Component-based Construction of Heterogeneous Real-time Systems in BIP
(“MoCC”s and related issues in BIP)

Joseph Sifakis
VERIMAG

In collaboration with: A. Basu, M. Bozga, G. Goessler

MoCC - Models of Computation and Communication
Zurich, November 16-17, 2006
Key-issues: Component-based construction

Develop a rigorous and general basis for real-time system design and implementation:

• Concept of component and associated composition operators for incremental description and correctness by construction

• Concept for real-time architecture encompassing heterogeneity, paradigms and styles of computation e.g.
  ▪ Synchronous vs. asynchronous execution
  ▪ Event driven vs. data driven computation
  ▪ Distributed vs. centralized execution

• Automated support for component integration and generation of glue code meeting given requirements
Key-issues: Component-based construction
Existing approaches

- Theory such as process algebras and automata
- SW Component frameworks, such as
  - Coordination languages extensions of programming languages: Linda, Javaspaces, TSpaces, Concurrent Fortran, NesC
  - Middleware e.g. Corba, Javabeans, .NET
  - Software development environments: PCTE, SWbus, Softbench, Eclipse
- System modeling languages: SystemC, Statecharts, UML, Simulink/Stateflow, Metropolis, Ptolemy

Lack of
- frameworks treating interactions and system architecture as first class entities that can be composed and analyzed (usually, interaction by method call)
- rigorous models for behavior and in particular aspects related to time and resources.
## Key issues: Heterogeneity [Henzinger & Sifakis, FM06]

<table>
<thead>
<tr>
<th>Heterogeneity of interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Atomic or non atomic</td>
</tr>
<tr>
<td>• Rendezvous or Broadcast</td>
</tr>
<tr>
<td>• Binary or n-ary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Heterogeneity of execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Synchronous execution</td>
</tr>
<tr>
<td>• Asynchronous execution</td>
</tr>
<tr>
<td>• Combinations of them</td>
</tr>
</tbody>
</table>

| Heterogeneity of abstraction e.g. granularity of execution |

*We need a framework *directly* encompassing heterogeneity*
Key issues: Heterogeneity - Example

A: Atomic interaction  R: Rendezvous  B: Broadcast

Asynchronous Computation

Synchronous Computation

Matlab/Simulink
VHDL
Synchronous languages
Overview

• About component-based construction
  • Interaction modeling
  • Priority modeling
  • Implementation
  • Modeling systems in BIP
  • Discussion
Build a component $C$ satisfying a given property $P$, from

- $\mathcal{C}_0$ a set of *atomic* components modeling behavior
- $\mathcal{G} = \{g_{l_1}, \ldots, g_{l_i}, \ldots\}$ a set of glue operators on components

Glue operators

- model mechanisms used for communication and control such as protocols, controllers, buses.
- restrict the behavior of their arguments, that is

$$gl(C_1, C_2, \ldots, C_n) | A_1 \text{ refines } C_1$$
Component-based construction – Formal framework

Semantics:
• Atomic components $\rightarrow$ behavior
• Glue operators transform sets of components into components

The process algebra paradigm
• Components are terms of an algebra of terms $(\mathcal{C}, \equiv)$ generated from $\mathcal{C}_0$ by using operators from $\mathcal{G}$
• $\equiv$ is a congruence compatible with semantics
Find sets of glue operators meeting the following requirements:

1. Incremental description
2. Correctness-by-construction
3. Expressiveness (discussed later)
Component-based construction – Incremental description

1. Decomposition

\[ g/l \approx C_1 \oplus C_2 \oplus \ldots \oplus C_n \]

2. Flattening

\[ g/l_1 \oplus g/l_2 \approx C_1 \oplus C_2 \oplus \ldots \oplus C_n \]

Flattening can be achieved by using a (partial) associative operation \( \oplus \) on GL.
Building correct systems from correct components

We need compositionality results about preservation of progress properties such as deadlock-freedom and liveness.
Integrated components preserve essential properties.

Composability means non interference of properties of integrated components. Lack of results for guaranteeing property stability e.g.

- non composability of scheduling algorithms
- feature interaction
Component-based construction – The BIP framework

Layered component model

Composition (incremental description)
Overview

• About component-based construction

• Interaction modeling

• Priority modeling

• Implementation

• Modeling systems in BIP

• Discussion
Interaction modeling

- A *connector* is a set of ports which can be involved in an interaction.

- Port attributes (*complete.*, *incomplete.*) are used to distinguish between rendezvous and broadcast.

