

FlowServer

A Framework for Workflow-oriented Scripting in Java Applications

Dr. Mathias Richter

Cap Gemini Switzerland



Overview



- Processes in Applications

Overview

- Processes in Applications
- FlowServer Architecture

Overview

- Processes in Applications
- FlowServer Architecture
- Workflow Definition = Java + Scripting

Overview

- Processes in Applications
- FlowServer Architecture
- Workflow Definition = Java + Scripting
- FlowServer Architecture (revisited)

Overview



- Processes in Applications
- FlowServer Architecture
- Workflow Definition = Java + Scripting
- FlowServer Architecture (revisited)
- Design & Implementation

Overview



- Processes in Applications
- FlowServer Architecture
- Workflow Definition = Java + Scripting
- FlowServer Architecture (revisited)
- Design & Implementation
- Conclusions

Processes in Applications

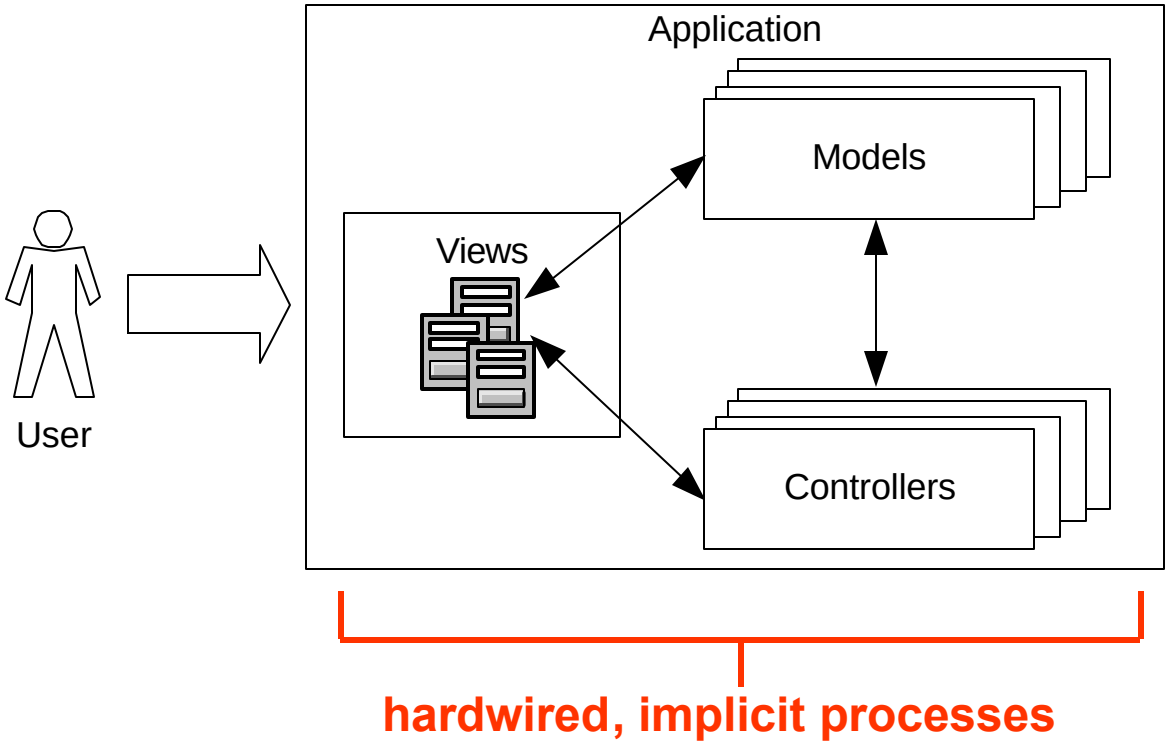
- „Enterprise Java“
  - Forms-based applications
  - numerical/alphanumerical data
  - some degree of modality
  - (almost) no direct manipulation

