Model-Driven Development with the jABC

Bernhard Steffen, Ralf Nagel,
Sven Jörges, Christian Kubczak

Chair of Programming Systems
Universität Dortmund

Tiziana Margaria, Clemens Renner

Chair of Service and Software Engineering
Universität Potsdam

jABC Framework

- A modelling tool for service-oriented design
- Predecessors range back to 1995
- Used in several industrial projects
  - Intelligent Networks
  - Web-based distributed decision support systems
  - Test automation environments

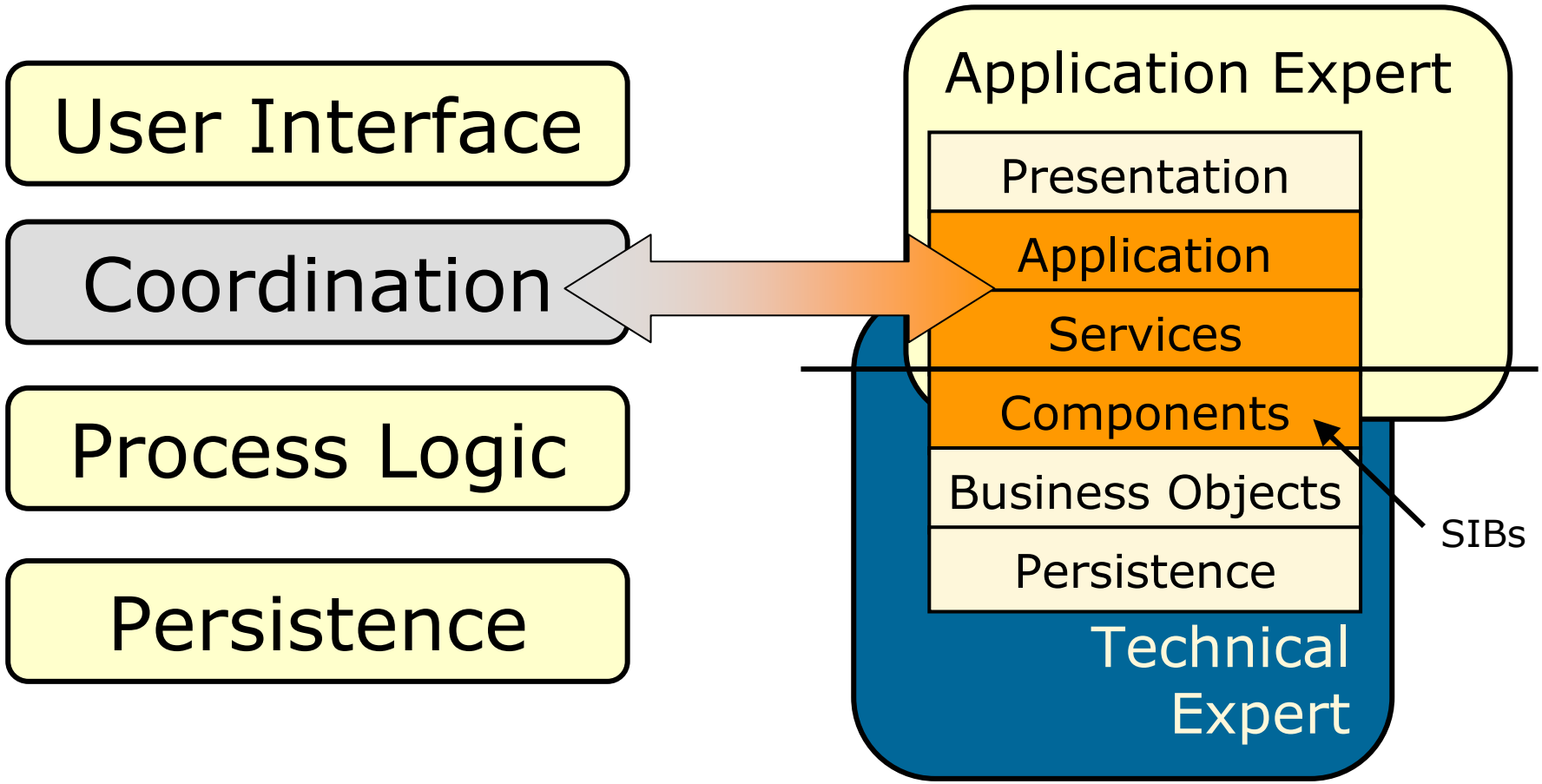
jABC Key Features

- Simplicity
- Agility
- Customisability
- Consistency
- Verification
- Service orientation
- Executability
- Universality

Feature-oriented Description

1. Feature-oriented service description:
   base system and *optional features*
2. Feature and system behaviour:
   specified as *Service Logic Graphs* (SLGs)
3. SLGs are realized using a library of reusable
   Service-Independent Building Blocks (SIBs)
4. Description also includes abstract requirements
5. Temporal logic *constraints* regulate *interactions*
   between features
6. Feature compositions are allowed if no
   constraints are violated

jABC Architecture



Service-Independent Building Blocks (SIBs)

- Are nodes in the Service Logic Graphs
- Outgoing branches are chosen according to the result of execution
- Placeholders for future functionality can be automatically generated
- Self-documentation options
- Hierarchy via *GraphSIBs*
- May be as simple as making a library call
- May even alter the control flow

The structure of SIBs (1)

- SIBs are self-contained Java classes
- Name is the Java class name
- Parameters are the public fields
- Branches are reserved public fields

```
1  import de.metaframe.jabc.framework.sib.annotation.SIBClass;
2
3  @SIBClass("20861480163021731e729e8f:c8fed88d54c0:6d1c73a9d71047059ea61b")
4  public class Example {
5
6      public int counter = 0;
7      public int value = 1;
8
9      public static final String[] BRANCHES = { "fix1", "fix2" };
10     public String[] branches = { "var1", "var2" };
11
12     public Example() {
13         // SIB Constructor
14     }
15
16 }
```

The structure of SIBs (2)

