UNIVERSITY OF WESTMINSTER

Grid Integrated Development Environment

Vladimir Getov
V.S.Getov@westminster.ac.uk

GRIDS@WORK, 21 Oct. 2008

Overview

- Background – software crisis
- Component framework overview - GCM and GridCOMP
- Problem-to-solution development pipeline
- Component-oriented integrated environment
- Case Study: Biometric Identification System
- Initial experience and results
- Future Research Topics and Conclusions

Component Framework Overview

Starting point: Fractal component model.
The main technical features of the component framework are:

- **Support for primitive and composite distributed components and hierarchical composition.**
- **XML based architecture description language (ADL).**
- **Collective interfaces to comply with specific multi-way communication requirements.**
- **A comprehensive runtime API.**
- **Support for non-functional aspects such as component control, skeletons, and autonomy.**
- **Advanced component scheduling/deployment via the notion of virtual nodes and deployment descriptors.**

GCM and GridCOMP

GCM

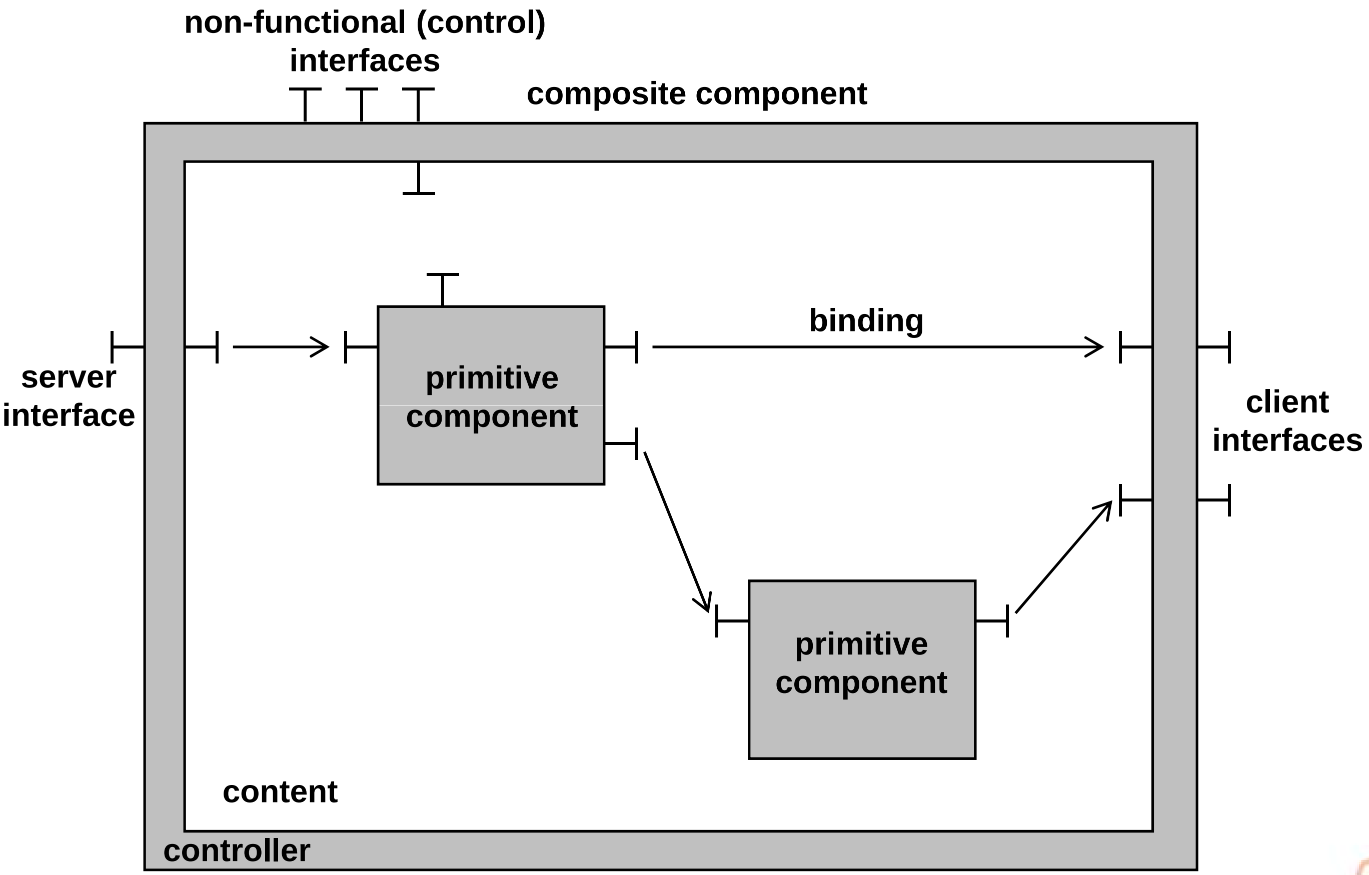
- A new advanced Grid Component Model (GCM) providing high level of abstraction and specifically designed for large scale dynamic Grid infrastructures.
- Specified within the CoreGRID Euopean project.

