ArchJava

Connecting Software Architecture to Implementation

Jonathan Aldrich
Craig Chambers
David Notkin

University of Washington

Software Architecture

- Software Architecture:

  the organization of software systems as a collection of components, connections between the components, and constraint on the interactions between components.

Why should we care?

ADLs

- Old ADLs decouple implementation code from architecture
- ArchJava's contribution:
  - Add architecture to language
    - Architecture updated as code evolves
  - Architecture is enforced by type system

A Parser Component

```
public component class Parser {

}
```

Component class
- Defines architectural object
- Must obey architectural constraints

A Parser Component

```
public component class Parser {
  public port in {
    requires Token nextToken();
  }
  public port out {
    provides AST parse();
  }

}
```

Components communicate through *Ports*

- A two-way interface
- Define *provided* and *required* methods

A Parser Component

```
public component class Parser {
  public port in {
    requires Token nextToken();
  }
  public port out {
    provides AST parse();
  }
  AST parse() {
    Token tok =in.nextToken();
    return parseExpr(tok);
  }
  AST parseExpr(Token tok) { ... }
  ...
}
```

Can fill in architecture with ordinary Java code

Hierarchical Composition


```
public component class Compiler {
  private Scanner scanner = new Scanner();
  private Parser parser = new Parser();
  private CodeGen codegen = new CodeGen();
```

Subcomponents

- Component instances inside another component
- Communicate through connected ports

Hierarchical Composition


```
public component class Compiler {
  private Scanner scanner = new Scanner();
  private Parser parser = new Parser();
  private CodeGen codegen = new CodeGen();
  connect scanner.out, parser.in;
  connect parser.out, codegen.in;
```

Connections

- Bind required methods to provided methods

Evaluating Questions

- Does ArchJava guarantee consistent architecture?
- Is ArchJava *expressive* enough for real systems?
- Can ArchJava aid *software evolution* tasks?

Evaluation Questions

- Does ArchJava guarantee consistent architecture?
  - Yes, using the type system
- Is ArchJava *expressive* enough for real systems?
- Can ArchJava aid *software evolution* tasks?

Evaluation Questions

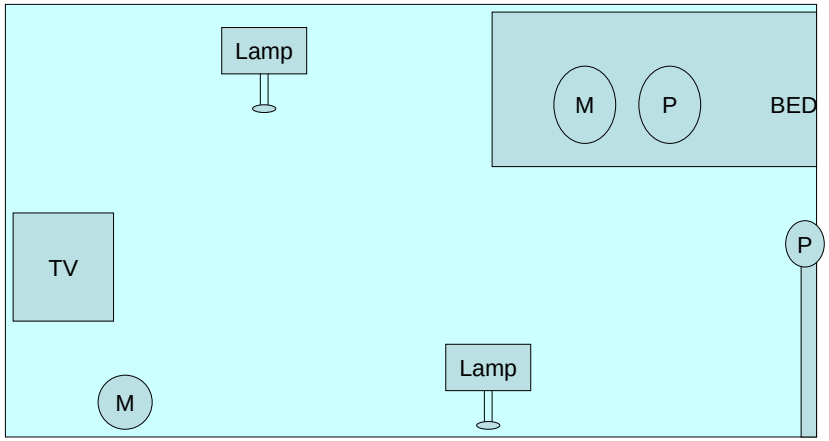
- Does ArchJava guarantee consistent architecture?
  - Yes, using the type system
- Is ArchJava *expressive* enough for real systems?
  - Yes, tested in several case studies
- Can ArchJava aid *software evolution* tasks?

Case Study
River in the Light of ArchJava

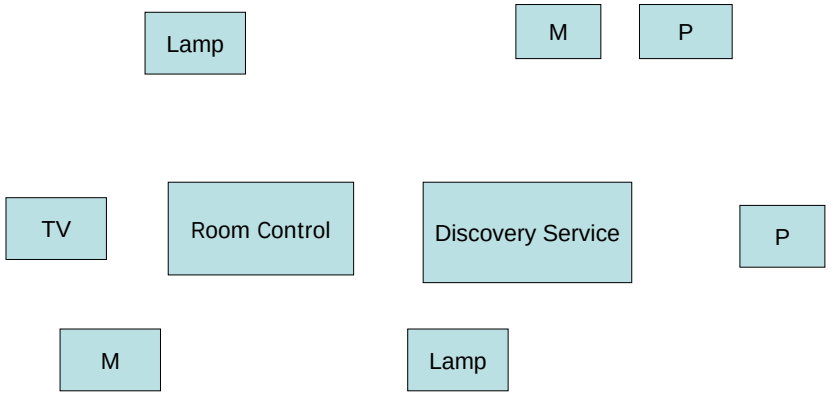
The use of user-defined connectors in ubiquitous computing systems

