

## 14. ArchJava – A Lightweight Java Extension for Architecture – Provided and Required Ports and Services

Prof. Dr. Uwe Aßmann  
Technische Universität Dresden  
Institut für Software- und Multimediatechnik  
<http://st.inf.tu-dresden.de>  
Version 13-1.0, May 14, 2013



CBSE, © Prof. Uwe Aßmann

1

## Literature (To Be Read)

- ▶ J. Aldrich, G. Chambers, D. Notkin. Architectural Reasoning in ArchJava. European Conference on Object-Oriented Programming (ECOOP) 2002, LNCS
- ▶ <http://www.archjava.org>



Prof. U. Aßmann, CBSE

2

## The Problem

- ▶ An architectural description language needs many constructs that are already available in a standard language
  - Control-flow constructs
  - Iteration, Recursion
  - Data types
- ▶ Reasoning is simpler if components and architecture are described in the same language (same analysis tools)



Prof. U. Aßmann, CBSE

3

## Communication Integrity

Provided and required interfaces enable **explicit dependencies between components** and **communication integrity**:

Communication integrity:  
Every implementation component can only communicate with the neighbors that were specified in the interface or the architecture (connection topology)

Communication integrity relies on provided and required interfaces.



Prof. U. Aßmann, CBSE

4



## Ports in ArchJava

- ▶ In ArchJava, ports are *call services (call ports)*
  - Required, provided, broadcast ports

```
public component class Parser {
  public port in {
    provides void setInfo(Token symbol, SymTabEntry e);
    requires Token nextToken() throws ScanException;
  }
  public port out {
    provides SymTabEntry getInfo(Token t);
    requires void compile(AST ast);
  }
  public void parse() {
    Token tok = in.nextToken();
    AST ast = parseFile(tok);
    out.compile(ast);
  }
  AST parseFile(Token lookahead) { ... }
  void setInfo(Token t, SymTabEntry e) {...}
  SymTabEntry getInfo(Token t) { ... }
  ...
}
```



5



## Connections and Subcomponents

- ▶ Connections between ports are specified with **connect** keyword (as in Unicon)
- ▶ Broadcast ports are similar to required ports, but can be connected to many recipients
- ▶ Nested component hierarchies are possible with nested subcomponents (*final* means *atomic*)

```
public component class Compiler {
  private Scanner scanner = ...;
  private final Parser parser = ...;
  private final CodeGen codegen = ...;

  connect scanner.out, parser.in;
  connect parser.out, codegen.in;

  public static void main(String args[]) {
    new Compiler().compile(args);
  }
  public void compile(String args[]) {
    // for each file in args do:
    ...parser.parse();...
  }
}
```

Nested subcomponents  
also atomic ones



6



## Asynchronous Connections

- ▶ By default, only call connections (and broadcasting call connections) are supplied
- ▶ Asynchronous connections must be defined by the user
- ▶ No connectors are provided
  - ▶ But can be programmed as specific communicating components



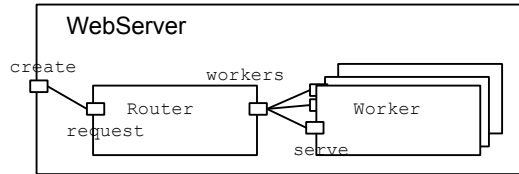
## Architecture Modelling

- ▶ All object-oriented concepts can be used to model architectures
  - Inheritance for sharing features of architectures
  - Abstract and generic classes for architectural frameworks
  - Framework hook technology for frameworks
  - Dynamic architectures with polymorphism, constructors, and abstract factories





## Run-Time, Dynamic Architectures



```
public component class WebServer {
  private final Router r = new Router();
  // initial configuration
  connect r.request, create;
  // A connection pattern allows for dynamic calls to connect function
  connect pattern Router.workers, Worker.serve;

  public void run() { r.listen(); }
  private port create {
    provides r.workers requestWorker() {
      final Worker newWorker = new Worker();
      // dynamic connection of new workers with port r.workers
      r.workers.connection = connect(r.workers, newWorker.serve);
      // connect expressions return connection objects
      return connection;
    }
  }
}
```

Port interface type

9



## Experience with ArchJava

- ▶ Taprats is a pattern-designing program for islamic tile patterns
  - 12.5 KLOC Java
- ▶ Was reengineered with
  - 5.5 hours, 30 minutes per KLOC
  - Since ArchJava enforces communication integrity, code had to be reengineered, dependencies must be cut
  - Violations of the Law of Demeter create problems
  - Law of Demeter: "Don't call grandneighbors, only neighbors"



Prof. U. Aßmann, CBSE

10



## What Have We Learned?

- ▶ ArchJava is a ready-to-use architectural extension of Java
- ▶ Inherits benefits from object-orientation and architectural languages
- ▶ Violations of the Law of Demeter create problems
- ▶ ArchJava components, ports, and connections can easily be integrated into other modular and object-oriented languages



Prof. U. Aßmann, CBSE

11



## The End



Prof. U. Aßmann, CBSE

12