GridCOMP

- EU project: Grid programming with COMPonents.
- INRIA, ERCIM, Univ Westminster, Univ Pisa, CNR, IBM ZRL, Atos Origin, Grid Systems, Tsingua Univ, Univ Chile, Univ Melbourne
- Design and implementation of a Grid component framework based on GCM.
- Includes the development of a Grid IDE and several use case applications.
- Middleware reference platform implementation.

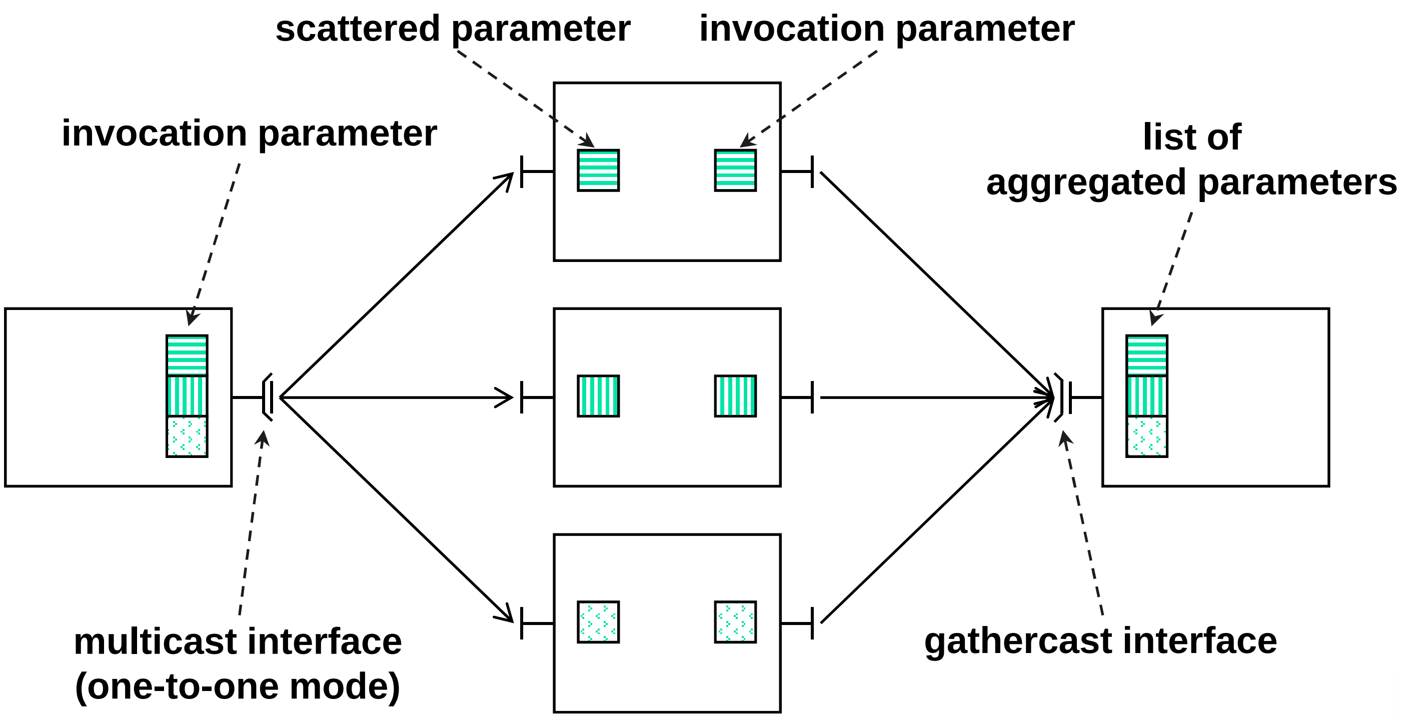
GridCOMP Component Framework Overview

Hierarchical composition: all three components can be distributed



