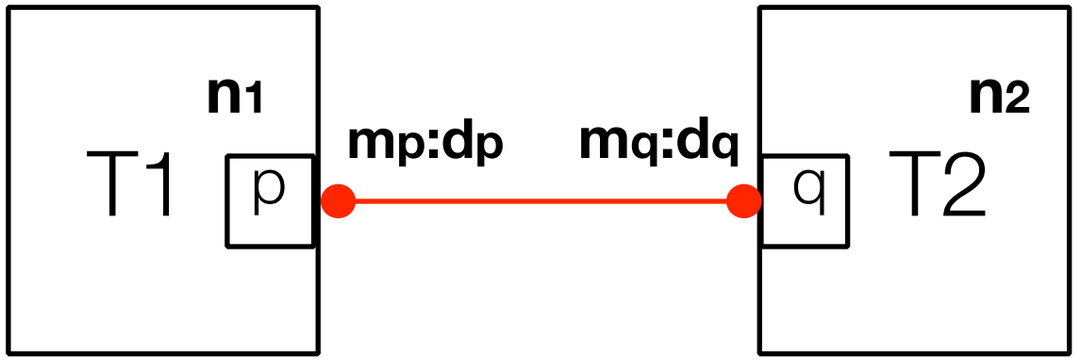
Architecture diagrams

- An architecture diagram consists of:
  - component types
    - port types
  - cardinality
  - connector motifs
    - degree
    - multiplicity



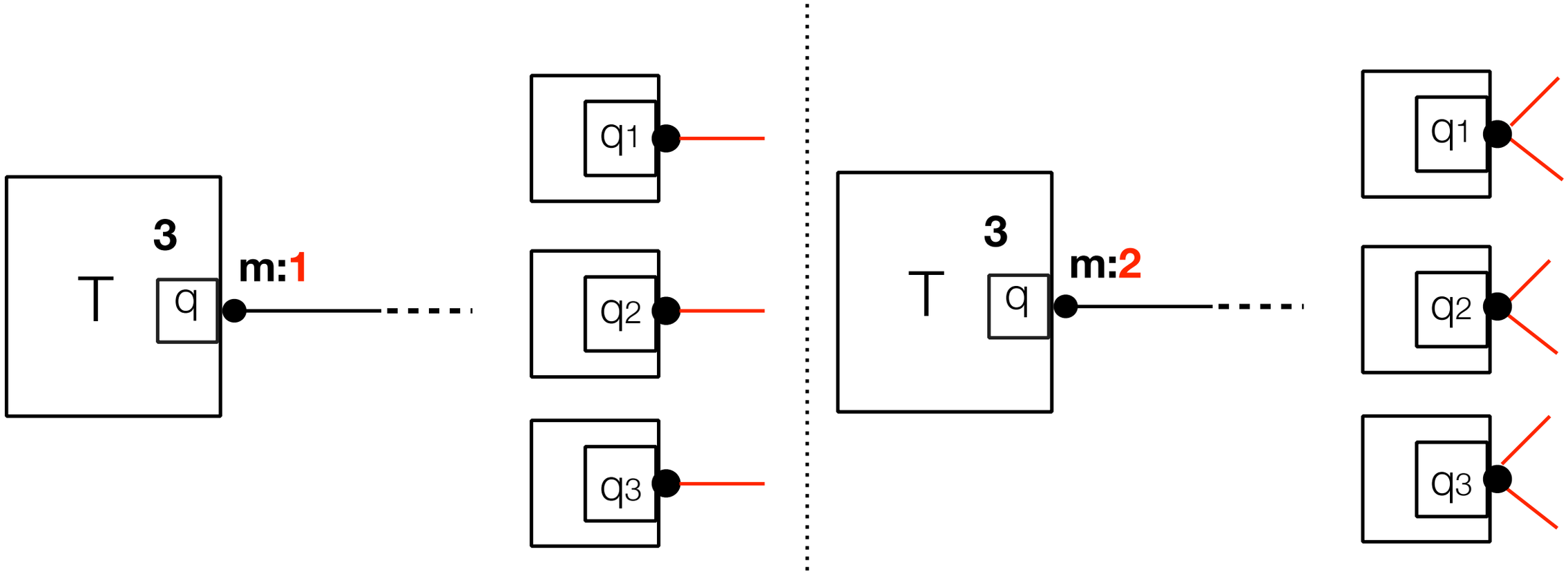
Architecture diagrams

- An architecture diagram consists of:
  - component types
    - port types
  - cardinality
  - connector motifs
    - degree
    - multiplicity



