Generating Well-Synchronized Multithreaded Programs from Software Architecture Descriptions

Marco Bernardo and Edoardo Bontà

Università di Urbino “Carlo Bo” - Italy
Istituto di Scienze e Tecnologie dell’Informazione
Overview

• Multithreading provides an adequate support for concurrent programming, but requires the software developer to take care of the correct synchronization and exchange of data among threads.

• The software developer should be provided with a suitable support for being confident in the correctness of the way in which the thread synchronization is dealt with.

• A complicated combination of several different synchronization techniques – sleep and wakeup primitives, semaphores, monitors, etc. – may be necessary.
An Architecture-driven Approach

- Developed a [Java] package of software components related to the main architectural concepts, which take care of the details of the thread synchronization in a way that is transparent to the software developer.

1. Provide an architectural description [PADL] of the multithreaded program, which specifies at a high level of abstraction the program topology in terms of thread instances and their interactions.

2. Use a translator [PADL2Java] that, from the PADL specification of the multithreaded Java program, automatically generates (a skeleton of) the program based on the Java package, in a way that guarantees the correct thread synchronization.

No global variables.
No passing object references while keeping a copy of them.
Example: Real-Time Audio Processing System
Example: Real-Time Audio Processing System

Screenshot of a real application
Java package Sync: Layer Architecture

public abstract class Architecture
extends ThreadElem {...}

Advantages deriving from inheritance:
• Hierarchy
• Composition
• Concurrency
Java package Sync: Layer ThreadElem

public abstract class ThreadElem
extends Thread {...}

- Constructors of the Thread class redefined in the ThreadElem class with the same parameters.
- Additional methods for easily generating Java code for the thread.
Java package **Sync:**

**Layer Port**

**Architecture**

```
<table>
<thead>
<tr>
<th>ThreadElem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
</tr>
<tr>
<td>Connector</td>
</tr>
</tbody>
</table>
```

**Synchronous / Asynchronous Ports:**

- uni - sender
- and - sender
- or - sender
- uni - receiver
- and - receiver
- or - receiver
Java package **Sync**: Layer Connector

Connector types:
- asynchronous-to-asynchronous
- asynchronous-to-synchronous
- synchronous-to-asynchronous
- synchronous-to-synchronous
## Structure of a PADL

**Textual Description**

<table>
<thead>
<tr>
<th>ARCHI_TYPE</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCHI_ELEM_TYPES</td>
<td>Architectural element types: behaviors and interactions</td>
</tr>
<tr>
<td>ARCHI_TOPOLOGY</td>
<td></td>
</tr>
<tr>
<td>ARCHI_ELEM_INSTANCES</td>
<td>Architectural element instances</td>
</tr>
<tr>
<td>ARCHI_INTERACTIONS</td>
<td>Architectural interactions</td>
</tr>
<tr>
<td>ARCHI_ATTACHMENTs</td>
<td>Architectural attachments</td>
</tr>
</tbody>
</table>
Generating Code

PADL2Java [-p/-c/-a] file1.padl  file2.java

import Sync;

class <Element Type Name> extends ThreadElem {
    Sections:
    • DEFINING CONSTRUCTOR
    • DEFINING BEHAVIOR (skeleton)
    • INSTANTIATING INPUT INTERACTIONS
    • INSTANTIATING OUTPUT INTERACTIONS
}

class <Architecture Type Name> extends Architecture {
    Sections:
    • DECLARING THREADS (element types)
    • DECLARING ARCHITECTURAL INTERACTIONS
    • DEFINING CONSTRUCTOR
    • BUILDING ARCHITECTURE
    • RUNNING ARCHITECTURE
}
Building Architecture:
Architectural Element Instances

- Console
- Input Audio Device Driver
- Sound Processor
- Output Audio Device Driver
Building Architecture: Architectural Interactions

- Console
- Input Audio Device Driver
- Sound Processor
- Output Audio Device Driver
Building Architecture: Architectural Attachments
Future Work

• Automatic generation of the thread behavior.

• Preserving the properties proved at the architectural level by means of the analysis techniques for PADL.

• Integrating the translator in the software tool TwoTowers.