Generating Safe GCM Components

Antonio Cansado Eric Madelaine

INRIA Sophia Antipolis

GridCOMP - Pekin, 2007

INRIA

Outline

Outline

Reusing and Assembling Components

- Safe Assembly of Components
  - Static typing of bound interfaces
  - Compatibility of dynamic behaviour
    - Formal specification of Components
- Choice
  - Integrate ADL and BDL
- Difficulty
  - Provide a framework for non-specialists

Reusing and Assembling Components

- Safe Assembly of Components
  - Static typing of bound interfaces
  - Compatibility of dynamic behaviour
    - Formal specification of Components
- Choice
  - Integrate ADL and BDL
- Difficulty
  - Provide a framework for non-specialists

Reusing and Assembling Components

- Safe Assembly of Components
  - Static typing of bound interfaces
  - Compatibility of dynamic behaviour
    - Formal specification of Components
- Choice
  - Integrate ADL and BDL
- Difficulty
  - Provide a framework for non-specialists

Outline

Structure of the Framework

Specify using High-Level Specification Language

- Vercors Component Environment (VCE)
- UML 2

Generate behavioural models

- Validate and Verify

Generate Java control code

- Strong guarantees

Structure of the Framework

Specify using High-Level Specification Language

- Vercors Component Environment (VCE)
- UML 2

Generate behavioural models

- Validate and Verify

Generate Java control code

- Strong guarantees

Structure of the Framework

Specify using High-Level Specification Language

- Vercors Component Environment (VCE)
- UML 2

Generate behavioural models

- Validate and Verify

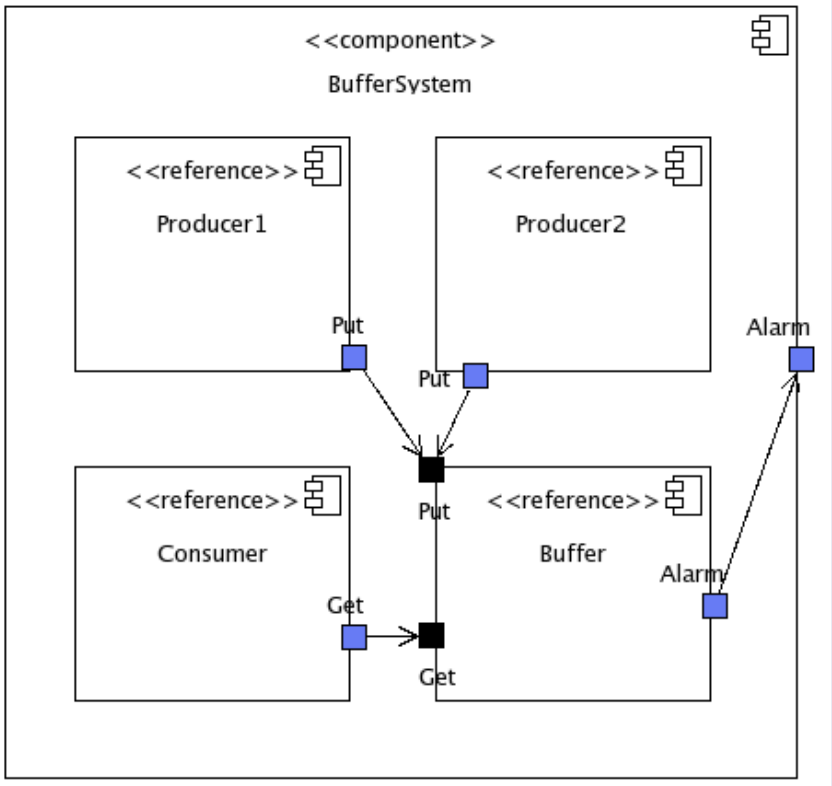
Generate Java control code

- Strong guarantees

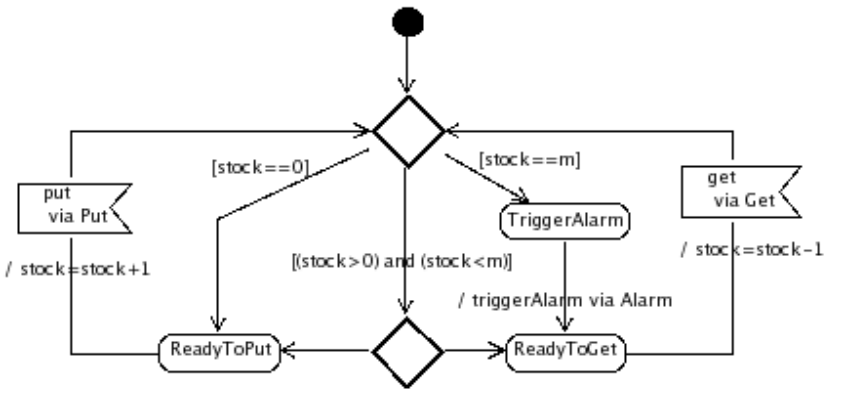
Outline

Unifying ADL and BDL

Component Diagram



State Machine Diagram



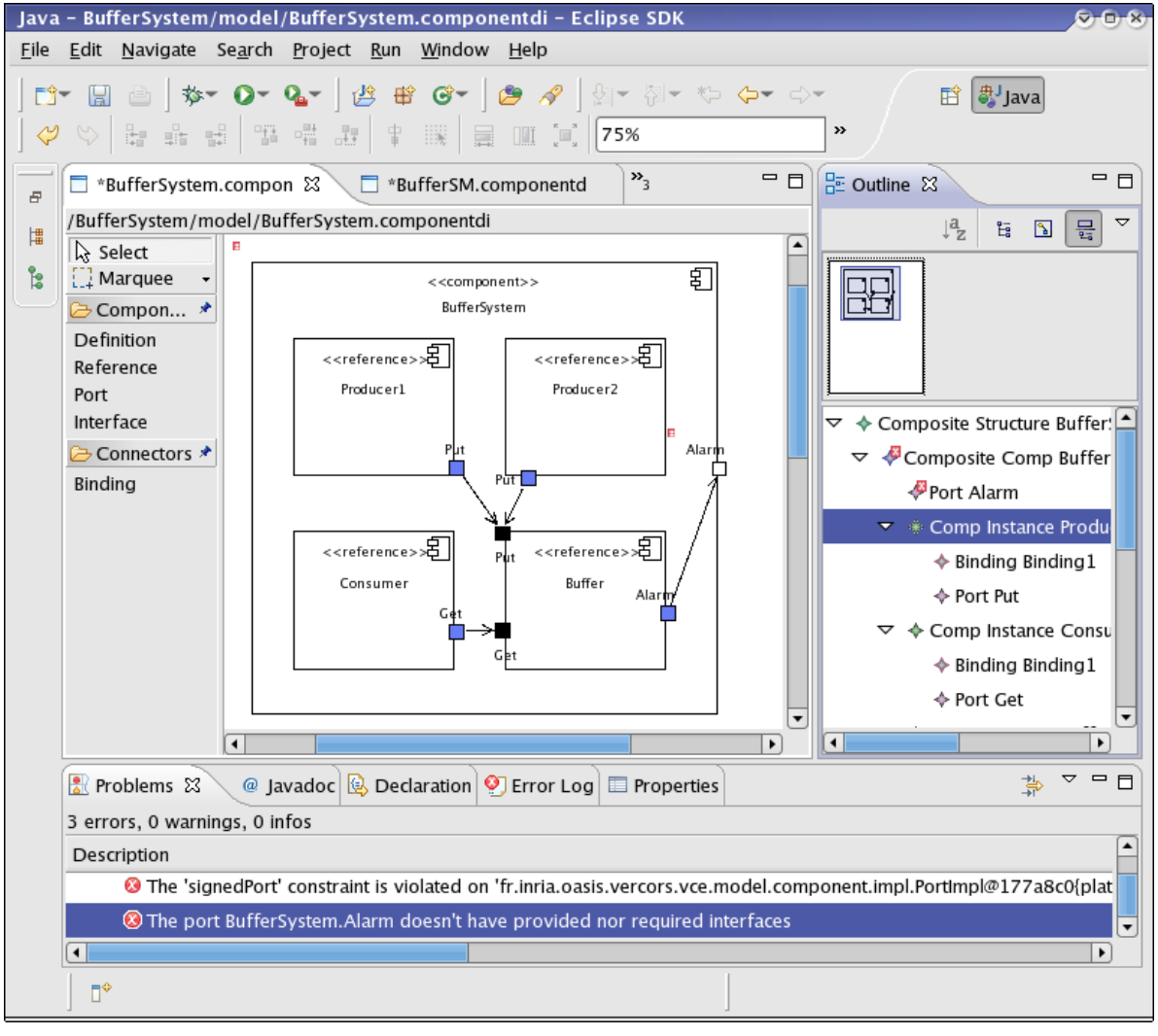
Data influencing the control-flow and the topology