Aiman Erbad
Advisor : Craig Chambers
University of Washington

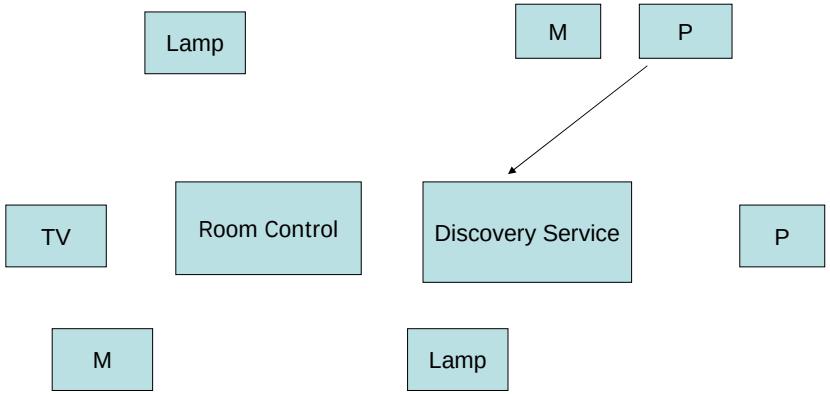
Ubiquitous computing system



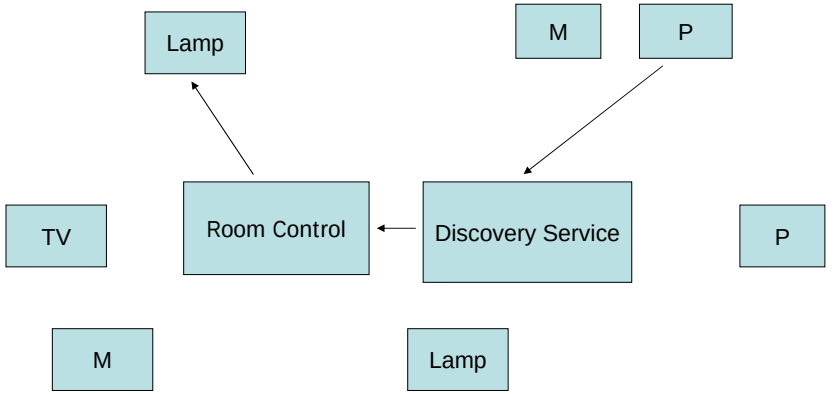
Abstract representation



Abstract representation



Abstract representation



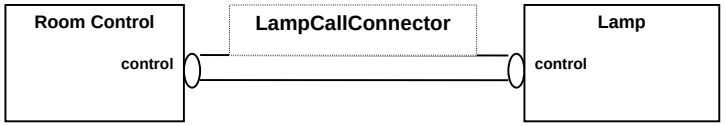
River

- River
  - Two communication primitive function calls, and events
  - Use SQL to specify the end point

```
String query = " Select id
                From location-to-id
                Where type = 'lamp'  AND
                      location = ‘room'  "
```

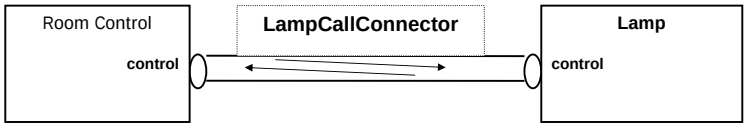
User-defined connectors

- The connector abstraction



User-defined connectors

- The connection abstraction



- User defined connectors:
  - Capture communication between services

New Design

```
public class LampCallConnector extends RiverCallConnector {
    static String query = " Select id
                            From location-to-id
                            Where type = 'lamp'  AND
                                  location = 'kitchen' “;
                  // typecheck and invoke functions are inherited from
                  //RiverCallConnector
}
```

New Design

```
public class LampCallConnector extends RiverCallConnector {
    static String query = " Select id
                            From location-to-id
                        Where type = 'lamp'  AND
                                location = 'kitchen' "
                // functions are inherited from RiverCallConnector
}

/* Inside client – Room Control */

Control con = connect(this.control, Lamp.control)
        with new LampCallConnector;
con.turnoff();
```

Evaluation Questions

- Does ArchJava guarantee consistent architecture?
  - Yes, using the type system
- Is ArchJava *expressive* enough for real systems?
  - Yes, in several case studies
- Can ArchJava aid *software evolution* tasks?
  - Preliminary experience suggests:
    - ArchJava highlights refactoring opportunities
      - Library of connectors: Most of the code is written in a library class, RiverCallConnector
    - ArchJava encourages loose coupling
    - ArchJava may aid defect repair

Conclusion

- ArchJava integrates architecture with Java code
- Control architecture consistency
  - Keeps architecture and code synchronized
- Initial experience
  - ArchJava can express real program architectures
  - ArchJava may aid in software evolution tasks
- ***Download the ArchJava compiler and tools***

http://archjava.fluid.cs.cmu.edu/