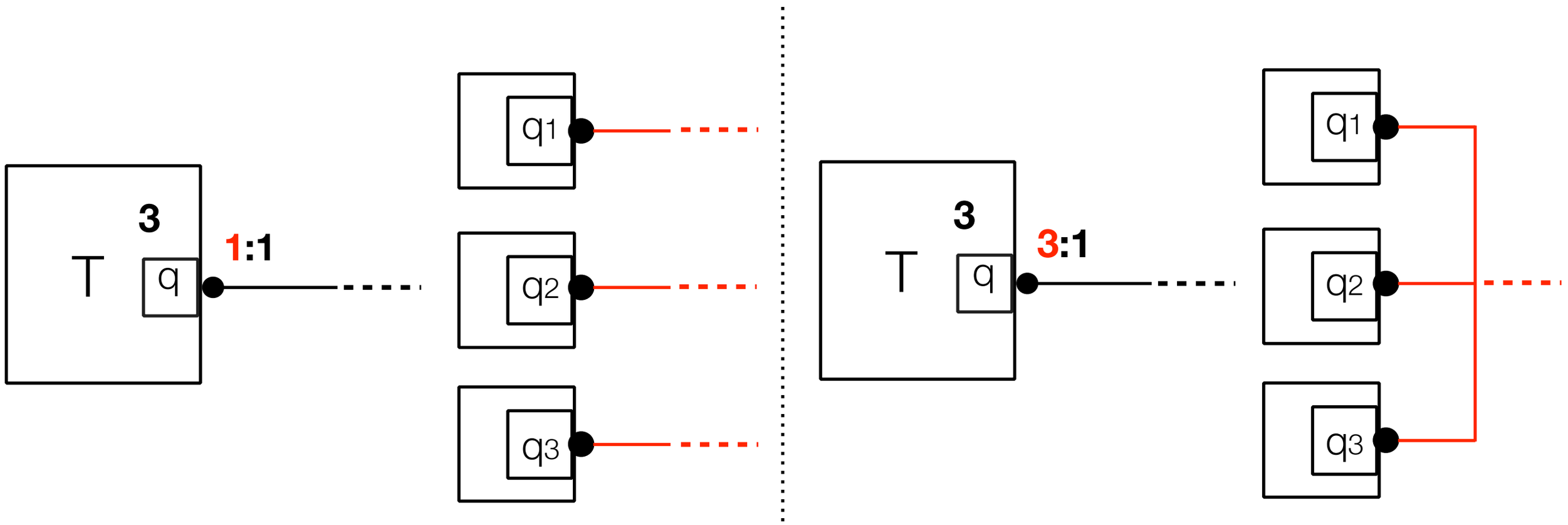
Degree constraint

Degree constraints the number of connectors attached to any instance of the port type