INRIA

Prototype

- Functional specification of components
- Component libraries
- *Bottom-up* and *Top-down* specification
  - Specification given as a State Machine
  - Implementation given as a composition of subcomponents
- Integrated into Eclipse as plugins
- Generation of behavioural model

Snapshot


Java – BufferSystem/model/BufferSystem.componentdi – Eclipse SDK
File Edit Navigate Search Project Run Window Help
Java
75%
*BufferSystem.compon
*BufferSM.componentd
/BufferSystem/model/BufferSystem.componentdi
Select
Marquee
Compon...
Definition
Reference
Port
Interface
Connectors
Binding
<<component>>
BufferSystem
<<reference>>
Producer1
<<reference>>
Producer2
Put
Put
Alarm
<<reference>>
Consumer
Put
<<reference>>
Buffer
Alarm
Get
Get
Outline
Composite Structure Buffer
Composite Comp Buffer
Port Alarm
Comp Instance Produ
Binding Binding1
Port Put
Comp Instance Consu
Binding Binding1
Port Get
Problems
Javadoc
Declaration
Error Log
Properties
3 errors, 0 warnings, 0 infos
Description
The 'signedPort' constraint is violated on 'fr.inria.oasis.vercors.vce.model.component.impl.PortImpl@177a8c0{plat
The port BufferSystem.Alarm doesn't have provided nor required interfaces

Validate and Verify

- Sound semantic model – pNets
  - Hierarchical, Parameterized Networks of Labelled Transition Systems
- Generate Behavioural Models
  - Functional and Non-Functional concerns
- Model-checking
  - Deadlocks, Reachability, Safety, Liveness
  - Properties specified as automata
  - Functional and Non-Functional verification

Validate and Verify

- Sound semantic model – pNets
  - Hierarchical, Parameterized Networks of Labelled Transition Systems
- Generate Behavioural Models
  - Functional and Non-Functional concerns
- Model-checking
  - Deadlocks, Reachability, Safety, Liveness
  - Properties specified as automata
  - Functional and Non-Functional verification

Validate and Verify

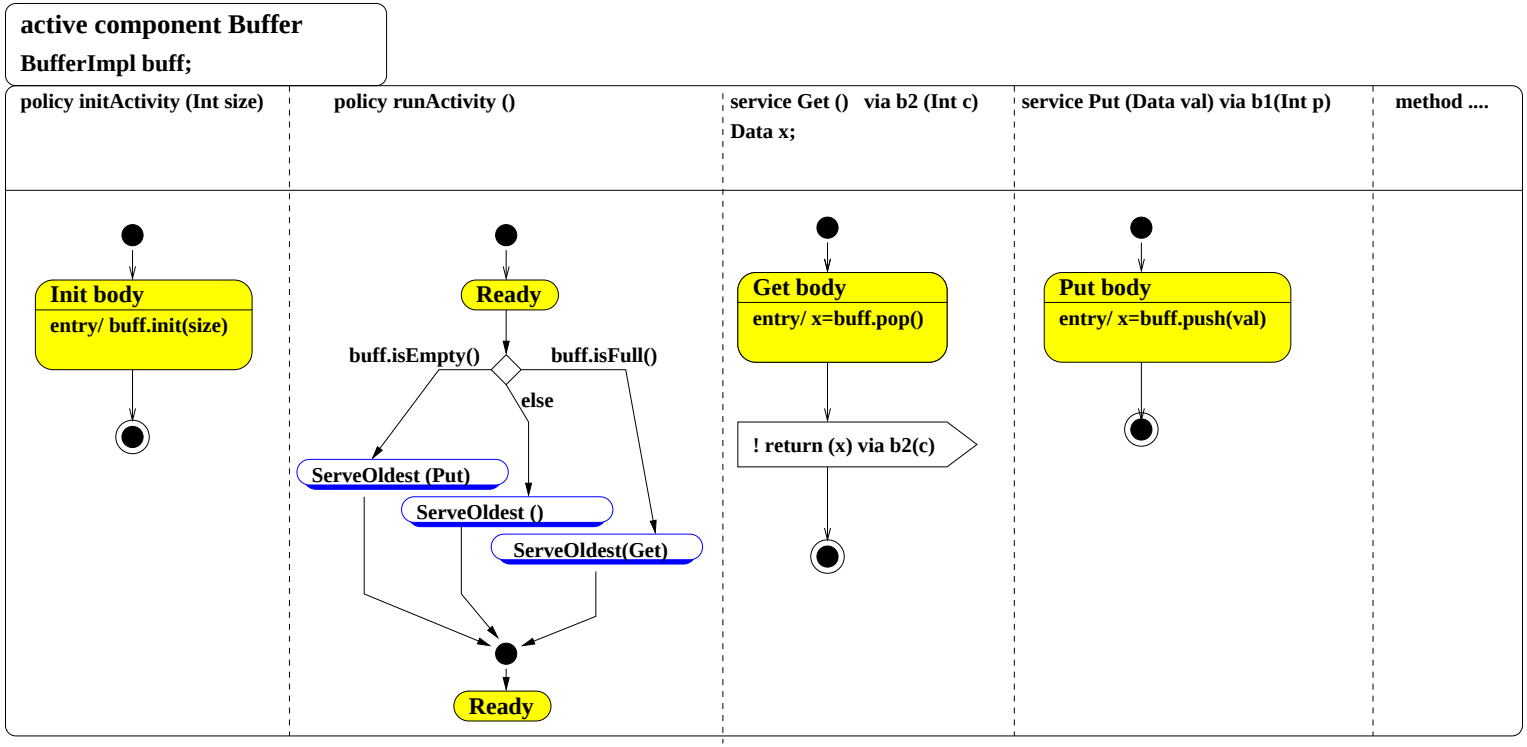
- Sound semantic model – pNets
  - Hierarchical, Parameterized Networks of Labelled Transition Systems
- Generate Behavioural Models
  - Functional and Non-Functional concerns
- Model-checking
  - Deadlocks, Reachability, Safety, Liveness
  - Properties specified as automata
  - Functional and Non-Functional verification