Processes in Applications

- „Enterprise Java“
  - Forms-based applications
  - numerical/alphanumerical data
  - some degree of modality
  - (almost) no direct manipulation
- User interaction
  - sequence of input-feedback-cycles
  - $\Rightarrow$ Processes

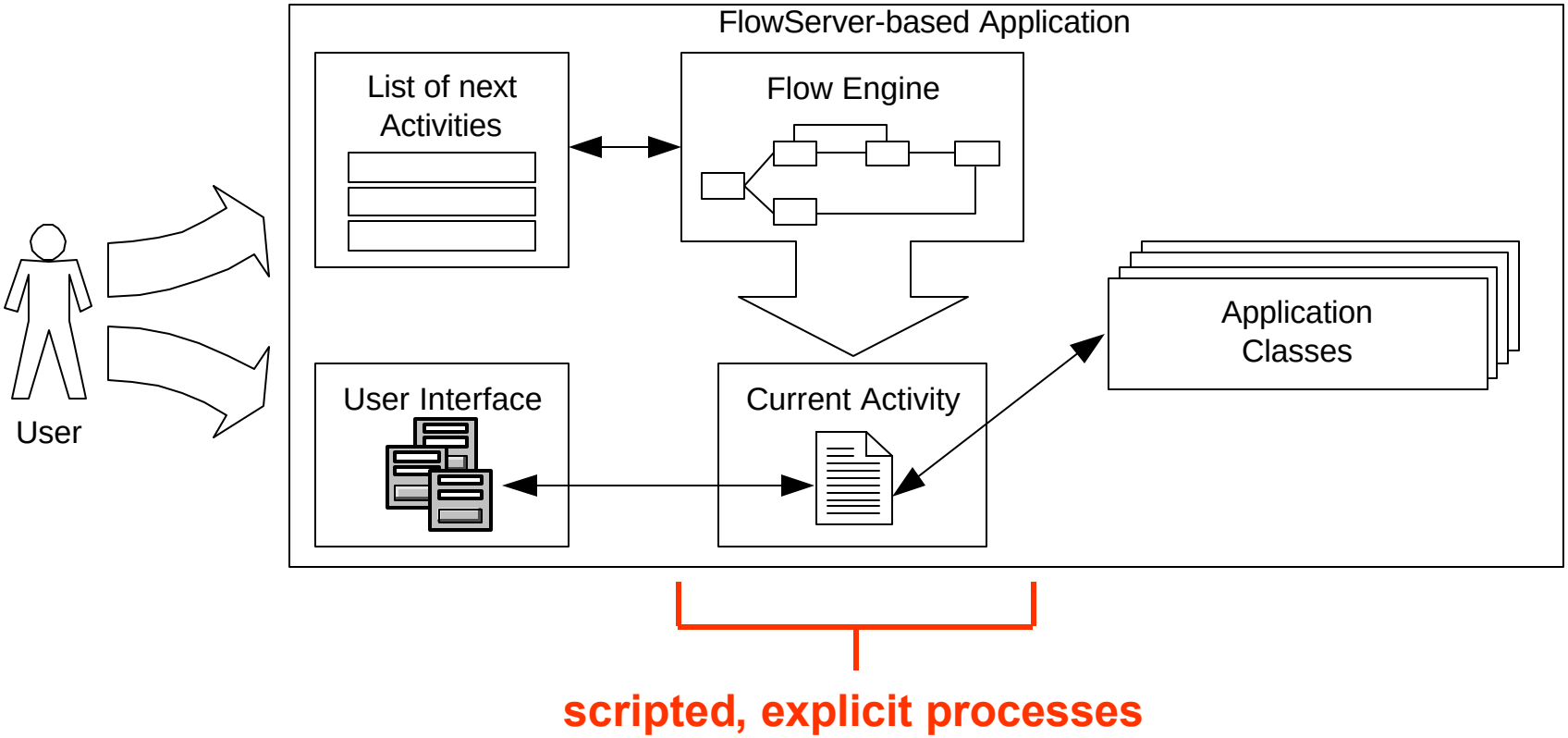
Processes in Applications

- MVC-based architectures (a rude oversimplification)