GridCOMP Component Framework Overview

Collective interfaces: The framework takes care of parallel invocations, data distribution, and synchronization.

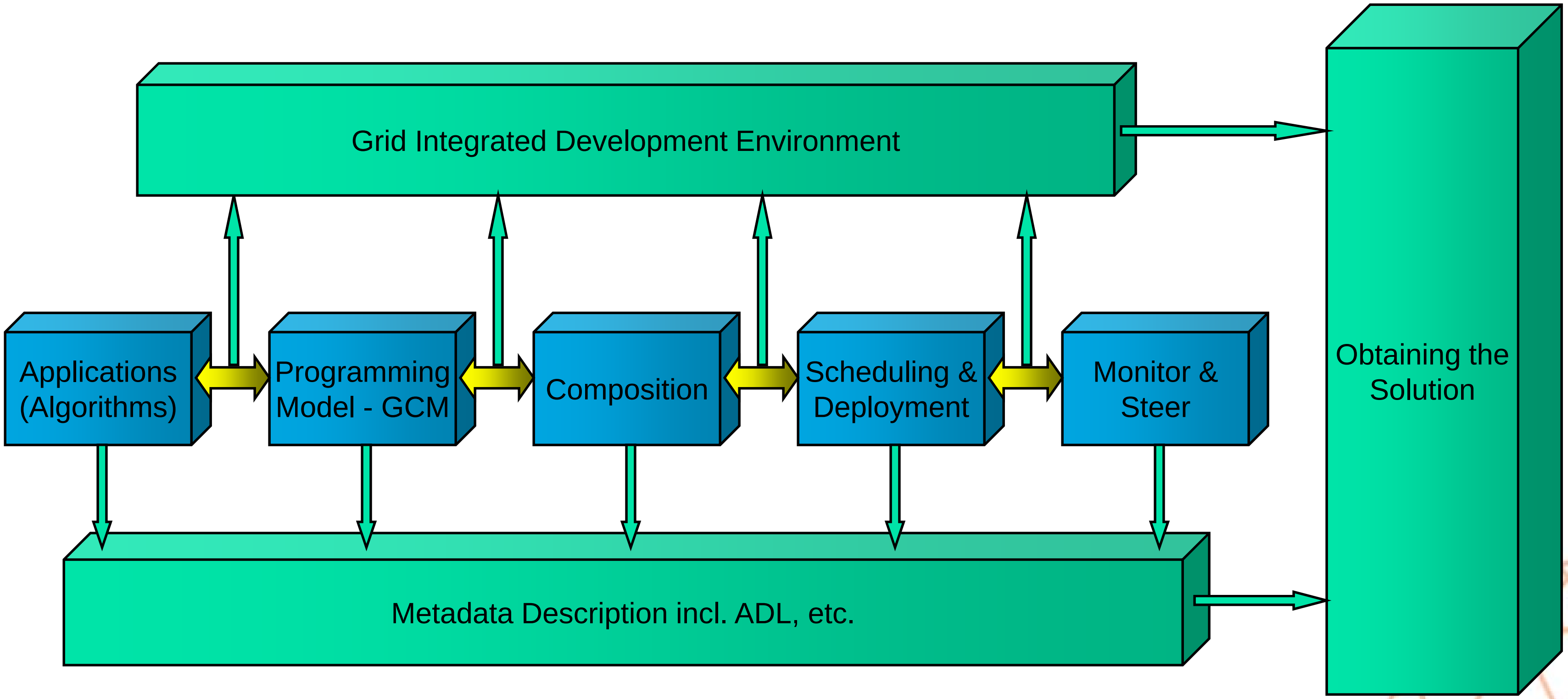


Approach

- Building on existing developments
- Integration of application and Grid infrastructure into a single adaptable platform
- Focus on two target users:
  - Composers/Developers
  - Data Centre Operators
- “Lets not restrict developers”
- “Lets protect operators from the details”