- SIBs may provide documentation (and an icon)

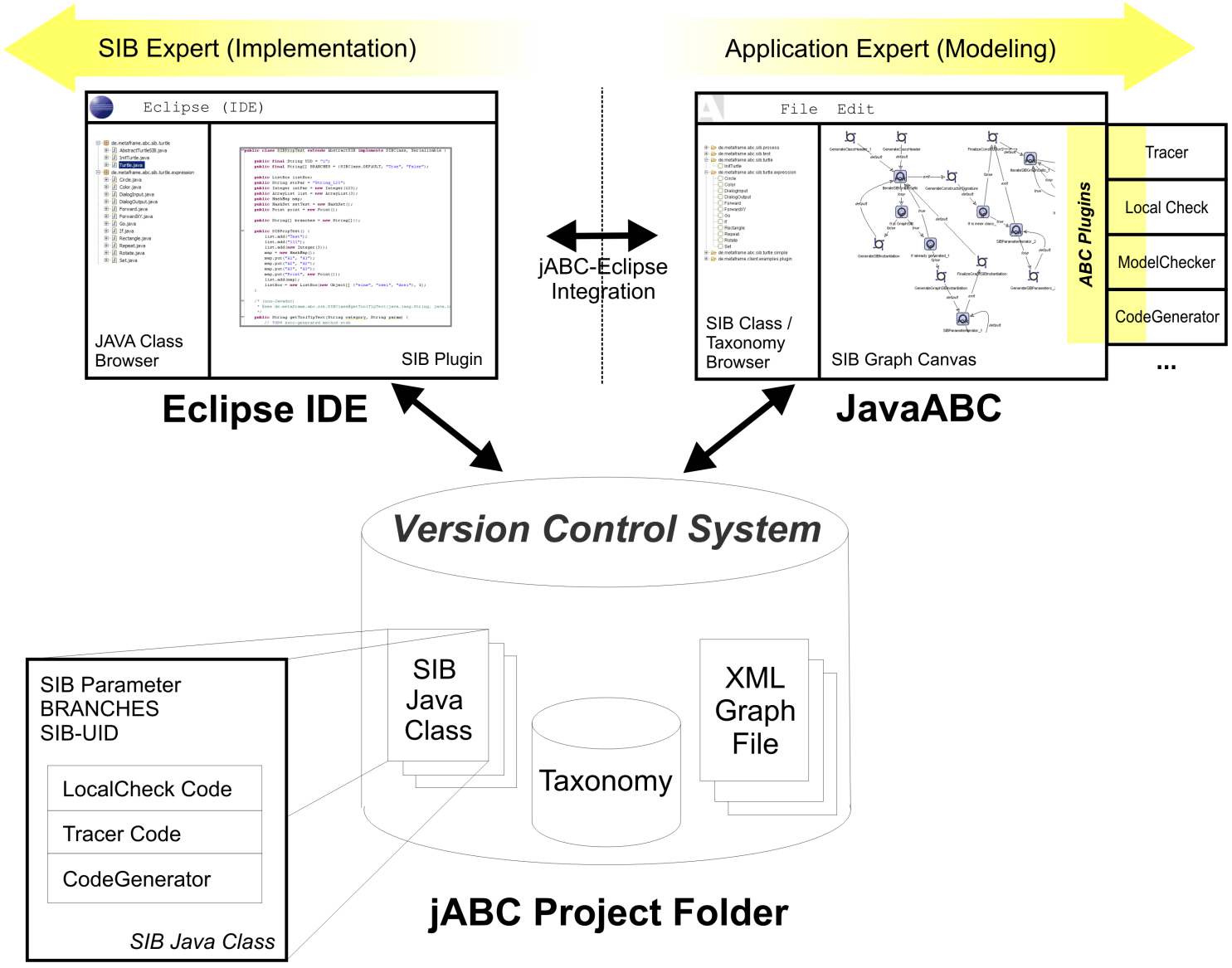
```
1  import de.metaframe.jabc.framework.sib.DocType;
2  import de.metaframe.jabc.framework.sib.annotation.SIBClass;
3
4  @SIBClass("20861480163021731e729e8f:c8fed88d54c0:6d1c73a9d71047059ea61b")
5  public class Example {
6
7      public int counter = 0;
8      public int value = 1;
9
10     public static final String[] BRANCHES = { "fix1", "fix2" };
11     public String[] branches = { "var1", "var2" };
12
13     public Example() {
14         // SIB Constructor
15     }
16
17     public String getDocumentation(DocType category, String param) {
18         if(category.equals(DocType.SIB))
19             return "A demo SIB.";
20         else
21             return null;
22     }
23
24 }
```