FlowServer Architecture

- Factoring the processes out of the objects:



Workflow Definition I

```
process name
{ local data { ... }
  input data { ... }
  output data { ...}

  activities
  { activity name
    { local data { ... }
      input data { ... }
      output data { ...}

      startup { ... }
      precondition { ... } on failure: name;
      postcondition { ... } on failure: name;
      execute { ... }
      traverse { ... }
      teardown { ... }
    }
  }

  structure
  { first name;
    after name: name;
    after name: name mapping (variable to variable);
    after name: (name or name or name);
    last name;
  }
}
```

process state

atomic (stateful) activities

map of activities & processes

Workflow Definition II

```
process name
{ local data { ... }
  input data { ... }
  output data { ...}

  activities
  { activity name
    { local data { ... }
      input data { ... }
      output data { ...}

      startup { ... }
      precondition { ... } on failure: name;
      postcondition { ... } on failure: name;
      execute { ... }
      traverse { ... }
      teardown { ... }
    }
  }

  structure
  { first name;
    after name: name;
    after name: name mapping (variable to variable);
    after name: (name or name or name);
    last name;
  }
}
```

Java variable declarations for storing process state

Workflow Definition III

```
process name
{ local data { ... }
  input data { ... }
  output data { ...}

  activities
  { activity name
    { local data { ... }
      input data { ... }
      output data { ...}

      startup { ... }
      precondition { ... } on failure: name;
      postcondition { ... } on failure: name;
      execute { ... }
      traverse { ... }
      teardown { ... }
    }
  }

  structure
  { first name;
    after name: name;
    after name: name mapping (variable to variable);
    after name: (name or name or name);
    last name;
  }
}
```

Java variable declarations for storing activity state

Activity semantics defined using Java code

Workflow Definition IV

```
process name
{ local data { ... }
  input data { ... }
  output data { ...}

  activities
  { activity name
    { local data { ... }
      input data { ... }
      output data { ...}

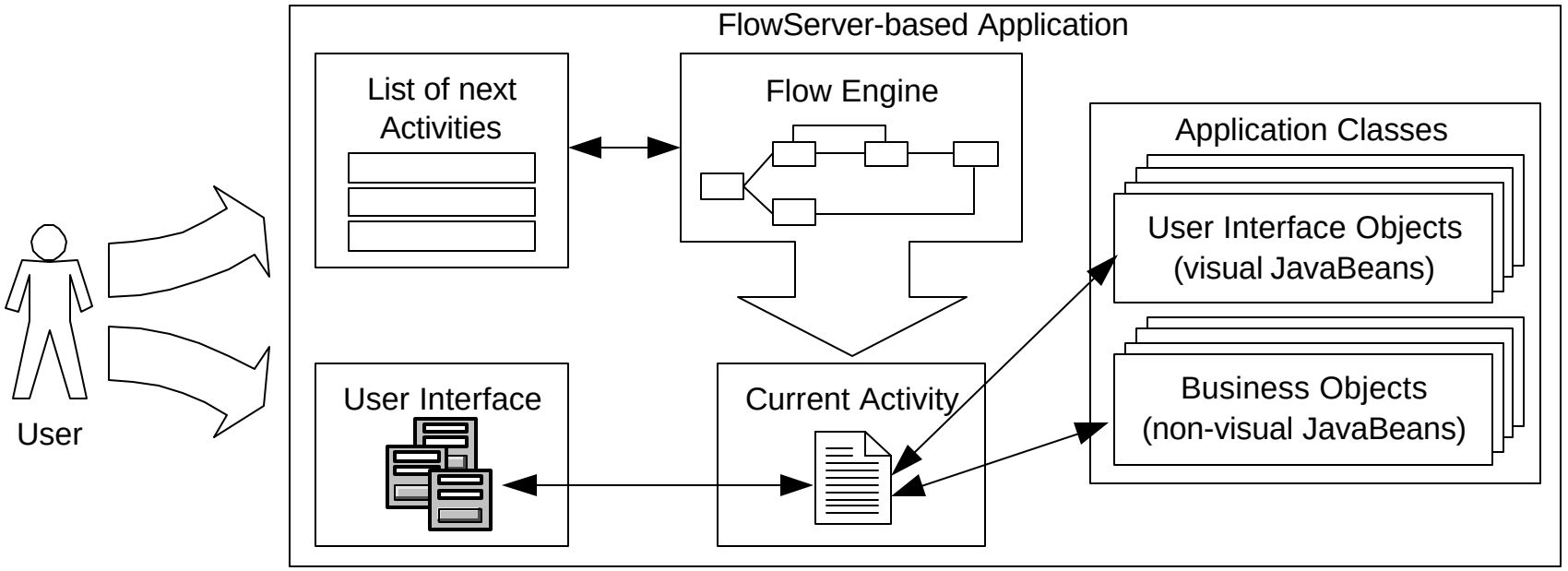
      startup { ... }
      precondition { ... } on failure: name;
      postcondition { ... } on failure: name;
      execute { ... }
      traverse { ... }
      teardown { ... }
    }
  }

  structure
  { first name;
    after name: name;
    after name: name mapping (variable to variable);
    after name: (name or name or name);
    last name;
  }
}
```