Component-Centric Problem-to-Solution Pipeline

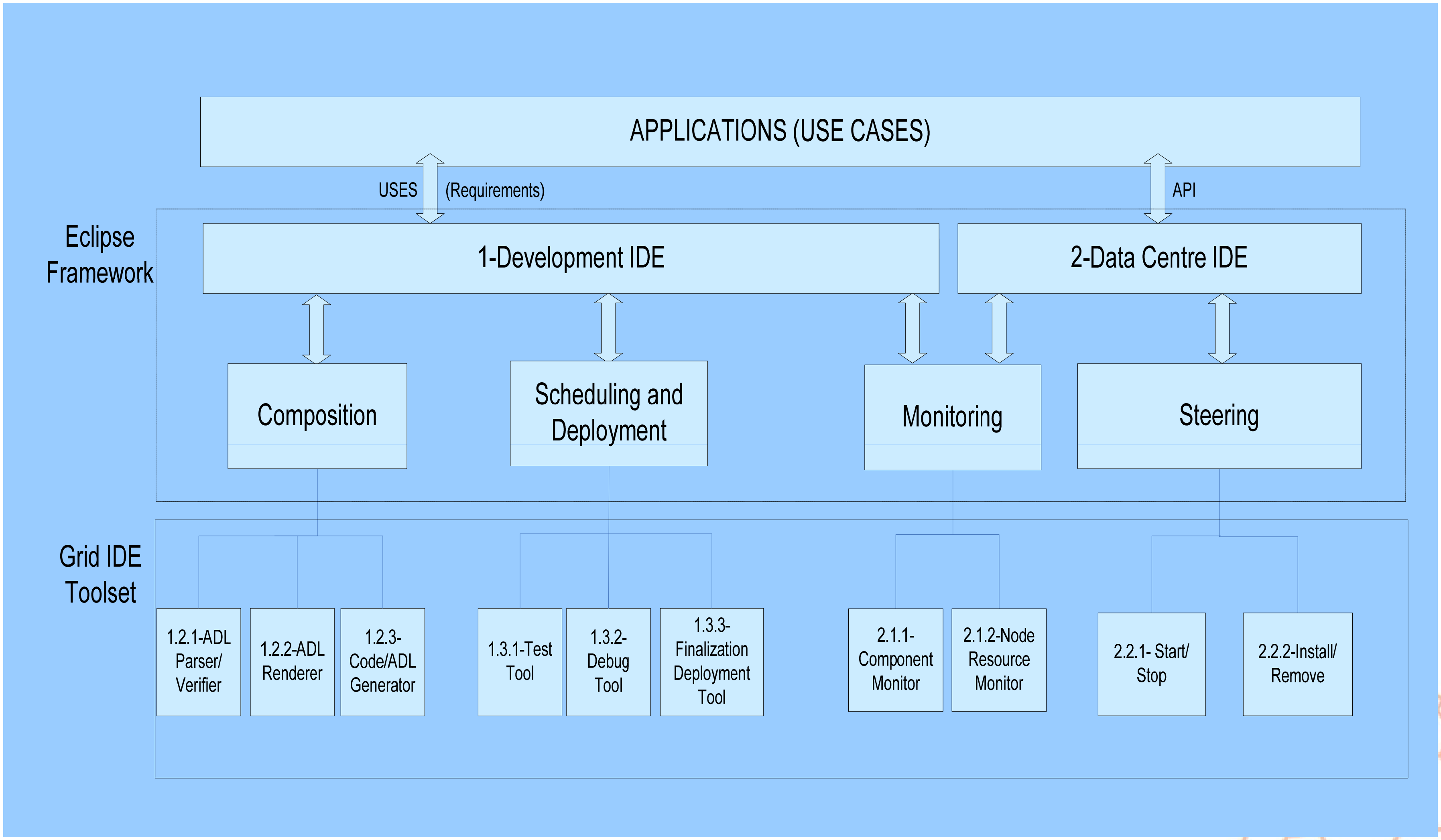
- Main issues: composition and dynamic properties – deployment, monitoring and steering
- Component-based Grid platform design methodology