- An *interaction* of a connector is a set of ports such that: either it contains some complete port or it is maximal.

```
Interactions:
{tick1,tick2,tick3} {out1} {out1,in2} {out1,in3} {out1,in2, in3}
```
Interaction modeling - Examples

cl1 -> cl2
CN: {cl1, cl2}
CP: ∅

out -> in
CN: {out, in}
CP: {out}

out -> in1 -> in2
CN: {in1, out, in2}
CP: {out}
Interaction modeling – Operational semantics

CN: {put, get} {prod} {cons}
CP: {prod} {cons}
Interaction modeling – Incremental Composition

CN[\(P,C\)]: \{put, get\}
CP[\(P,C\)]: \emptyset

CN[\(P\)]: \{put\}, \{prod\}
CP[\(P\)]: \{prod\}

CN[\(C\)]: \{get\}, \{cons\}
CP[\(C\)]: \{cons\}

CN: \{put, get\}, \{prod\}, \{cons\}
CP: \{prod\}, \{cons\}
Overview

• About component-based construction
• Interaction modeling
• Priority modeling
• Implementation
• Modeling systems in BIP
• Discussion
Priorities

Priorities are a powerful tool for restricting non-determinism:

• they allow straightforward modeling of urgency and scheduling policies for real-time systems
• run to completion and synchronous execution can be modeled by assigning priorities to threads
• they can advantageously replace (static) restriction of process algebras
Priorities – Priorities as controllers

A controller restricts the behavior (non determinism) of system \( S \) to enforce a property \( P \)

\[
\text{Controller for property } P
\]

\[
\text{Interactions}
\]

\[
\text{system } S
\]

**Results** [Goessler & Sifakis, FMCO2003]:

- Restrictions induced by controllers enforcing deadlock-free state invariants can be described by dynamic priorities

- Conversely, for any restriction induced by dynamic priorities there exists a controller enforcing a deadlock-free state invariant
**Priorities - Definition**

### Priority rules

<table>
<thead>
<tr>
<th>Priority rule</th>
<th>Restricted guard $g_1'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{true} \rightarrow p_1 \leftarrow p_2$</td>
<td>$g_1' = g_1 \cup \emptyset \cup g_2$</td>
</tr>
<tr>
<td>$C \rightarrow p_1 \leftarrow p_2$</td>
<td>$g_1' = g_1 \cup \emptyset (C \cup g_2)$</td>
</tr>
</tbody>
</table>
Priorities – Example: Mutual exclusion + FIFO policy

<table>
<thead>
<tr>
<th>Condition</th>
<th>Event</th>
<th>Priority 1</th>
<th>Priority 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1 \leq t_2$</td>
<td>$b_1 \xrightarrow{f_2}$</td>
<td>$b_1$</td>
<td>$b_2$</td>
</tr>
<tr>
<td>$t_2 &lt; t_1$</td>
<td>$b_2 \xrightarrow{f_1}$</td>
<td>$b_2$</td>
<td>$b_1$</td>
</tr>
<tr>
<td>True</td>
<td>$b_1 \xrightarrow{f_2}$</td>
<td>$true$</td>
<td>$b_2$</td>
</tr>
<tr>
<td>True</td>
<td>$b_2 \xrightarrow{f_1}$</td>
<td>$true$</td>
<td>$b_1$</td>
</tr>
</tbody>
</table>

Diagram:

- **sleep1**: $a_1$, start $t_1$
- **wait1**: $b_1$, $f_1$
- **use1**: $f_1$

- **sleep2**: $a_2$, start $t_2$
- **wait2**: $b_2$
- **use2**: $f_2$
Overview

• About component-based construction

• Interaction modeling

• Priority modeling

• Implementation

• Modeling systems in BIP

• Discussion
component C
port complete: p1, ... ; incomplete: p2, ...
data {# int x, float y, bool z, .... #}
init {# z=false; #}
behavior
state s1
  on p1 provided g1 do f1 to s1'
  ..................  ......  
on pn provided gn do fn to sn'

  state s2
  on .....  
  ....

  state sn
  on .....  
end
end
Implementation – the BIP language: connectors and priorities

**connector** BUS= \{p, p', ... , \}

**complete()**

**behavior**

```
on a1 provided g_{a1} do f_{a1}

...........
on an provided g_{an} do f_{an}
```

**end**

**priority** PR

```
if C1 (a1 < a2), (a3 < a4) , ...
if C2 (a < ...), (a < ...) , ...
...
if Cn (a <...), (a <...) , ...
```
Implementation – the BIP language: compound component

cOMPONENT name
    contains c_name1 i_name1(par_list)
        ……
    contains c_namen i_namen(par_list)