Connect activities/subprocesses & map states between them

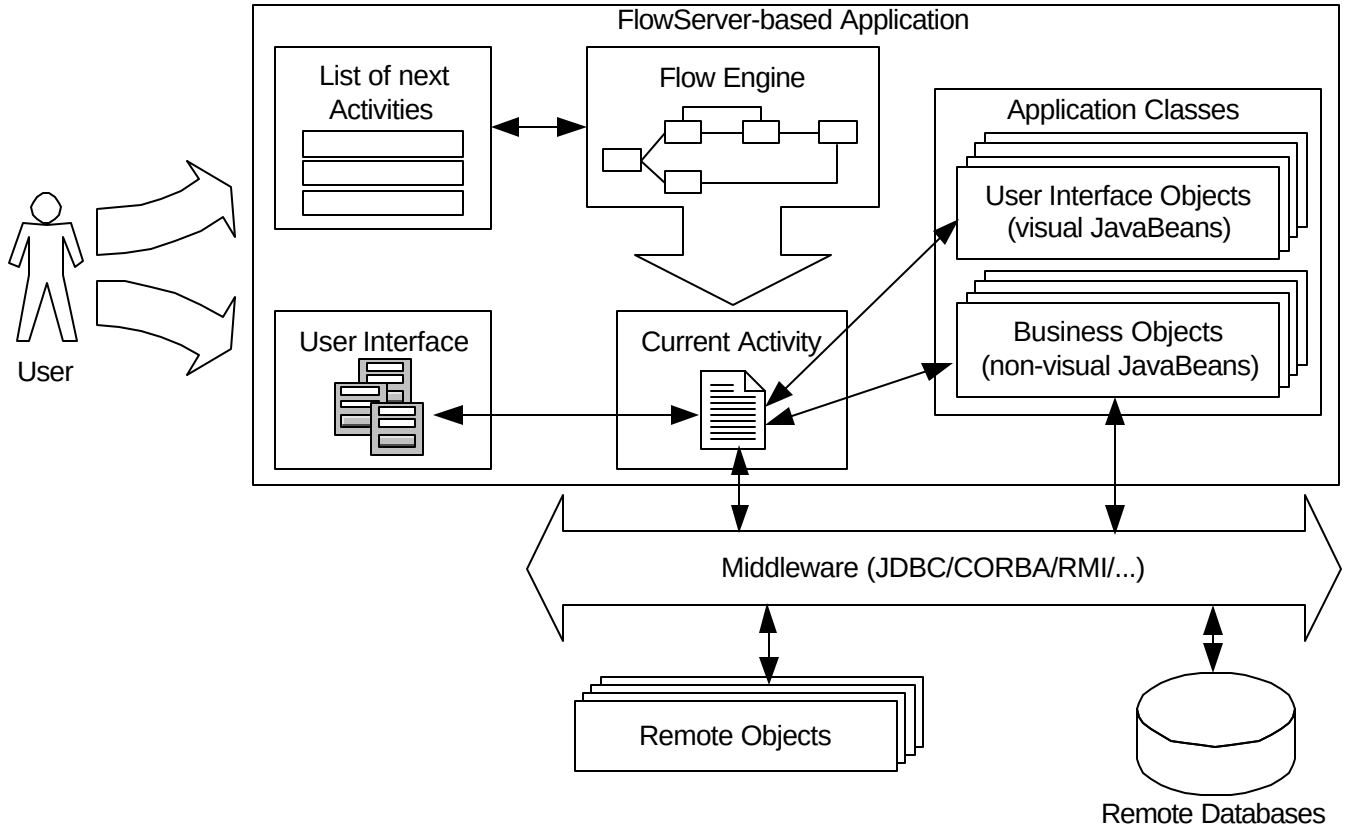
Architecture (revisited)