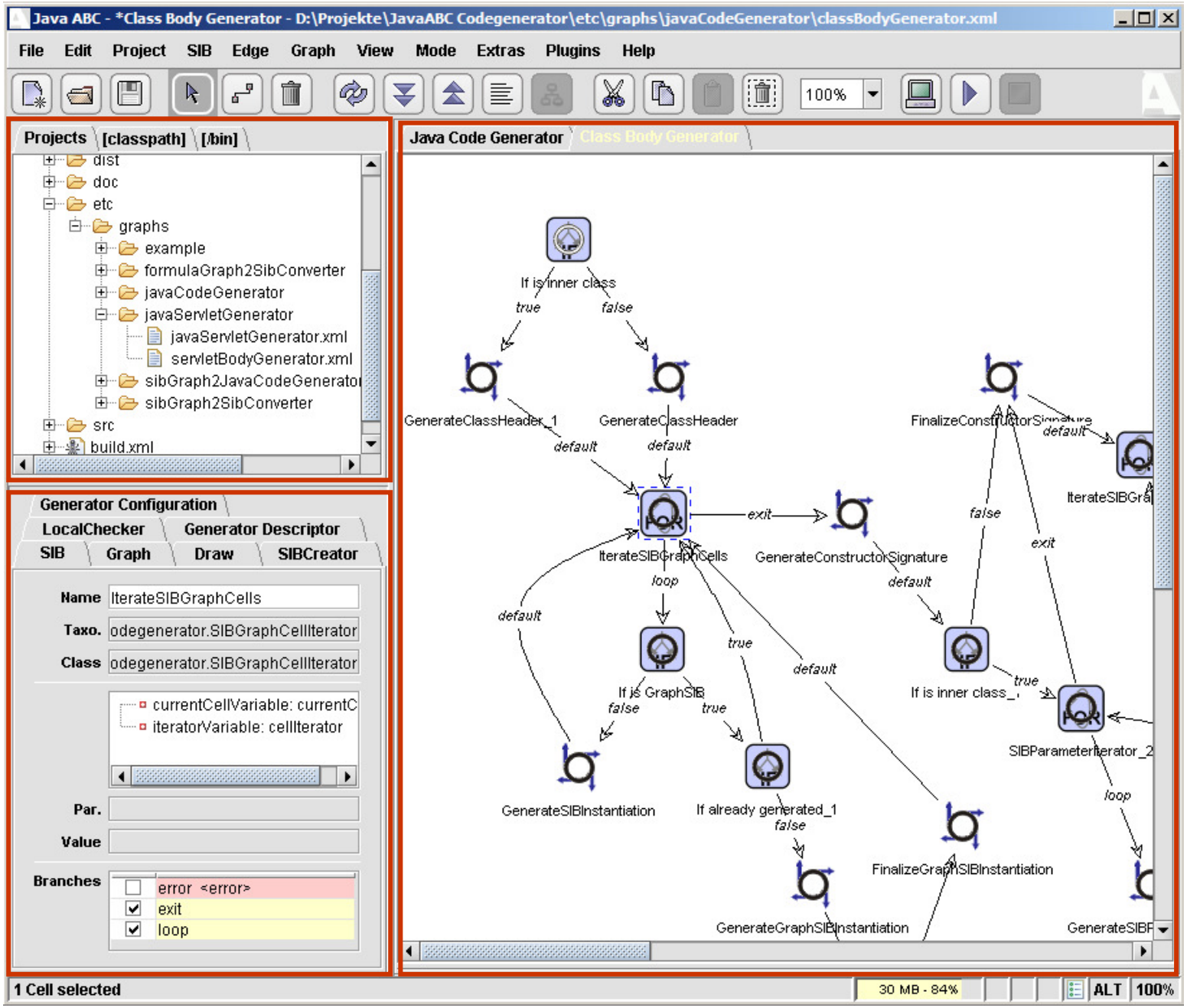
```java
import de.metaframe.abc.sib.Executable;
import de.metaframe.jabc.framework.execution.ExecutionEnvironment;
import de.metaframe.jabc.framework.sib.DocType;
import de.metaframe.jabc.framework.sib.annotation.SIBClass;

@SIBClass("208614801630217316729e8f:c8fed88d54c0:6d1c73a9d71047059ea61b")
public class Example implements Executable {

    public int counter = 0;
    public int value = 1;

    public static final String[] BRANCHES = { "fix1", "fix2" };
    public String[] branches = { "var1", "var2" };

    public Example() {
        // SIB Constructor
    }

    public String getDocumentation(DocType category, String param) {
        if(category.equals(DocType.SIB))
            return "A demo SIB.";
        else
            return null;
    }

    public String trace(ExecutionEnvironment env) {
        if (env.existsKey("branch"))
            return env.get("branch").toString();
        else
            return "fix1";
    }
}
```

SIB Taxonomies

- Users can specify their own taxonomies on the available SIB repository
- Dynamic arrangement into domain-specific SIB libraries
- No need to stick with the taxonomy given by the packages of the SIB Java classes

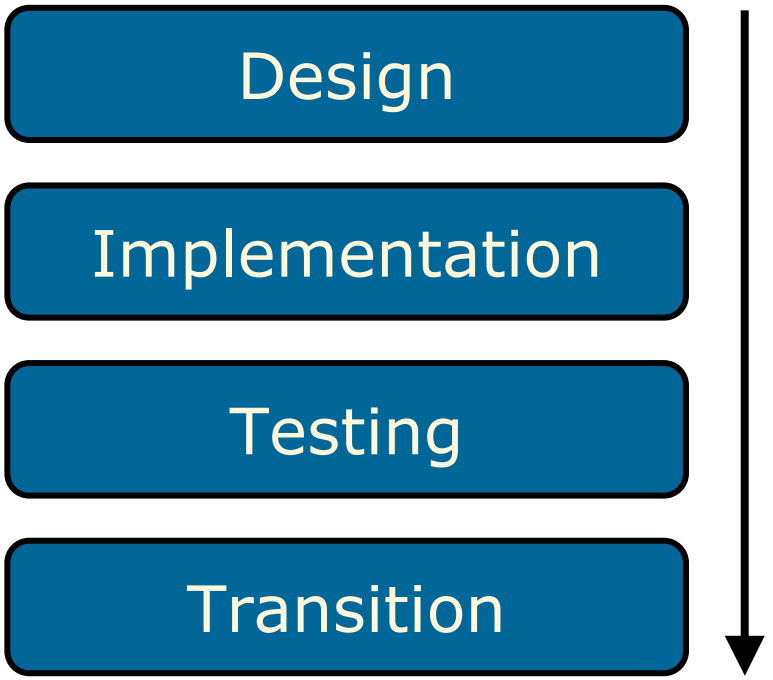
The Big Picture




Java ABC - *Class Body Generator - D:\Projekte\JavaABC Codegenerator\etc\graphs\javaCodeGenerator\classBodyGenerator.xml
File Edit Project SIB Edge Graph View Mode Extras Plugins Help
100%
Projects [classpath] [/bin]
dist
doc
etc
graphs
example
formulaGraph2SibConverter
javaCodeGenerator
javaServletGenerator
javaServletGenerator.xml
servletBodyGenerator.xml
sibGraph2JavaCodeGenerator
sibGraph2SibConverter
src
build.xml
Generator Configuration
LocalChecker Generator Descriptor
SIB Graph Draw SIBCreator
Name IterateSIBGraphCells
Taxo. odegenerator.SIBGraphCellIterator
Class odegenerator.SIBGraphCellIterator
currentCellVariable: currentC
iteratorVariable: cellIterator
Par.
Value
Branches
error <error>
exit
loop
Java Code Generator
Class Body Generator
If is inner class
true
false
GenerateClassHeader_1
GenerateClassHeader
default
IterateSIBGraphCells
exit
GenerateConstructorSignature
loop
If is GraphSIB
GenerateSIBInstantiation
If already generated_1
GenerateGraphSIBInstantiation
FinalizeGraphSIBInstantiation
If is inner class_1
SIBParameterIterator_2
1 Cell selected
30 MB - 84%
ALT
100%