Outline

Missing Features

- Asynchronous components
  - Method calls performed on client interfaces → Future
  - Data-usage → Wait-by-necessity
- Collective interfaces
- Parameterized components
- NF management

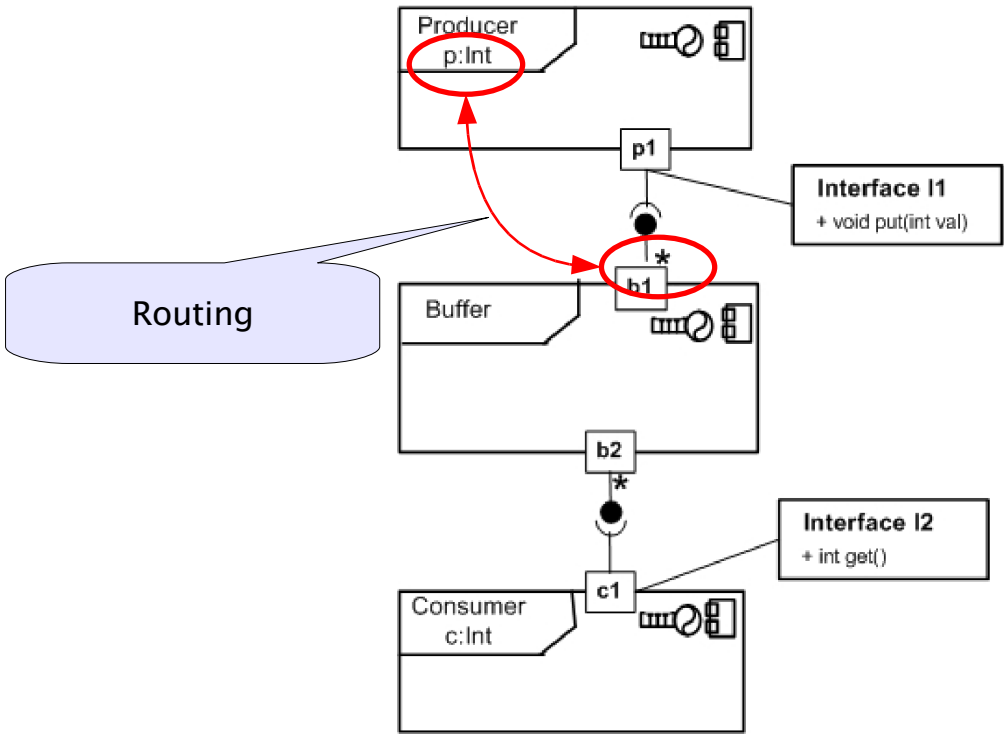
Decomposing the Behaviour



Generate skeletons for GCM components

INRIA

Parameterized Topologies



Code Generation

- Goal
  - Same behaviour as the specification
- Java code
  - GCM ADL
  - Final code of Fractal controllers
  - Skeletons of `runActivity()` and methods
- Hooks to fill-in final implementation
  - User-defined Business code