- Type I (local UI, local processes, local components)



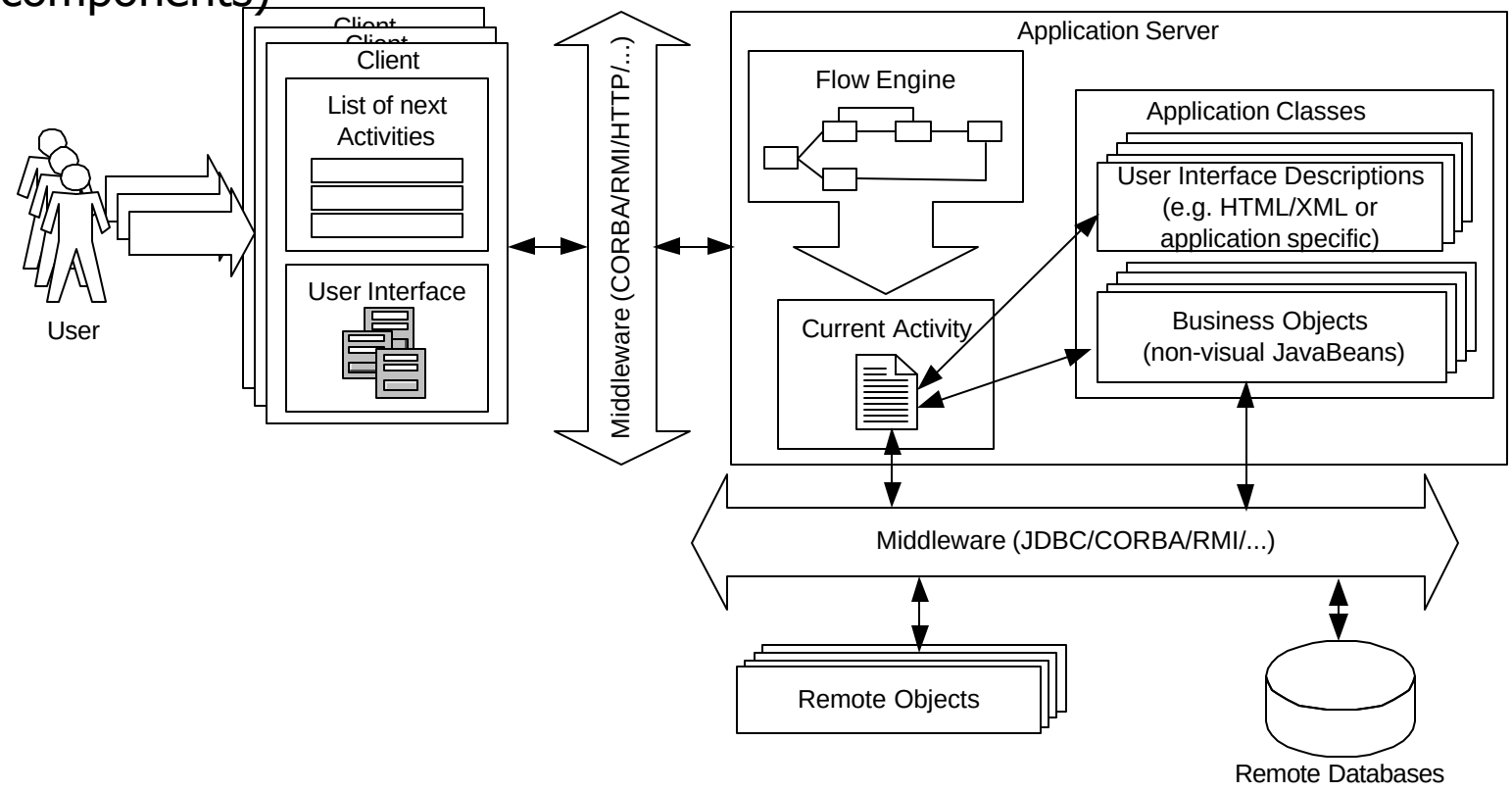
Architecture (revisited)

