Presentation of Open Simulation Architecture and Open Simulation Instrumentation Framework

Judicael RIBAULT[1]
judicael.ribault@sophia.inria.fr

1- MASCOTTE, INRIA, I3S, CNRS, Univ. Nice Sophia, Sophia Antipolis, France

COMRED, March 1, 2010

Outline

- OSA
  - Motivations
  - Softwares
  - Objectives
  - Examples
  - Conclusion
- OSIF
  - Motivations
  - Softwares
  - Objectives
  - Conclusion

Philosophy

To be or not to be ?

"What is a simulation ?"

- A representation of a situation with similar but simpler model
  - Can easily be manipulated
  - Can show the eventual real effects of a given situation

- Computer simulations:
  - model real-life or hypothetical situation
  - change variables easily

Philosophy

To be or not to be ?

"Do we build our own simulator or reuse an existing one ?"

- There is no perfect simulator BUT
  - All the elements of your perfect simulator already exist.
  - if not, build only the missing part !

"Which confidence level can I have in my results ?"

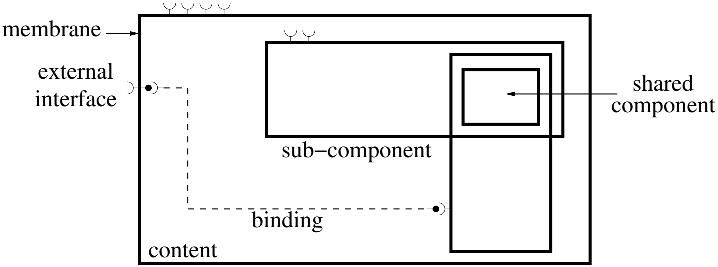
- More reusing → less validation

"Which credibility in comparing results with others studies ?"

- More sharing → more credibility

Component-based software engineering
CBSE

- separation of concerns
  - better understanding and maintainability
- similar to object-oriented programming
  - but at the general architecture software level
  - monolithic executable versus reusable bricks



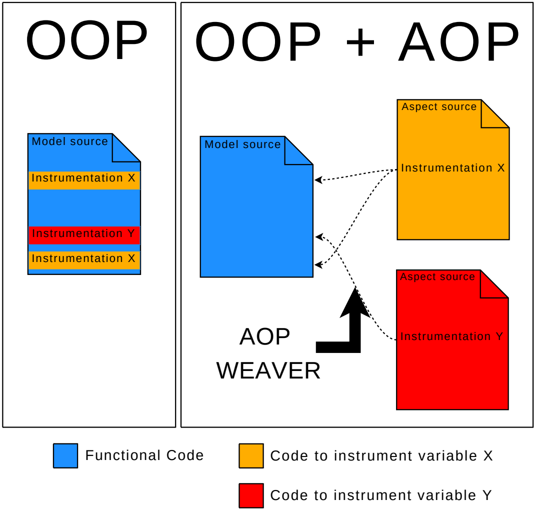
Fractal

- Primitive Component
  - Code Container
  - Client-server interactions
- Composite component
  - Hierarchical grouping
  - Strong Isolation
  - Shared sub-Component
- Dynamic (re)configuration
  - Factory & Template Components
  - Dynamic bindings
- Introspection
- Extensibility of non-functional services
  - Controllers
- Architecture Description Language

FractalADL

- Read Architecture from XML Files
  - Read definitions from multiple files
    - Each file embeds its own syntax
    - possibly several DTDs (means extension is unlimited)
  - Overloading capabilities
    - An XML definition may be partly overloaded by another
- Default ADL parser handles several concerns
  - Attribute settings, Component Naming, Distributed Execution
  - Modular, extensible structure (hierarchical comp)
    - New concerns may be added
    - Existing may be replaced

Aspect-Oriented Programming



Aspect-Oriented Programming

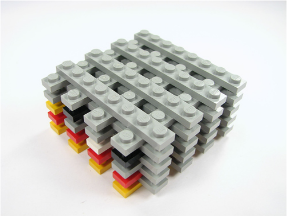
- Paradigm for modularizing applications with many concerns
- Goals are :
  - Separation of concerns
    - AOP instructions are placed in separate source files
  - Crosscutting interactions and Dependencies inversion
    - Identify particular instructions in an existing Code
    - To Apply Pre/Post/Replacement Processings
    - To Enrich/Extend existing code

Maven

- Maven is an Open Source project from Apache
- Maven manages among other
  - Builds
  - Documentation
  - Reporting
  - Dependencies
  - SCMs
  - Releases
  - Distribution
- Maven Archetype help starting new project from templates

Objectives

And OSA was born ...