Grid Integrated Development Environment
Applications (Algorithms)
Programming Model - GCM
Composition
Scheduling & Deployment
Monitor & Steer
Metadata Description incl. ADL, etc.
Obtaining the Solution

Strategy: Eclipse Framework for GIDE

- Simplify complexity through graphical composition/tools
- But, allowing ONLY graphical composition can be inflexible and inefficient
- Support for 3 levels of Development
  - Graphical Composition
  - Based on GCM and using ADL
  - Source code level
- Seamless integration with Eclipse
  - Widely supported with many potential plug-ins

Grid IDE Architecture - Core Block Diagram



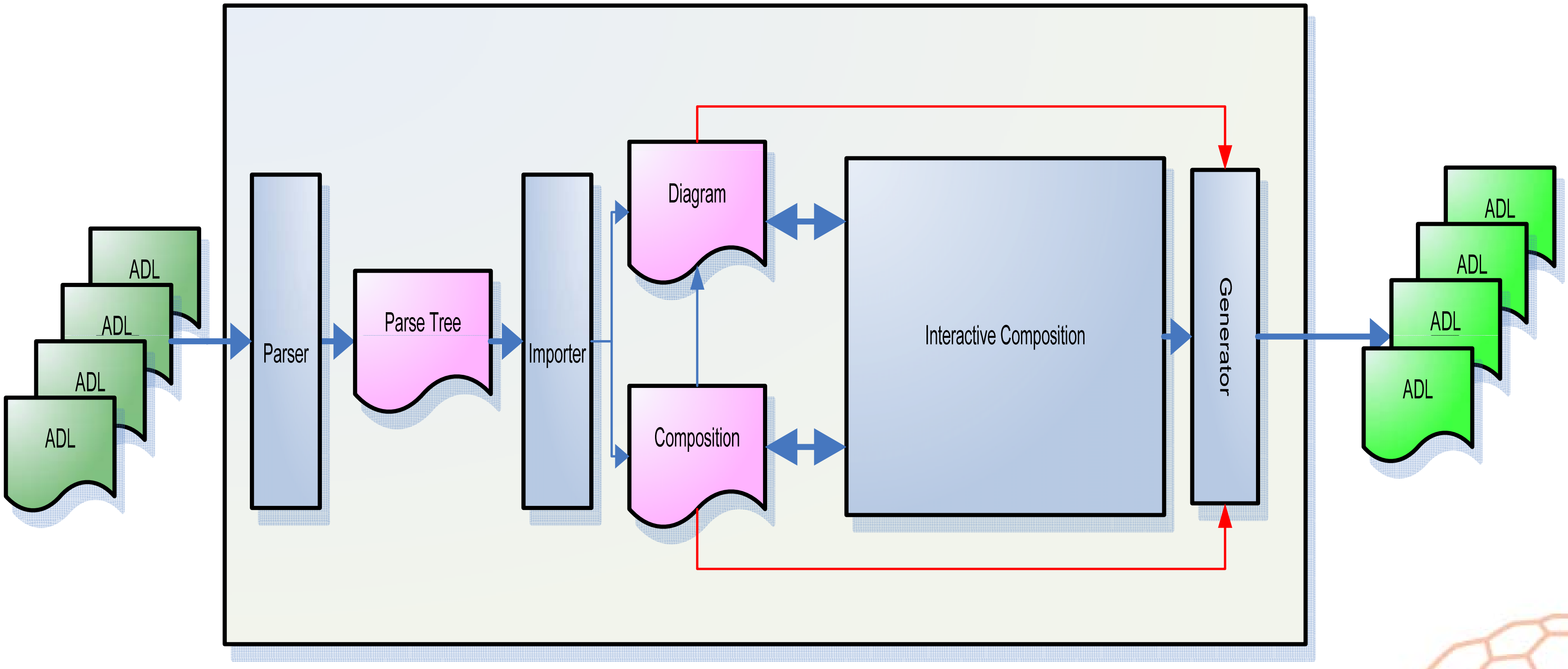
Eclipse Framework – Perspectives Overview