- Type II (local UI, local processes, distributed components)

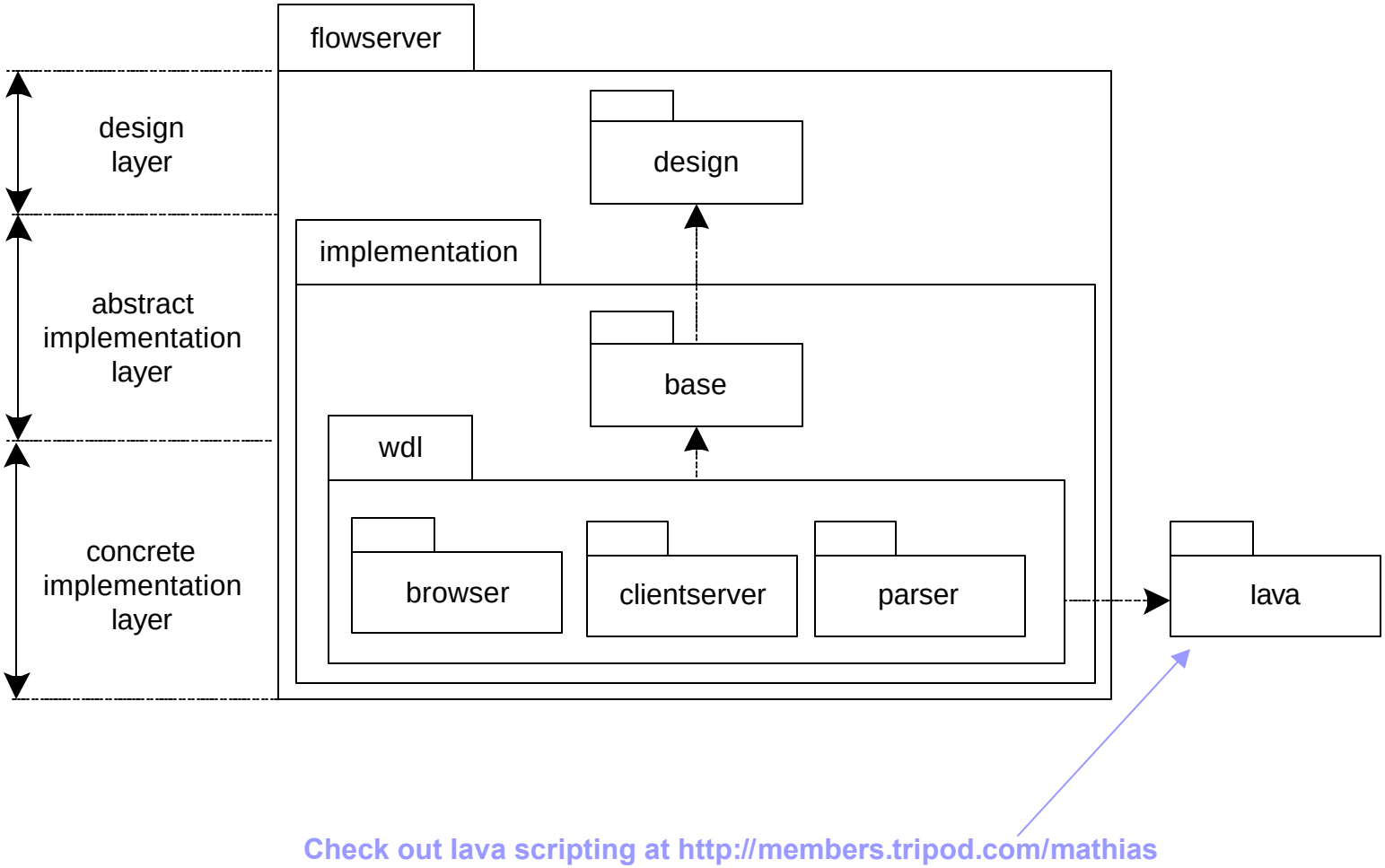


Architecture (revisited)

- Type III (distributed UI, distributed processes, distributed components)



Design & Implementation I



Design & Implementation II

- Type I/II architecture
  - use 3-5 classes

Design & Implementation II

- Type I/II architecture
  - use 3-5 classes

```
private WDLFlowEngine fEng;
private WDLRepository fRep;
public void startApp()
{           fRep = new WDLRepository();
            fRep.processFile( „wdlFileName“, false );
            fEng = new WDLFlowEngine( fRep.getDefinition( „processName“ ) );
            fEng.start();
            while ( !fEng.isFinished() )
            {           WDLControlItem[] items = fEng.getNext();
                        this.displayNavigation( items );
            }
}
public void advance( WDLControlItem chosen )
{           fEng.advance( chosen );
}
```

Design & Implementation II

- Type I/II architecture
  - use 3-5 classes

```
private WDLFlowEngine fEng;
private WDLRepository fRep;
public void startApp()
{           fRep = new WDLRepository();
            fRep.processFile( „wdlFileName“, false );
            fEng = new WDLFlowEngine( fRep.getDefinition( „processName“ ) );
            fEng.start();
            while ( !fEng.isFinished() )