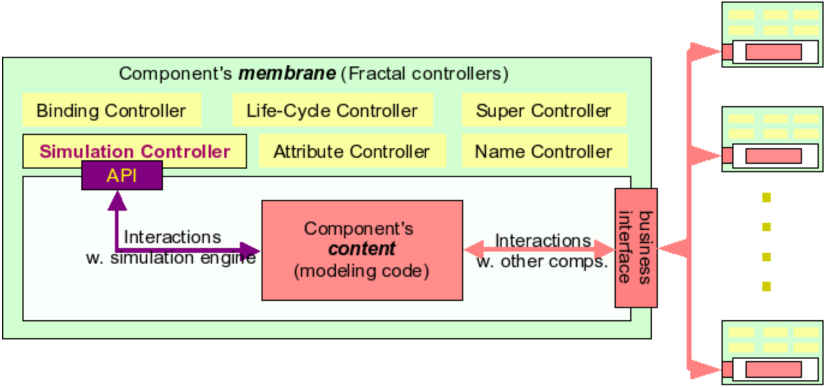


- Separation of modeling concerns
  - → component-based framework
- Separation of simulation concerns
  - → layered approach
- Bridge between concerns
  - → aspect-oriented programming
- Backup and replayability
  - → maven project management

- GOAL
  - build or reuse existing parts from others simulators and third-party tools

Open Simulation Architecture

A component-based framework



The Fractal Component Framework

Example

P2P system

- Simulation of a P2P system:
  - Components: Peer, Network, Simulator
  - Bindings: Peer<->Network, Peer<->Simulator, Network<->Simulator
  - Controls: LocalSimulator
- Objectives:
  - hundreds of thousands of peers, one network, 1 simulator

Annotations

Fraclet

```java
@Component
@Membrane(controller = "simPrimitive")
public class Hello implements HelloItf {

  @Requires(name = "world")
  private WorldItf world;

  @Controller(value = "simulation-controller")
  private SimulationControllerAPI simulationController_;

  public void printHello() {
    simulationController_.waiting(10);
    world.print("Hello");
  }

}
```

Modeling Example

Helloworld

```
<definition name="fr.inria.osa.models.helloworld">

    <component name="Hello"
        definition="fr.inria.osa.models.Hello"/>

    <component name="World"
        definition="fr.inria.osa.models.World"/>

    <binding client="Hello.world" server="World.world" />

</definition>

```

Modeling Example

Man-in-the-middle

```xml
<definition name="fr...hellomitm" extends="fr...helloworld">

    <component name="Hello"/>
    <component name="World"/>

    <component name="Mitm"
        definition="fr.inria.osa.models.Mitm"/>

    <binding client="Hello.world" server="Mitm.world" />
    <binding client="Mitm.world" server="World.world" />

</definition>
```

Modeling Example

Spyware

- ADL:

```
<definition name="fr...hellospy" extends="fr...helloworld">

    <component name="Hello"/>

    <component name="Spy"
        definition="fr.inria.osa.models.Spy">
    </component>

    <binding client="Hello.spy" server="Spy.spy" />

</definition>
```

- AOP:
  - before(Hello hello): execution(void Hello.printHello()) && this(hello){ . . . code . . . }

Modeling Example

All in one

```
<definition name="fr...Allinone"
    extends="fr...hellomitm,fr...hellospy">
</definition>
```

Conclusion

What is OSA

- OSA is a simulator framework
  - separation of concerns
  - each layer of the simulation could be replaced or improved
    - engine, model, scenario, instrumentation, deployment, ...
  - AOP enable bridge between concerns
    - such as between modeling and instrumentation
- OSA could be used
  - to build your perfect simulator
  - to conduct simulation studies
  - as a testbed for simulation algorithm, methodology, ...
  - to learn simulation focusing on a specific concern