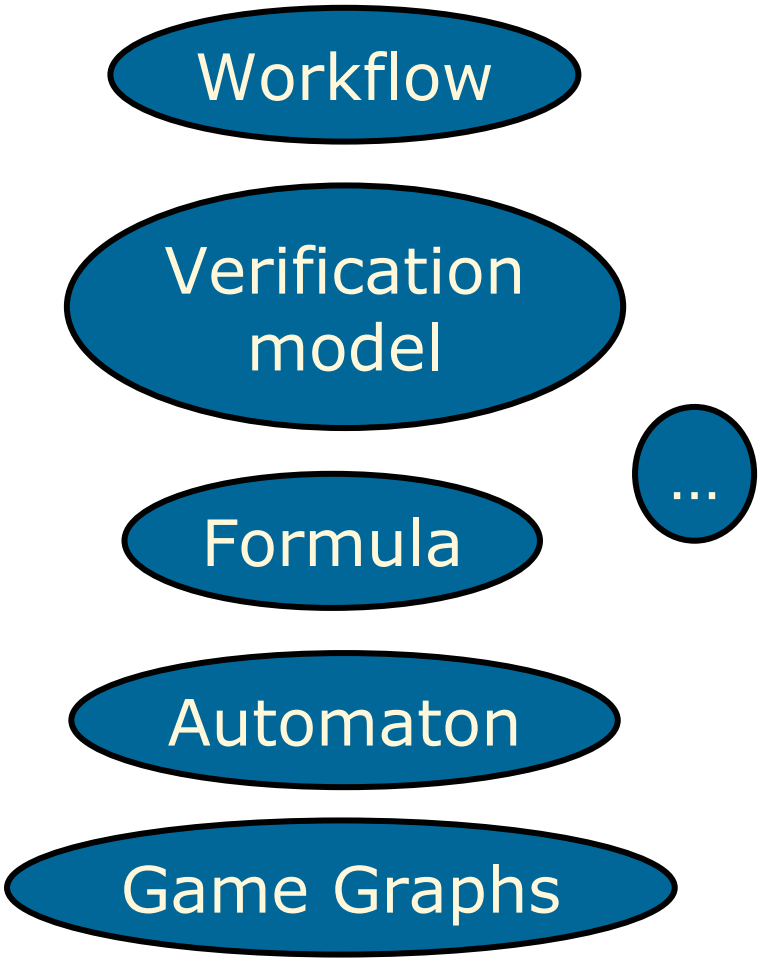
Model-Based Lifecycle Management (1)

- One tool for the *entire* development process

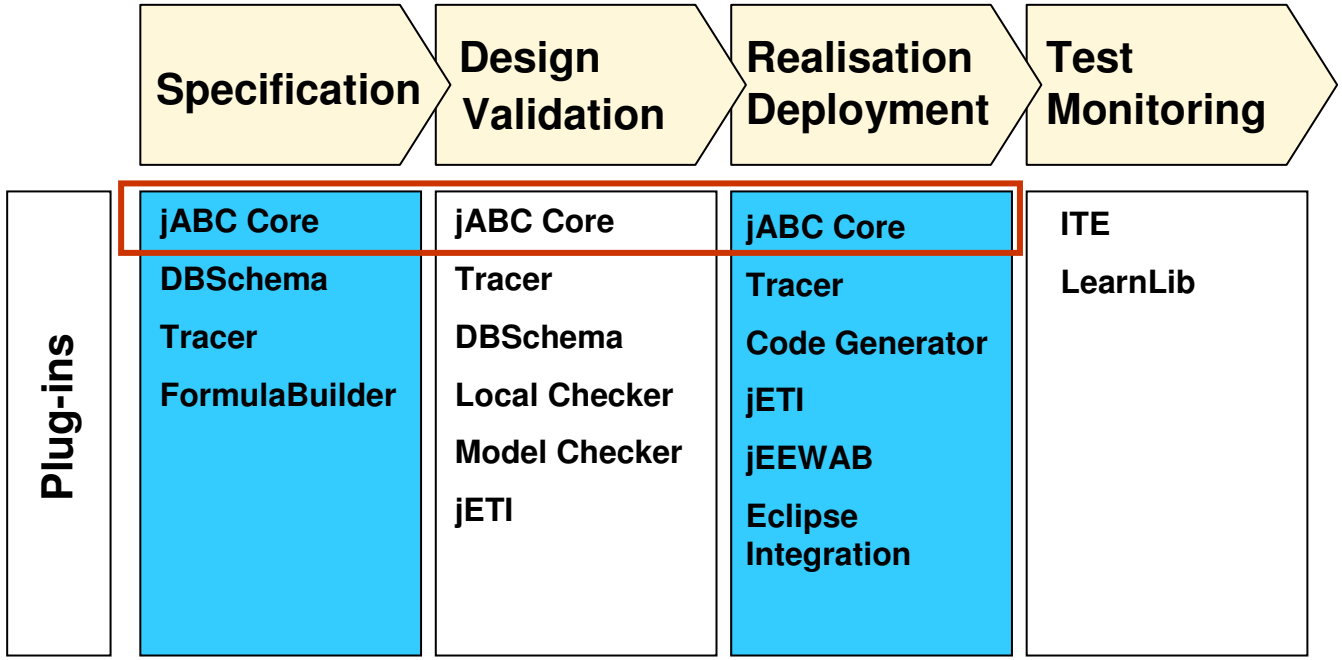


Model-Based Lifecycle Management (2)