            {           WDLControlItem[] items = fEng.getNext();
                        this.displayNavigation( items );
            }
}
public void advance( WDLControlItem chosen )
{           fEng.advance( chosen );
}
```

- Type III architecture
  - use/extend ~10 classes, implement ~3 interfaces

Conclusions

- Process-oriented, lightweight, embeddable

Conclusions

- Process-oriented, lightweight, embeddable
- Scripting-based (Iava), can be compiled (Java)

Conclusions

- Process-oriented, lightweight, embeddable
- Scripting-based (Iava), can be compiled (Java)
- Supports different degrees of distribution

Conclusions

- Process-oriented, lightweight, embeddable
- Scripting-based (Iava), can be compiled (Java)
- Supports different degrees of distribution
- Type III architectures involve some work (UI)

Conclusions

- Process-oriented, lightweight, embeddable
- Scripting-based (Iava), can be compiled (Java)
- Supports different degrees of distribution
- Type III architectures involve some work (UI)
- Is in productive use (type III architecture)

Conclusions

- Process-oriented, lightweight, embeddable
- Scripting-based (Iava), can be compiled (Java)
- Supports different degrees of distribution
- Type III architectures involve some work (UI)
- Is in productive use (type III architecture)
- As all frameworks: we'll do it better next time ☺