Code Generation

- Goal
  - Same behaviour as the specification
- Java code
  - GCM ADL
  - Final code of Fractal controllers
  - Skeletons of `runActivity()` and methods
- Hooks to fill-in final implementation
  - User-defined Business code

Code Generation

- Goal
  - Same behaviour as the specification
- Java code
  - GCM ADL
  - Final code of Fractal controllers
  - Skeletons of `runActivity()` and methods
- Hooks to fill-in final implementation
  - User-defined Business code

Outline

Generate GCM / ProActive code

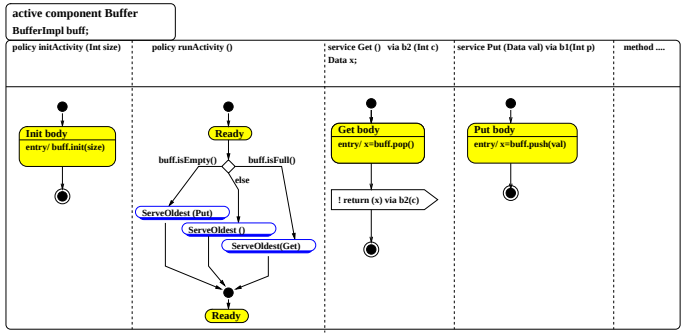
Fractal Controllers

```
public String[] listFc() {
  return new String[]{ CASHBOXEVENTIF_BINDING };
}
public Object lookupFc (String clientItfName) {
  if (CASHBOXEVENTIF_BINDING.equals(clientItfName))
    return cashBoxEventIf;
  return null;
}
public void bindFc (String clientItfName, Object serverItf) {
  if (CASHBOXEVENTIF_BINDING.equals(clientItfName))
    cashBoxEventIf = (CashBoxEventIf)serverItf;
}
public void unbindFc (String clientItfName) {
  if (CASHBOXEVENTIF_BINDING.equals(clientItfName))
    cashBoxEventIf = null;
}
```

Outline

Generation of Skeletons: From State Machines

- `runActivity()`
  - Service policy
- Service methods
  - Server Interfaces of Primitive Components
- Control-flow and data-flow
  - Control structure
  - Method calls performed on client interfaces
  - Data-usage points



INRIA

Service Policy

runActivity()

```
public void runActivity(Body body) {
   Service service = new Service(body);
   while (body.isActive()) {
      cashBoxEventIf.saleStarted();
      cashBoxEventIf.saleFinished();
      if ((new AnyBool()).prob(50)) {
         cashMode();
         cashAmount();
         service.blockingServeOldest("changeAmountCalculated");
         cashBoxEventIf.cashBoxClosed();
      }else
         creditCardMode();
} }
```

Control-Flow and Data-Flow

Service Method

```
public void pinEntered(PIN pin) {
  if (creditInfo != null) {
    Transaction transId = bankIf.validateCard(creditInfo, pin);
    if (ProActive.getFutureValue(transId) != null) {
      Info info = bankIf.debitCard(transId, runningTotal);
      if (ProActive.getFutureValue(info) != null){
        Sale sale = new SaleImpl(
                    new PaymentModeImpl(PaymentModeImpl.CREDIT),
                    products, runningTotal);
                    saleRegisteredIf.bookSale(sale);
        info.getInfo(); // wait-by-necessity
        init();
  ...
}
```

Conclusions and Perspectives

Short-term

- Tool for GCM Specification
- Validation of Behavioural properties
- Generation of Safe code

Long-term

- Multicast / Gathercast interfaces
- Specify Non-Functional controllers in the membrane

Conclusions and Perspectives

Short-term

- Tool for GCM Specification
- Validation of Behavioural properties
- Generation of Safe code

Long-term

- Multicast / Gathercast interfaces
- Specify Non-Functional controllers in the membrane

References

http://www-sop.inria.fr/oasis/Vercors

S. Ahumada, L. Apvrille, T. Barros, A. Cansado, E. Madelaine, and E. Salageanu.

Specifying Fractal and GCM Components With UML.

In *Proc. of the XXVI International Conference of the Chilean Computer Science Society (SCCC'07)*, Iquique, Chile, November 2007. IEEE.

A. Cansado, D. Caromel, L. Henrio, E. Madelaine, M. Rivera, and E. Salageanu.

A Specification Language for Distributed Components implemented in GCM/ProActive.

Lecture Notes in Computer Science. Springer, (To be published) 2007.

A. Cansado, L. Henrio, and E. Madelaine.

Towards real case component model-checking.

In *5th Fractal Workshop*, Nantes, France, July 2006.

INRIA