- One tool for several interpretations of the same model via plug-ins



Model-Based Lifecycle Management (3)



Enriching the JavaABC functionality using plug-ins

- Flexible plug-in framework
- Example plug-ins include
  - Local and global model verification
  - Automaton generation
  - Visual modelling of formulas
  - Step-wise execution and debugging
  - Code generation
  - Database schema design

Plug-ins overview (1)

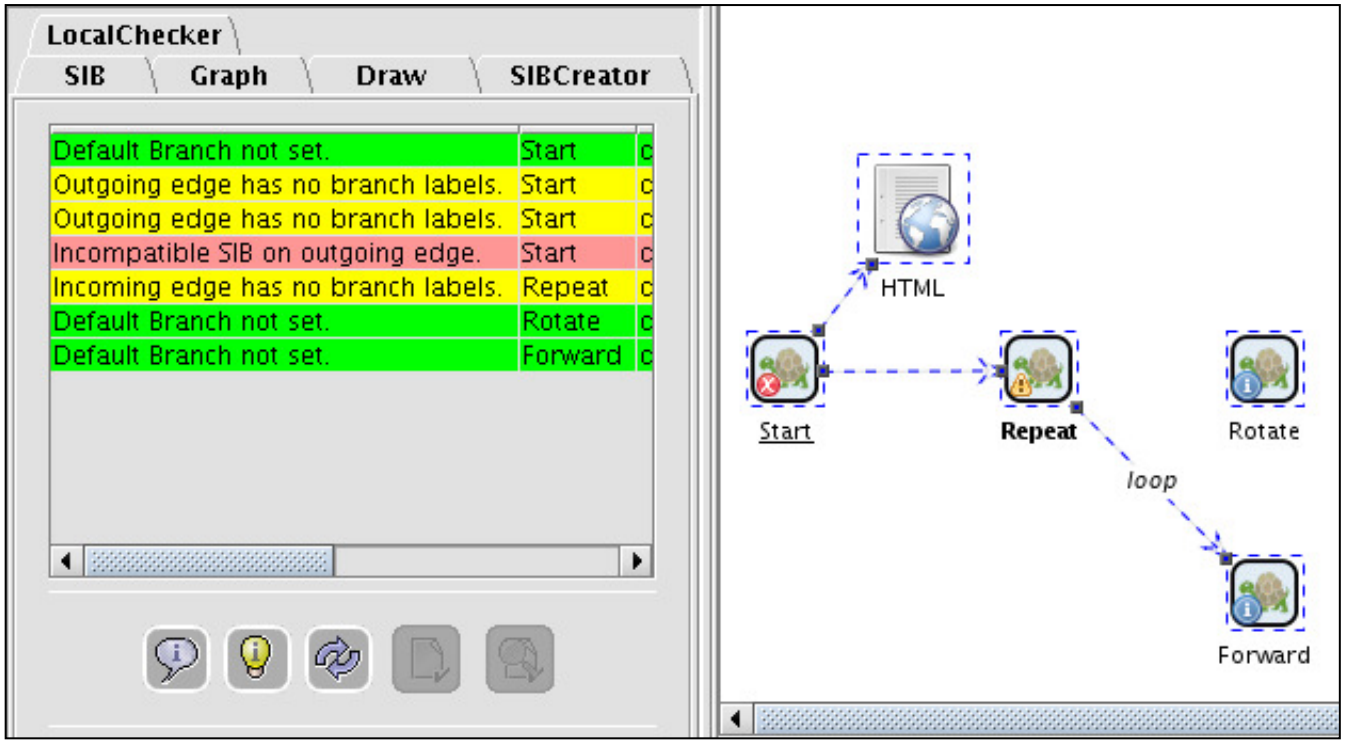
Plug-in	local	global	internal	interface
Bean Shell	X			X
Code Generator		X		X
FormulaBuilder		X		X
DBSchema		X		X
Docbook	X	X		X
Eclipse	X	X		X
GEAR		X	X	
jETI	X			X

Plug-ins overview (2)

Plug-in	local	global	internal	interface
JEEWAB		X		X
jMoSeL		X		X
LearnLib		X		X
LocalChecker	X		X	
SIBCreator	X			X
Taxonomy Editor	X		X	
Tracer	X	X	X	