- Composition Perspective
  - Graphical but also allow code editing
- Deployment and/or Scheduling Perspective
  - Drag/drop to deployment list
- Monitoring Perspective
  - Dynamic component architecture view
  - Host and node info via plug-ins
  - Resource list view
- Steering Perspective
  - Basic steering (start/stop components)
  - More advanced features based on non-functional support

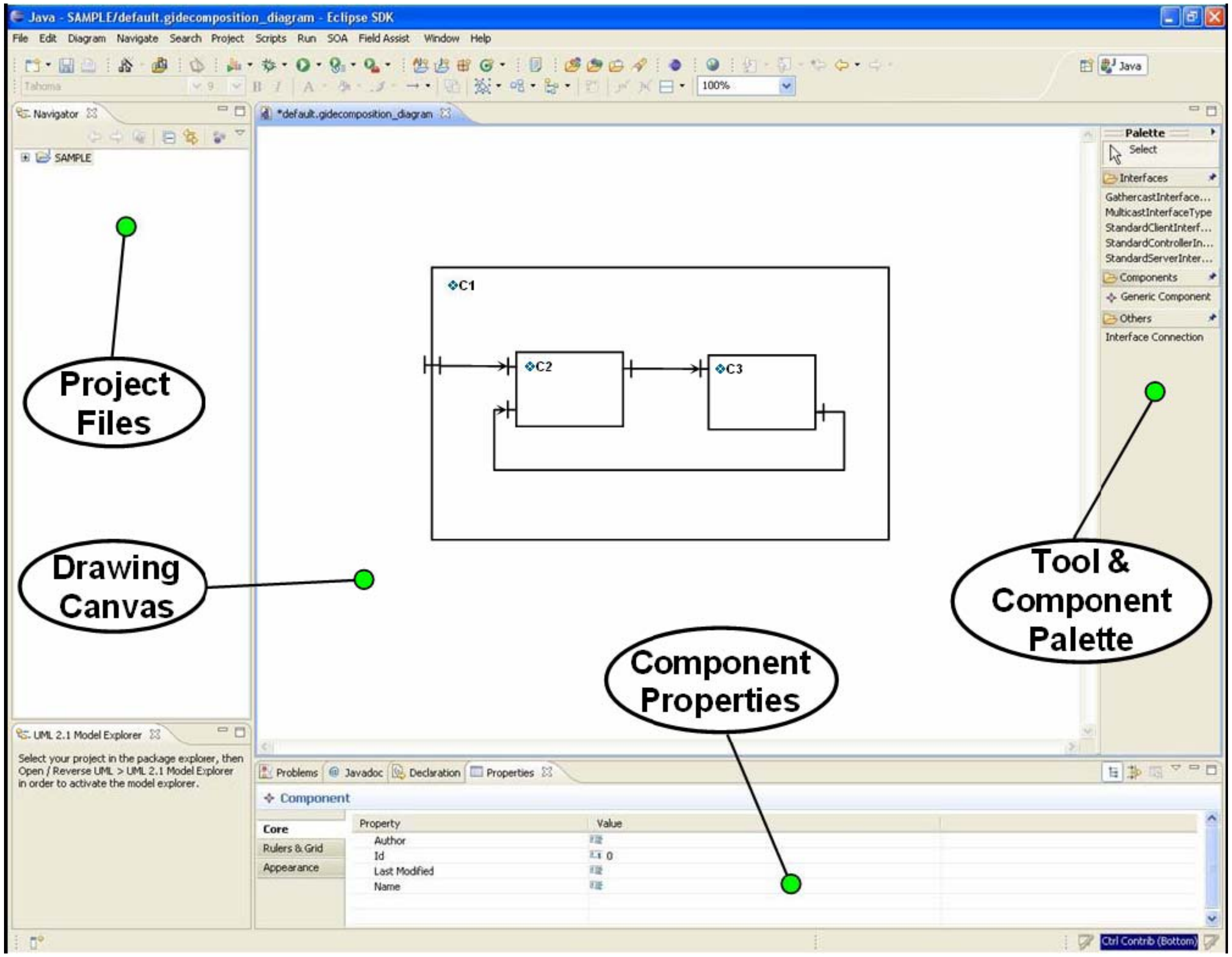
GIDE – An Insight into Composition

- GIDE builds on GMF for providing graphical front-end
- The IDE includes
  - Built-in ADL parser and exporter
  - Verifier
  - Diagram and Semantic generators
  - Component Repository (allow packaging and sharing)
  - Java source code generation
- ADL files are verified, parsed and then appropriate internal representations of compositions (semantic representation) and diagrams are generated.
- GIDE delegates the user-interactions to these internal representations

GIDE – An Insight into Composition



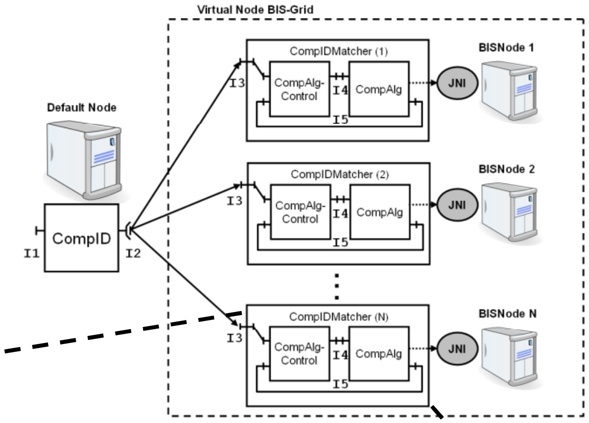
Development – IBM ZRL BIS Use Case



Select your project in the package explorer, then Open / Reverse UML > UML 2.1 Model Explorer in order to activate the model explorer.

Property	Value
Author	
Id	0
Last Modified	
Name	

Case Study: Biometric Identification System (BIS)



ADL Example: the composite component *CompIDMatcher*

```
<definition name="com.ibm.bis.CompIDMatcher">
  <interface name="server" role="server" signature="com.ibm.bis.I3"/>
  <interface name="client" role="client" signature="com.ibm.bis.I3"/>
  <component name="CompAlgControl“ definition="com.ibm.bis.CompAlgControl"/>