Multiplicity constraint

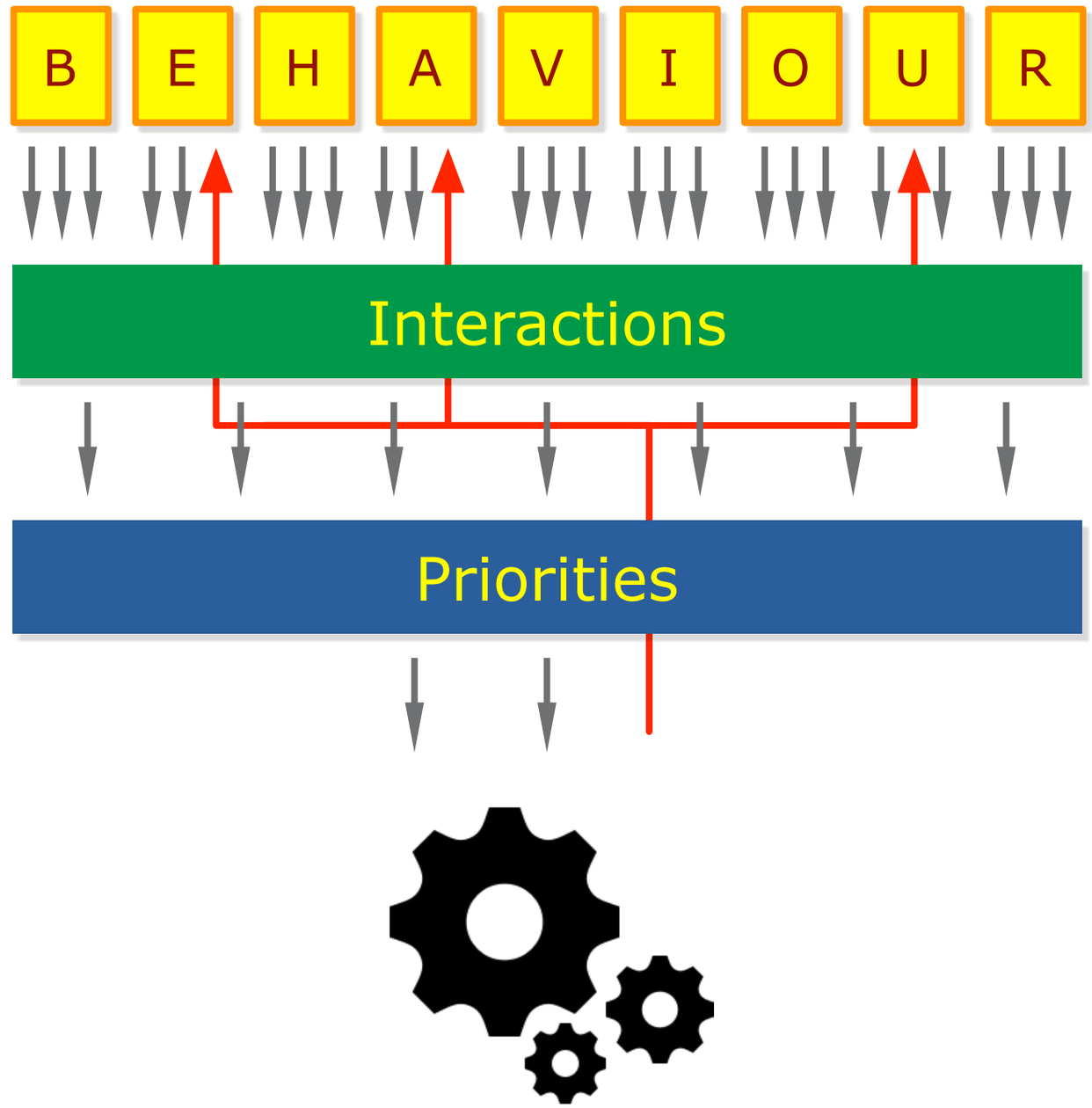
Multiplicity constraints the number of instances of the port type that must participate in a connector



Engine-based execution

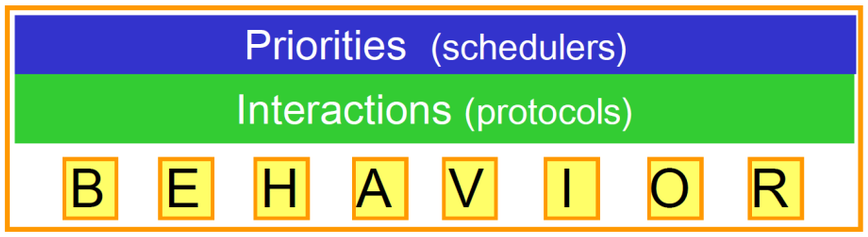
1. Components notify the Engine about enabled transitions.

2. The Engine picks an interaction and instructs the components.



BIP summary

- Compositional approach
  - allows modeling complex, hierarchical components
    - atomic components
    - interaction and priority composition operators
- Execution is orchestrated by the BIP engine
- Expressiveness
- Small number of notions
- Separation of concerns between behavior and interaction
- Architecture diagrams in BIP
  - reusability
  - allow to deal with model complexity and size



Why WebGME?

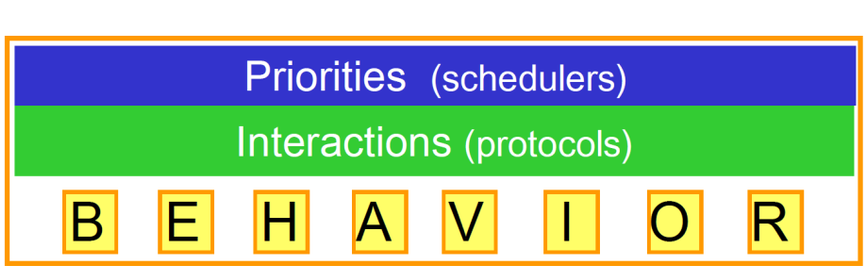
- WebGME allows:
  - Web-based
  - Collaborative
  - Versioned model editing
  - Formalized metamodeling process
    - with FORMULA

webgme.org

formula.codeplex.com

M. Maroti, T. Kecskes, R. Kereskenyi, B. Broll, P. Volgyesi, L. Juracz, T. Levendovszky, and A. Ledeczi, “Next generation (meta) modeling: Web-and cloud-based collaborative tool infrastructure.” in MPM@ MoDELS, 2014, pp. 41–60.