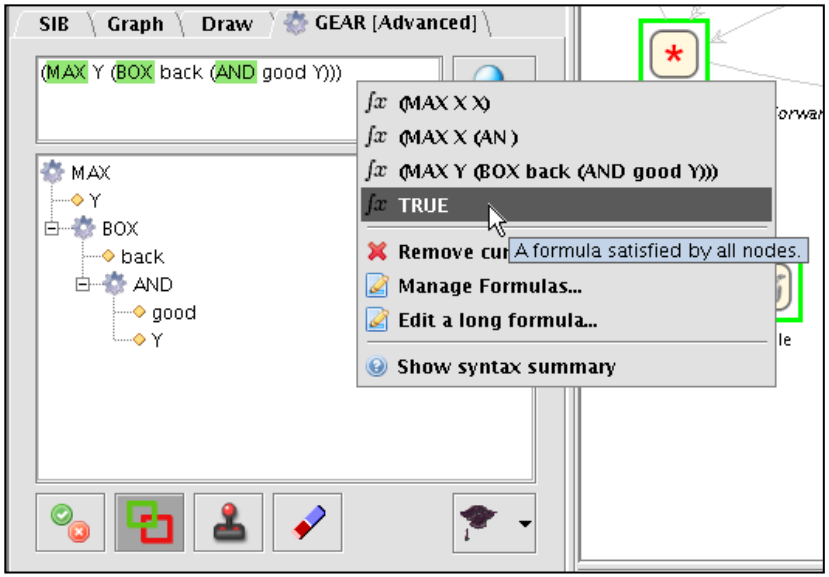
Local model verification

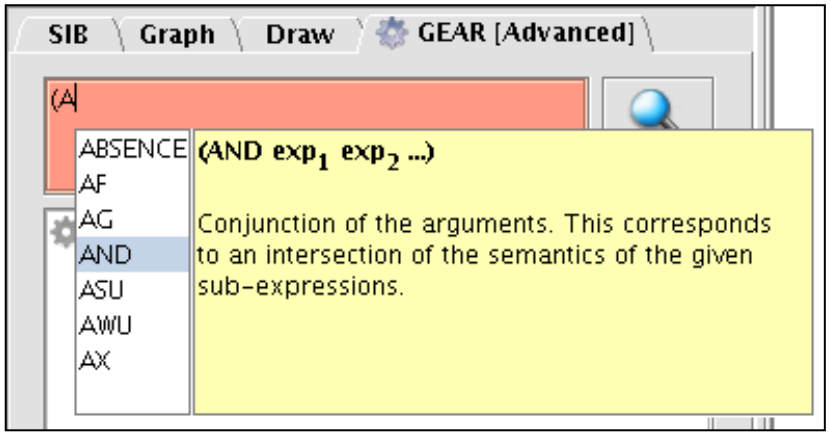
- **LocalChecker**
  run-time checking of SIB usage
- Java code written by the SIB expert



Global model verification

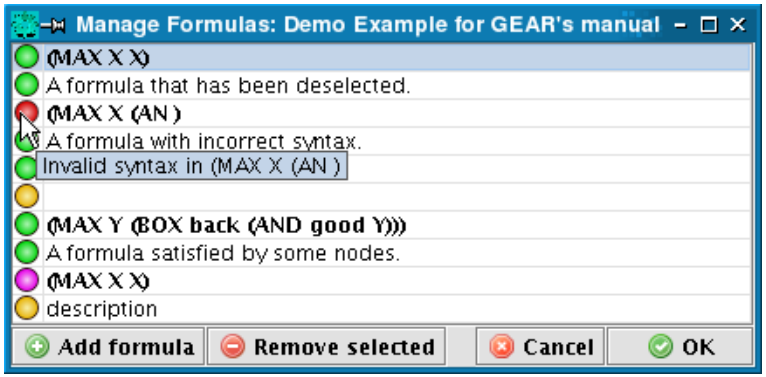
- **GEAR**
  Game-based model checker for the modal μ-calculus
- Rich macro mechanism to extend syntax and semantics
- Several views for different contexts

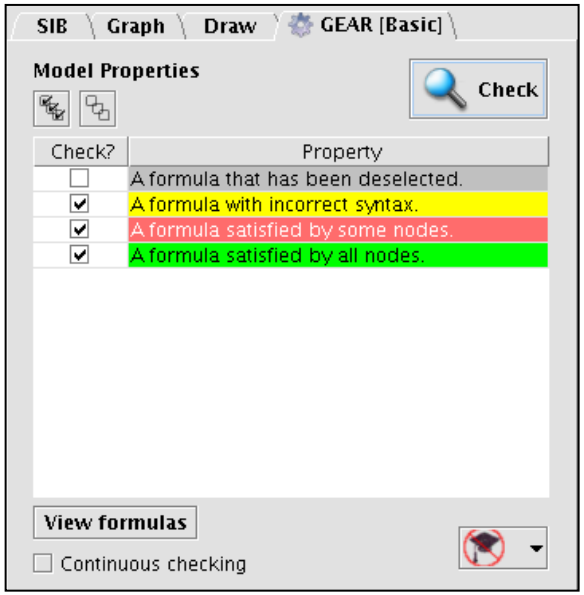




Global model verification

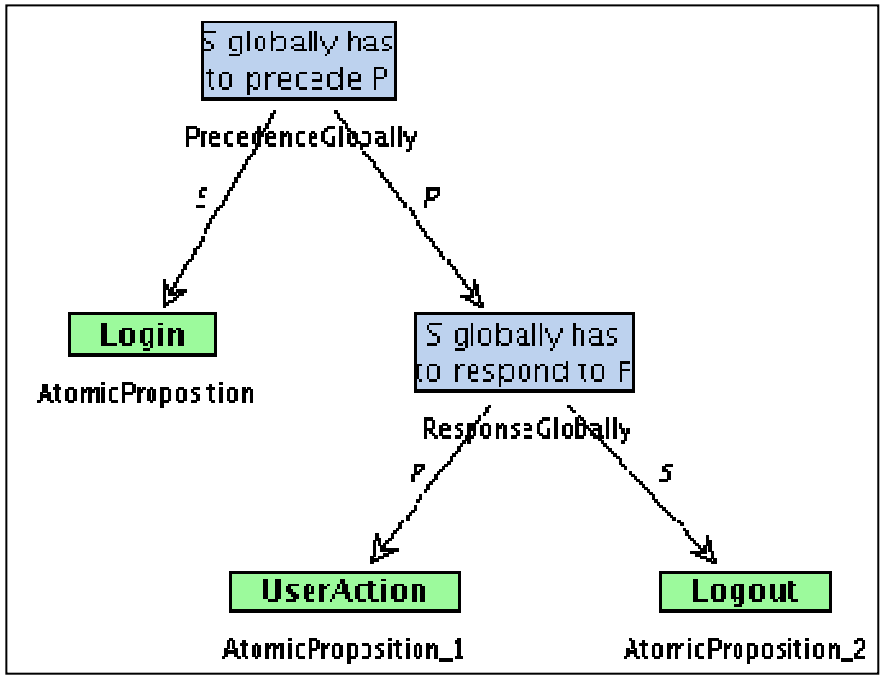
- **GEAR**
  Game-based model checker for the modal μ-calculus
- Rich macro mechanism to extend syntax and semantics
- Several views for different contexts





