Component-based Construction of Heterogeneous Real-time Systems in BIP

Joseph Sifakis
VERIMAG
FETCH07
Villard-de-Lans, January 2007
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]

Heterogeneity of interaction
- Atomic or non-atomic
- Rendezvous or Broadcast
- Binary or n-ary

Heterogeneity of execution
- Synchronous execution
- Asynchronous execution
- Combinations of them

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

Lotos CSP

Java UML

SDL UML

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
- $\mathcal{GL} = \{gl_1, \ldots, gl_i, \ldots\}$ 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 → behavior
• Glue operators transform sets of components into components

The process algebra paradigm
• Components are terms of an algebra of terms \((\mathcal{C}, \cong)\) generated from \(\mathcal{C}_0\) by using operators from \(\mathcal{G}\)
• \(\cong\) 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

\[ C_1, C_2, \ldots, C_n \]  
\[ \cong \]

\[ C_1, C_2, \ldots, C_n \]

2. Flattening

Flattening can be achieved by using a (partial) associative operation \( \oplus \) on GL

Joseph Sifakis FETCH07
Component-based construction - Correctness by construction: Compositionality

Building correct systems from correct components

\[
\begin{align*}
\text{We need compositionality results about preservation of progress properties such as deadlock-freedom and liveness.}
\end{align*}
\]
Component-based construction - Correctness by construction: Composability

Integrated components preserve essential properties

\[
\begin{align*}
\text{sat } & P \\
\text{sat } & P' \\
\text{implies } & P \land P'
\end{align*}
\]

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

Priorities (Conflict resolution)

Interaction Model (Collaboration)

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

1. \(CN: \{\text{cl1, cl2}\}\)
   \(CP: \emptyset\)

2. \(CN: \{\text{out, in}\}\)
   \(CP: \{\text{out}\}\)

3. \(CN: \{\text{in1, out, in2}\}\)
   \(CP: \{\text{out}\}\)
Interaction modeling – Operational semantics

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

Operational Semantics
Interaction modeling – Incremental Composition

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

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

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

CN: \{\text{put, get}, \{\text{prod}\}, \{\text{cons}\}\}
CP: \{\text{prod}, \{\text{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
A controller restricts the behavior (non determinism) of system S to enforce a property P

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

<table>
<thead>
<tr>
<th>Priority rule</th>
<th>Restricted guard $g_1'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$true \rightarrow p_1 \preceq p_2$</td>
<td>$g_1' = g_1 \land \neg g_2$</td>
</tr>
<tr>
<td>$C \rightarrow p_1 \preceq p_2$</td>
<td>$g_1' = g_1 \land \neg (C \land g_2)$</td>
</tr>
</tbody>
</table>
Priorities – Example: Mutual exclusion + FIFO policy

<table>
<thead>
<tr>
<th></th>
<th>t1 ≤ t2 → b1 ( b2</th>
<th>t2 &lt; t1 → b2 ( b1</th>
</tr>
</thead>
<tbody>
<tr>
<td>true → b1 ( f2</td>
<td>true → b2 ( f1</td>
<td></td>
</tr>
</tbody>
</table>

- \( sleep1 \)
  - a1
  - start t1
- \( wait1 \)
  - b1
- \( use1 \)
  - f1

- \( sleep2 \)
  - a2
  - start t2
- \( wait2 \)
  - b2
- \( use2 \)
  - f2
Overview

• About component-based construction
• Interaction modeling
• Priority modeling
• Implementation
• Modeling systems in BIP
• Discussion
Implementation – the BIP language: atomic component

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
.connector BUS = \{p, p', \ldots, \}
complete()
behavior
  on \alpha_1 \text{ provided } g_{\alpha_1} \text{ do } f_{\alpha_1}
  \ldots \ldots
  on \alpha_n \text{ provided } g_{\alpha_n} \text{ do } f_{\alpha_n}
end

.priority PR
  if C_1 (\alpha_1 < \alpha_2), (\alpha_3 < \alpha_4), \ldots
  if C_2 (\alpha < \ldots), (\alpha < \ldots), \ldots
  \ldots
  if C_n (\alpha < \ldots), (\alpha < \ldots), \ldots
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++

IF Platform

BIP Platform

THINK

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

Component Meta-model

Interaction Meta-model

Priority Meta-model

Engine

BIP Platform

BIP model

C→a(b)
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
  - Filter w.r.t. priorities
  - Compute legal interactions

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

- **choose**
  - Choose among maximal

- **filter**
  - Choose among maximal

Joseph Sifakis FETCH07
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

Joseph Sifakis FETCH07
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

Interaction (channels)

Model of Computation

Behavior
The BIP framework – Relating classes of components

Study transformations characterizing relations between classes of systems:
- Untimed – timed
- Synchronous – asynchronous
- Event triggered – data triggered
Modeling in BIP – Timed systems

Timed Component

PR: red_guards → tick (all_other_ports)

Timed architecture
Modeling in BIP – MPEG4 Video encoder: Componentization

Transform a monolithic program into a componentized one
  ++ reconfigurability, schedulability
  -- overheads (memory, execution time)

Video encoder characteristics:
  • 12000 lines of C code
  • Encodes one frame at a time:
    - grabPicture() : gets a frame
    - outputPicture() : produces an encoded frame
Modeling in BIP – Video encoder: The Encode component

GrabMacroBlock: splits a frame in \((W\times H)/256\) macro blocks, outputs one at a time

Reconstruction: regenerates the encoded frame from the encoded macro blocks.

connections

Joseph Sifakis FETCH07
Modeling in BIP – Video encoder: Atomic components

**GrabMacroBlock**

- **f_in**: \( c = \text{MAX} \), \( c := 0 \)
- **out**: \( c < \text{MAX} \), \( \text{grabMacroBlock}(), c := c + 1 \)

**Reconstruction**

- **in**: \( c < \text{MAX} \), \( c := c + 1 \)
- **f_out**: \( c = \text{MAX} \), \( c := 0 \)
- **reconstruction()**

**Generic Functional component**

- **in**: \( \text{fn}() \)
- **out**: \( \text{fn}() \)

**Notes**

- \( \text{MAX} = \frac{W \times H}{256} \)
- \( W = \text{width of frame} \)
- \( H = \text{height of frame} \)
Modeling in BIP – Video encoder: The BIP Encoder features

- BIP code describes a control skeleton for the encoder
  - Consists of 20 atomic components and 34 connectors
  - ~ 500 lines of BIP code
  - Functional components call routines from the encoder library

- The generated C++ code from BIP is ~ 2,000 lines

- The size of the BIP binary is 288 Kb compared to 172 Kb of monolithic binary.
Overhead in execution time wrt monolithic code:

• ~66% due to communication (can be reduced by composing components at compile time)
  – function calls by atomic components to the execution engine for synchronization.

• ~34% due to resolution of non determinism (can be reduced by narrowing the search space at compile time)
  – time spent by engine to evaluate feasible interactions

Problem: Reduce execution time overhead for componentized code
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 $CA = (\mathcal{B}, GL, \oplus, \simeq)$, where
  - $(GL, \oplus)$ is a commutative monoid
  - $\simeq$ is a congruence compatible with operational semantics

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

  $CA_1$ is more expressive than $CA_2$ if $\forall P \ \forall B_1, \ldots, B_n \ \exists gl_2 \in GL_2. \ gl_2(B_1, \ldots, B_n) \ sat \ P \ \Rightarrow \ \exists gl_1 \in GL_1. \ gl_1(B_1, \ldots, B_n) \ sat \ P$
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