CONNECTOR name1
    ……
CONNECTOR namem

priority name1
    ……
priority namek
end
Implementation – the BIP toolset

Graphic language
AADL or UML

BIP language

C++

BIP Platform

THINK

IF

IF Platform
Implementation – C++ code generation for the BIP platform

Component Meta-model

Interaction Meta-model

Priority Meta-model

Engine

BIP Platform
Implementation – The BIP platform

- Code execution and state space exploration features
- Implementation in C++ on Linux using POSIX threads
  - Thread assignments preserve semantics
Implementation – The BIP platform: The engine

- **init**
  - Launch atom’s threads

- **loop**
  - Wait all atoms
  - Compute legal interactions

- **execute**
  - Notify involved atoms
  - Execute chosen interaction transfer

- **choose**
  - Choose among maximal

- **filter**
  - Filter w.r.t. priorities

- **stable**
  - Filter w.r.t. priorities
Overview

• About component-based construction
• Interaction modeling
• Priority modeling
• Implementation
• Modeling systems in BIP
• Discussion
Modeling in BIP—Other approaches encompassing heterogeneity

Vanderbilt’s Approach

- Semantic Unit
  - Meta-model
    - Composition
      - Operators
    - Behavior

- Operational Semantics

ASML .net

Metropolis

- Semantic Domain
  - Quantity Manager
  - Media
  - Behavior

- Operational Semantics

Platform

PTOLEMY

- MoC
  - (Model of Computation)
    - Director
    - Channels
    - Behavior

- Operational Semantics

Platform
Modeling in BIP– System construction space

A system is defined as a point of the 3-dimensional space. Full separation of concerns: any combination of coordinates defines a system.
Modeling in BIP – System construction space (2)

Model construction space for PTOLEMY
Study transformations characterizing relations between classes of systems:
- Untimed – timed
- Synchronous – asynchronous
- Event triggered – data triggered
Modeling in BIP – Property preservation

- Property preservation
- +refinement
- +interaction
- System
- Architecture
- Deadlock-free
- State Invariant
Modeling in BIP – Timed systems

Timed Component

PR: red_guards → tick \ all_other_ports

Timed architecture
Modeling in BIP – Synchronous systems

Synchronous component

PR: syn\langle all\_other\_ports

Synchronous architecture
Overview

• About component-based construction
• Interaction modeling
• Priority modeling
• Implementation
• Modeling systems in BIP
• Discussion
**Discussion – Semantic frameworks**

**Denotational semantics:**
elegant and powerful but we absolutely need associated executable semantic models to be able to faithfully apply theory in methods and tools

**Operational semantics:**
inherent difficulties to deal with concurrency and resource modeling

**For both:**
We need « high level » semantic frameworks where structure is a first class entity.
Discussion – Structural Expressiveness

Find a notion of expressiveness different from existing ones which completely ignore structure e.g. all finite state formalisms are equally expressive

For given $B$, $IM$ and $PR$ which coordination problems can be solved (without modifying behavior of atomic components)?
• Study Component Algebras \( \text{CA}= (B, \text{GL}, \oplus, \equiv) \), where
  • \((\text{GL}, \oplus)\) is a commutative monoid
  • \(\equiv\) is a congruence compatible with operational semantics

• Given two component algebras defined on the same set of atomic components,

  **CA1 is more expressive** than CA2 if \( \forall P \\forall B_1, .., B_n \\exists \text{gl}_2 \in \text{GL}_2. \text{gl}_2(B_1, .., B_n) \text{ sat } P \Rightarrow \exists \text{gl}_1 \in \text{GL}_1. \text{gl}_1(B_1, ... B_n) \text{ sat } P \)
Discussion – Summary for BIP

Framework for component-based construction encompassing heterogeneity and relying on a minimal set of constructs and principles

Clear separation between structure (interaction +priority) and behavior

• Structure is a first class entity
• Layered description => separation of concerns => incrementality
• Correctness-by-construction techniques for deadlock-freedom and liveness, based (mainly) on sufficient conditions on the structure
Discussion - Work directions for BIP

Theory
- An algebraic framework based on structural expressiveness
- Correctness by construction
- Model transformation techniques – relating classes of systems

Methodology
- Using BIP as a programming model
- Modeling architectures in BIP

BIP toolset Implementation
- Generation of BIP models from system description languages such as SysML (IST/SPEEDS project), AADL and SystemC (ITEA/Spices project)
- Code generation and optimization for various platforms
- Validation techniques
More about BIP:

• http://www-verimag.imag.fr/index.php?page=tools

• Email to Joseph.Sifakis@imag.fr

THANK YOU