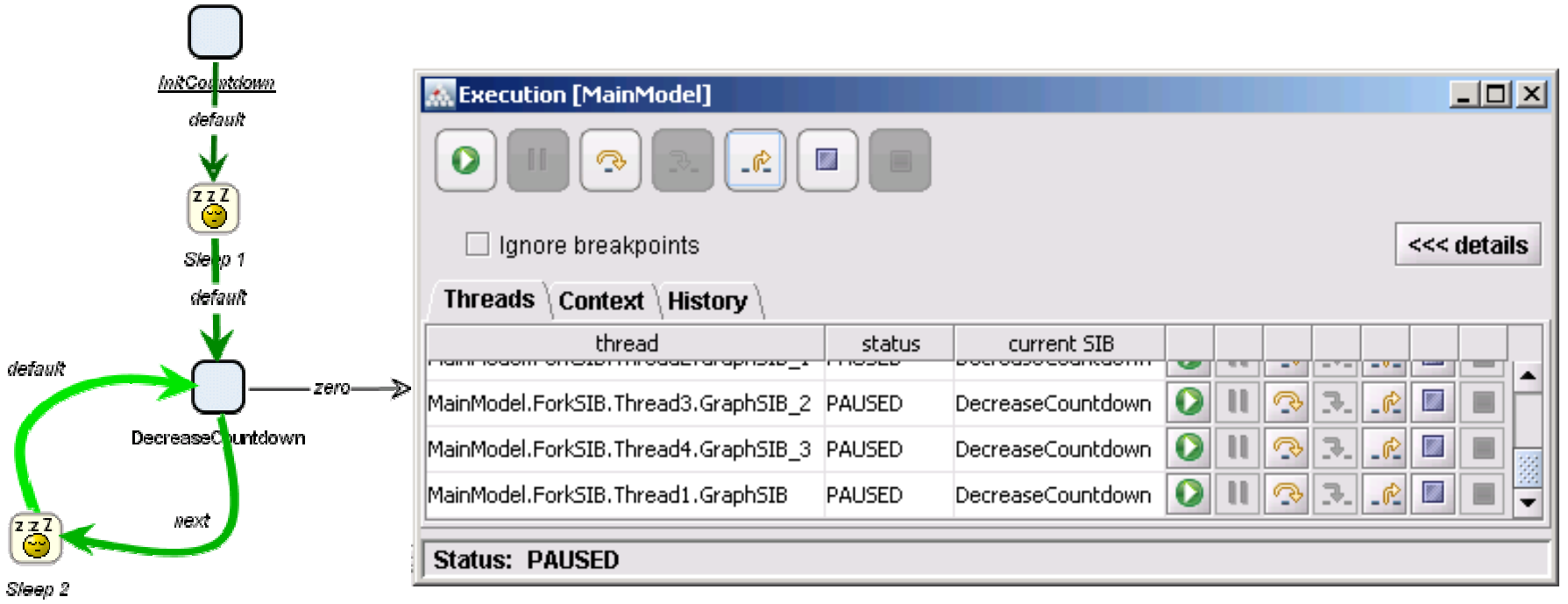
Visual modelling and generation of formulas

- **FormulaBuilder**
  Formula modelling and generation
- Export to pre-defined logic syntaxes, e.g. GEAR



Step-by-step execution and debugging

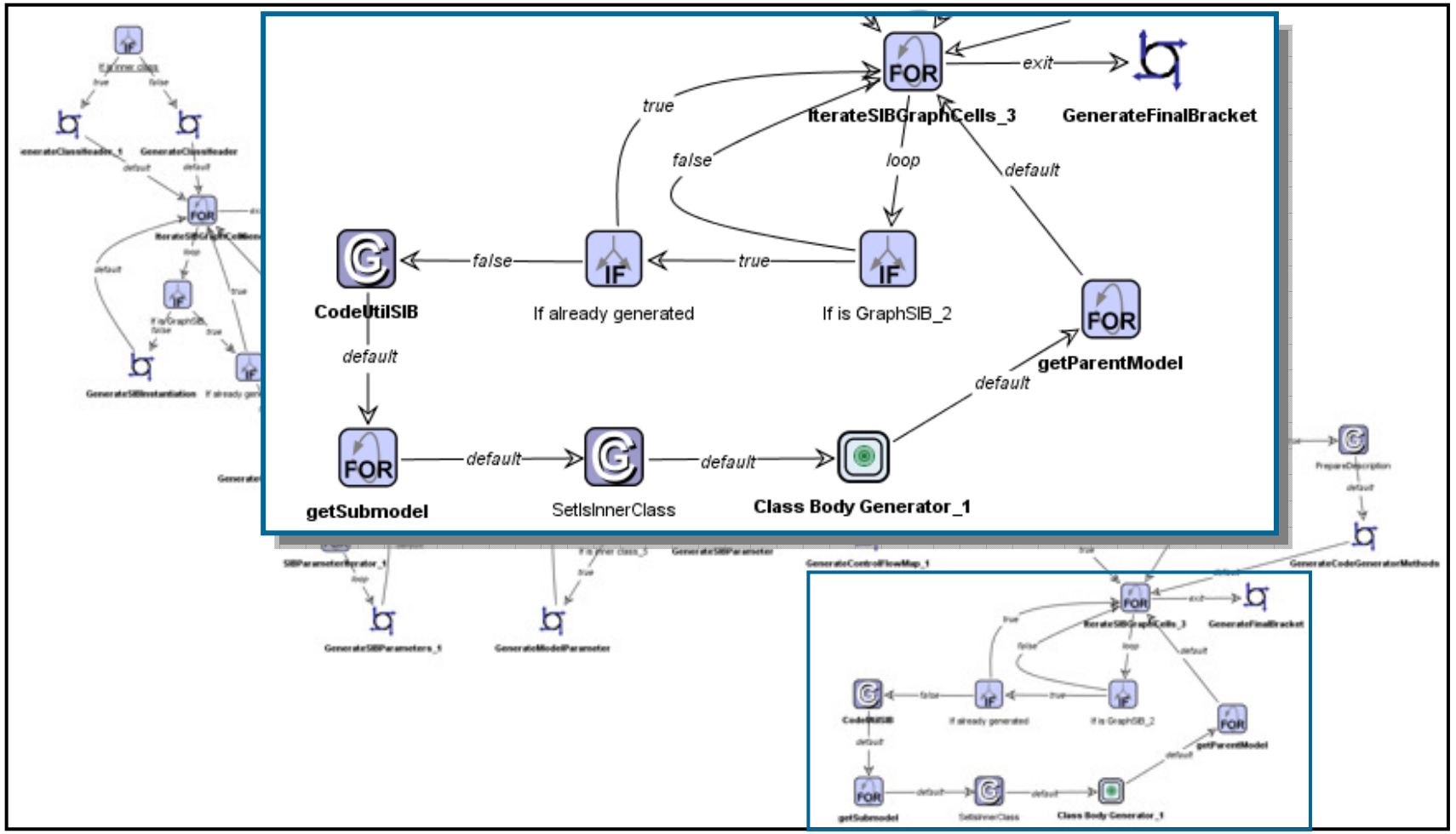
- **Tracer**
  step-wise execution of SIB-Graphs
- Enables debugging and inspection of contexts
- Fork/Join parallelism with threads