The design studio



BIP in WebGME

Semantic integration

Specification of the modeling language
JavaBIP metamodel in UML class diagrams

Model transformation
WebGME code generator plugins:
1. FSM to Java
2. Architecture to XML

Design guidance
WebGME plugin for encoding checking

Tool integration

Simulation and analysis
JavaBIP engine

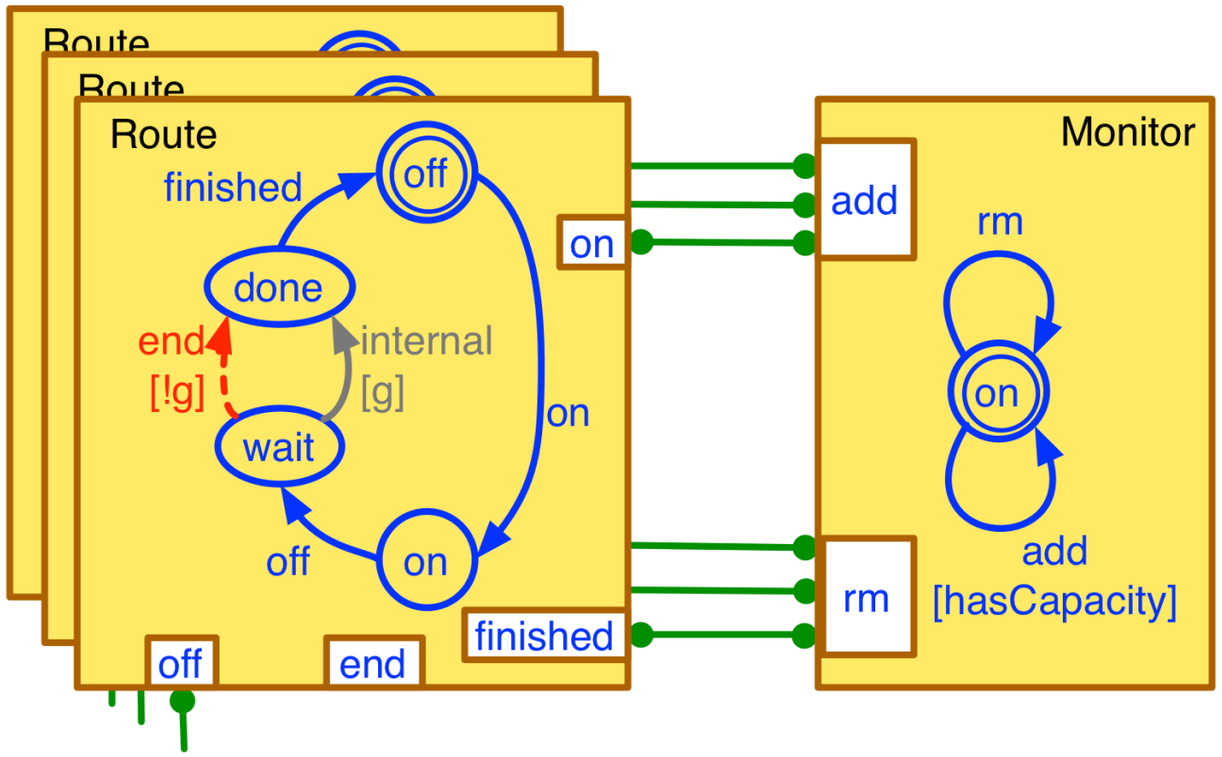
Accessibility
Web interface

Service Integration

Model repositories
1. Architecture styles
2. Component types

Simulated execution
Simulation of JavaBIP engine output

Hands-on WebGME-BIP



- Camel Routes case study
- Many independent routes share memory
  - We have to control the memory usage
  - e.g., by limiting to only a safe number of routes simultaneously

Conclusion

- The WebGME-BIP design studio
  - can be easily accessed through a web interface
  - is open-source
    - https://github.com/anmavrid/webgme-bip, https://webgme.org/
  - allows reusability of component types and parameterized models
  - allows coping with modeling complexity and size
    - component types, connector motifs
  - has formal semantics
    - allows connection with checkers and analysis tools
  - includes
    - dedicated editors for code, interaction, and behavior editing
    - code generator plugins
    - consistency checking plugin
    - integration with the JavaBIP engine and visualization of its output

Thank you for your attention