  <component name="CompAlg" definition="com.ibm.bis.CompAlg"/>
  <binding client="this.client" server="CompAlgControl.idServer"/>
  <binding client="CompAlgControl.client" server="CompAlg.server"/>
  <binding client="CompAlg.client“ server="CompAlgControl.controlServer"/>
  <controller desc="composite"/>
  <virtual-node name="BIS-Grid" cardinality="single"/>
</definition>
```

Initial Experiences

- **The high level of abstraction provided by the GIDE framework hides the complexity of Grid programming**
- **Behavioural skeletons reduce further the development effort**
- **The strict separation of concerns (ADL definitions, interface definitions, virtual nodes, deployment descriptors) separates the design from the deployment (physical) infrastructure**
- **The hierarchical nature allows fast composition and facilitates reuse**
- **The GIDE, implemented as an Eclipse plug-in using GMF.**

Summary

- Productivity based on higher abstraction
  - Enables the use of new modern technologies such as graphical composition
  - Source code generation
  - Repositories for components re-use
- Optional features
  - Dynamic composition validation using OCL
  - Static composition validation while generating final ADL file(s)
  - Domain-specific validation
  - Dynamic verification
- GIDE prototype packaging and testing

Conclusions

- Created the core framework using Eclipse
- Robust and friendly
- The full prototype of the GIDE toolset has been completed
- The component composition results are promising – development productivity
- GIDE WIKI: http://perun.hscs.wmin.ac.uk/dis/gide/