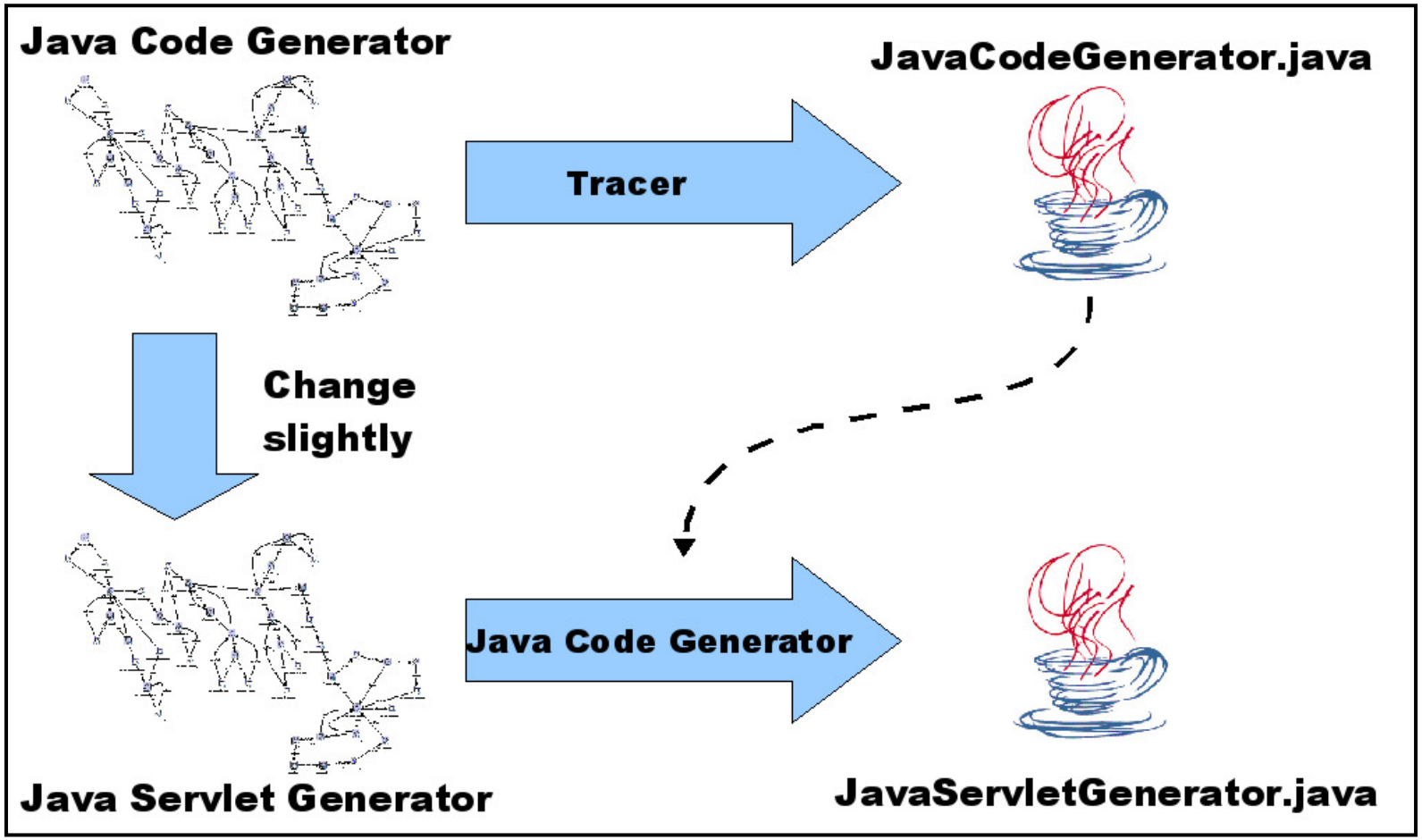
Generating code from SLGs

- **Code Generator**
  Generates code from SLGs
- Modelled using jABC
- Bootstrapped via the Tracer plug-in
- Builds Java classes or Java servlets
- C++ or J2ME code generation in progress
- LEGO Mindstorms RCX

Generating code from SLGs



Generating code from SLGs



Conclusion

- jABC as an LPC-based development method
- Improved validation technology
- Application domains cover
  - Supply chain management (IKEA)
  - Modelling and Execution of Bio-Informatics workflows
  - Semantic Web Service Challenge
  - Data-Flow Analysis of Java programs
  - Management framework for remote intelligent configuration of systems

Thank you for your time!

- jABC Framework
  http://www.jabc.de
- Chair of Programming Systems
  Universität Dortmund
  http://www.ls-5.de
- Chair of Software and Service Engineering
  Universität Potsdam
  http://www.cs.uni-potsdam.de/sse