A Design Studio for Modeling, Analyzing, and Generating Systems with BIP

Anastasia Mavridou, Joseph Sifakis, and Janos Sztipanovits

Institute for Software Integrated Systems

Verimag

Motivation

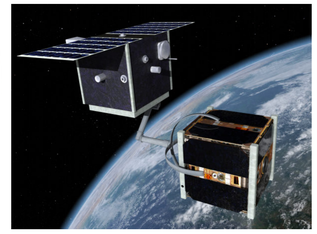
Managing system complexity requires:

- Manipulating models to raise the abstraction level
  - Expressive enough to avoid ad-hoc solutions
  - Simple enough to be acceptable by engineers
- Providing means for correctness-by-construction
- Provable equivalence between model and implementation
- Usable, easily accessed tools

BIP application examples

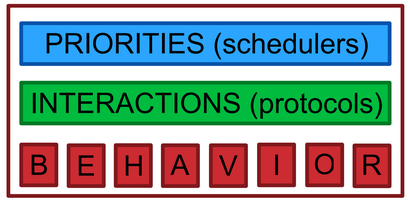
- Development of correct-by-construction satellite software
  - 49 safety properties enforced by construction
  - Compositional verification of deadlock-freedom with D-Finder
- Development of the Dala robot controller
  - > 250,000 lines of code
  - Compositional verification with D-Finder

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



Behavior-Interaction-Priority

Separation of Concerns



Strict partial order

Sets of ports

Finite State Machines

Formal Semantics

BIP component $B = (Q, P, \rightarrow)$

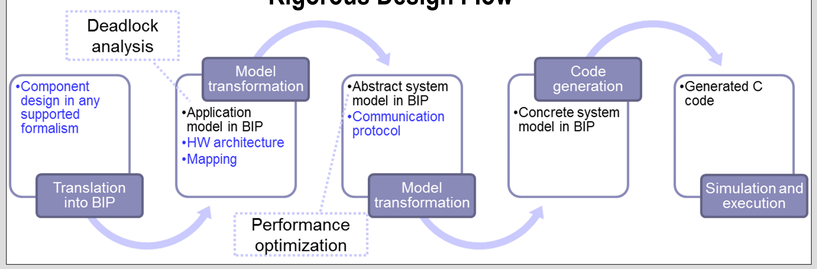
- Q set of states
- P set of communication ports
- $\rightarrow \subseteq Q \times P \times Q$ set of transitions

BIP System $\mathcal{B}_n = (Q, P, \rightarrow)$

- $Q = \prod_{i=1}^{n} Q_i$ set of states
- $\rightarrow$ least set of transitions satisfying the rule

$$\frac{a = \{p_i\}_{i \in I} \in \gamma \qquad \forall i \in I : q_i \rightarrow q_i' \qquad \forall i \notin I : q_i = q_i'}{(q_1, \ldots, q_n) \rightarrow (q_1', \ldots, q_n')}, \quad \gamma \subseteq 2^P$$

Rigorous Design Flow



Architecture-based Design

How to Model Architectures?

Mutual exclusion

$\gamma_{12} = \{\emptyset,\ b_1 b_{12},\ b_2 b_{12},\ f_1 f_{12},\ f_2 f_{12}\}$

How to Apply Architectures?

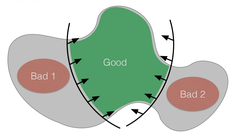
Contracts on functional components: MUX (C_1, C_2)

- Assumptions:
  - always f_1 => ($state_1 \neq$ work until b_1)
  - always f_2 => ($state_2 \neq$ work until b_2)
- Guarantee:
  - always ($state_1 \neq$ work || $state_2 \neq$ work)

How to Compose Architectures?

$$\left.\begin{array}{l} A_1(\mathcal{B}) \models \Phi_1 \\ A_2(\mathcal{B}) \models \Phi_2 \end{array}\right\} \implies (A_1 \oplus A_2)(\mathcal{B}) \models \Phi_1 \cap \Phi_2$$

Preservation of safety properties



How to Specify Architecture Styles?