Outline

- OSA
  - Motivations
  - Softwares
  - Objectives
  - Examples
  - Conclusion
- OSIF
  - Motivations
  - Softwares
  - Objectives
  - Conclusion

Motivation

- Simple API
  - instrumentation and modeling concerns are mixed together
    - useless data → slow down the simulation
    - missing data → need source modification
  - consume a lot of disk space / bandwidth
- Data processing
  - filter useful data
  - take a long time
  - often not reusable

COSMOS

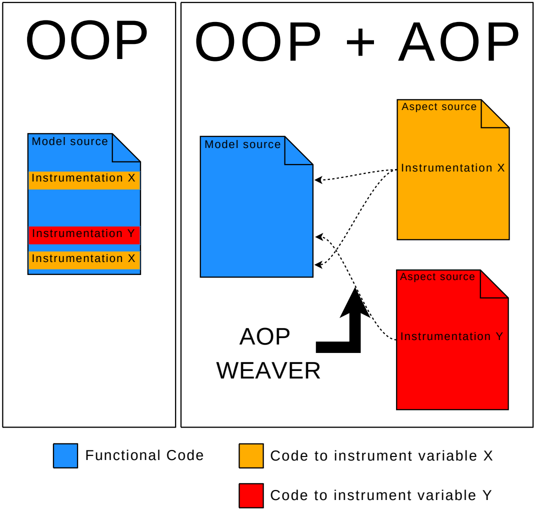
COntext entitieS coMpositiOn and Sharing

- Component-based framework for managing context data in ubiquitous applications
- Instrumentation built as a graph of processing nodes
- 3 COSMOS entities: collector, processor, policy
  - Placement of processors on the context nodes
    - passive/active
    - observation/notification
    - blocking/non-blocking
- Based on Fractal

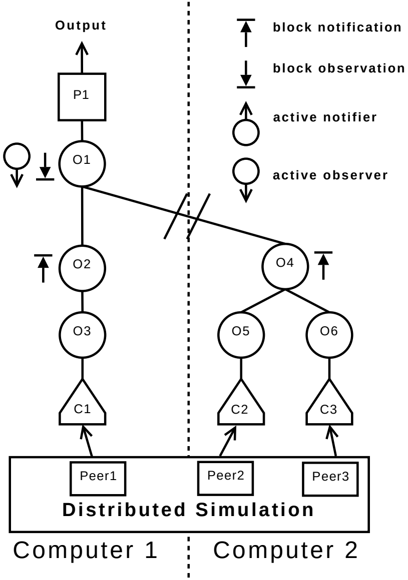
Objectives

- Separate instrumentation concern from modeling concern
  - → Aspect-Oriented Programming
- Live process data
  - → COSMOS
- Build reusable processing
  - → COSMOS (based on Fractal)
- Compose instrumentation on demand
  - → FractalADL

Separation of Concerns



Live-processing



Composition

- COSMOS Based on Fractal
  - → FractalADL allow composition by extension and overloading
- Benefits:
  - keep simple
    - → easier to manage and maintain
    - → reuse more
  - build complex

Real experiments processing

- COSMOS is for real applications
  - We succesfully use COSMOS for instrumentation and data processing in simulation
- Apply the same data processing on real experiment and simulation
  - validation of simulation results
  - sharing processing $\rightarrow$ more confidence

conclusion OSIF

- separation of concerns
  - favor model reuse
- live processing
  - save disk space / bandwidth / processing time
- composition
  - build / manage / maintain simple instrumentation and data processing
  - reuse data processing
  - build complex data processing by composition
- apply data processing on real experiment
  - reuse data processing
  - validate simulation
  - confidence increase

Conclusion

Actual and Future works

- OSA actually support
  - James II
    - plugins
    - DEVS engine
  - COSMOS
  - Scave (Omnet++ post-processing tool)
  - Deployment
    - FractalBF (RMI, RESTful, WebService)
    - FDF
- OSA could support in the near future
  - YOUR works :)

Conclusion

Thank you

- Contact:
  - Olivier Dalle   olivier.dalle@sophia.inria.fr
  - Judicael Ribault   judicael.ribault@sophia.inria.fr
- Website:
  - http://osa.inria.fr