Families of architectures with common characteristics:

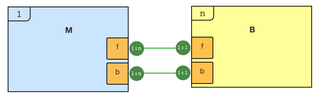
- Type of the involved components
- Properties they enforce

Correctness-by-construction

Configuration Logics

$$\begin{array}{l} \exists m : M.\ \Sigma b : B.((b.b\ m.b) + (b.f\ m.f)) \wedge \\ \forall m : M.\ \forall b : B.\ \forall b' : B\ (b \neq b'). \\ ((\overline{b.b\ b'.b}) \wedge (\overline{b.f\ b'.f}) \wedge (\overline{b.b\ b'.f}) \wedge (\overline{m.b\ m.f})) \end{array}$$

Architecture Diagrams



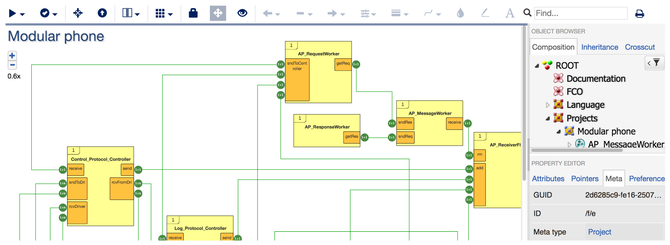
BIP Design Studio Services

- Web-based
- Collaborative environment
- Versioned model editing

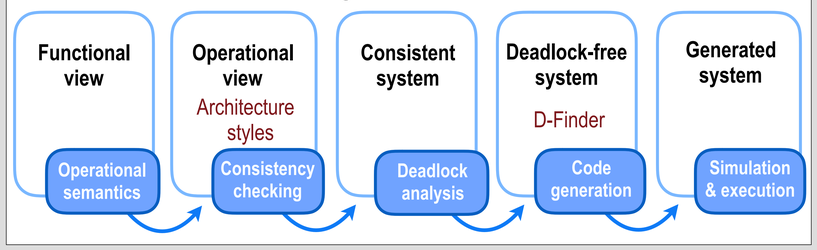
webgme.org

Generic Modeling Environment

- Multiple views
  - Functional view
  - Operational view
- Consistency checking
- Code generation
  - Finite State Machines to Java code
  - Connectors to boolean formulas in XML
- Tool Integration
  - JavaBIP engine
  - Output simulation
- Model repositories
  - Component types
  - Architecture styles



BIP Design Studio Workflow



BIP Design Studio Summary

- Promotes rigorous system design
  - Validate first, then generate the code
  - A sequence of semantic-preserving transformations
- Promotes architecture-based design
  - Allows coping with system complexity and size
  - Provides means for correctness-by-construction

All-in-one, web-based, open-source solution for building and analyzing systems with BIP

References

[1] Ananda Basu, Saddek Bensalem, Marius Bozga, Jacques Combaz, Mohamad Jaber, Thanh-Hung Nguyen, and Joseph Sifakis. "Rigorous component-based system design using the BIP framework." IEEE software 28, no. 3 (2011): 41-48.

[2] Miklos Maroti, Tamas Kecskes, Robert Kereskenyi, Brian Broll, Peter Volgyesi, Laszlo Juracz, Tihamer Levendovszky, and Akos Ledeczi. "Next Generation (Meta) Modeling: Web-and Cloud-based Collaborative Tool Infrastructure." In MPM@ MoDELS, pp. 41-60, (2014)

[3] Paul Attie, Eduard Baranov, Simon Bliudze, Mohamad Jaber, and Joseph Sifakis. "A general framework for architecture composability." In International Conference on Software Engineering and Formal Methods, pp. 128-143. Springer International Publishing, (2014).

[4] Anastasia Mavridou, Emmanouela Stachtiari, Simon Bliudze, Ivanov, Anton, Panagiotis Katsaros, and Joseph Sifakis. "Architecture-Based Design: A Satellite On-Board Software Case Study." in Proceedings of 13th International Conference on Formal Aspects of Component Software (2016): 260.