Model-Driven Design using I-Logix Rhapsody

Riandi Wiguna
MSDL
August 5, 2005

Overview

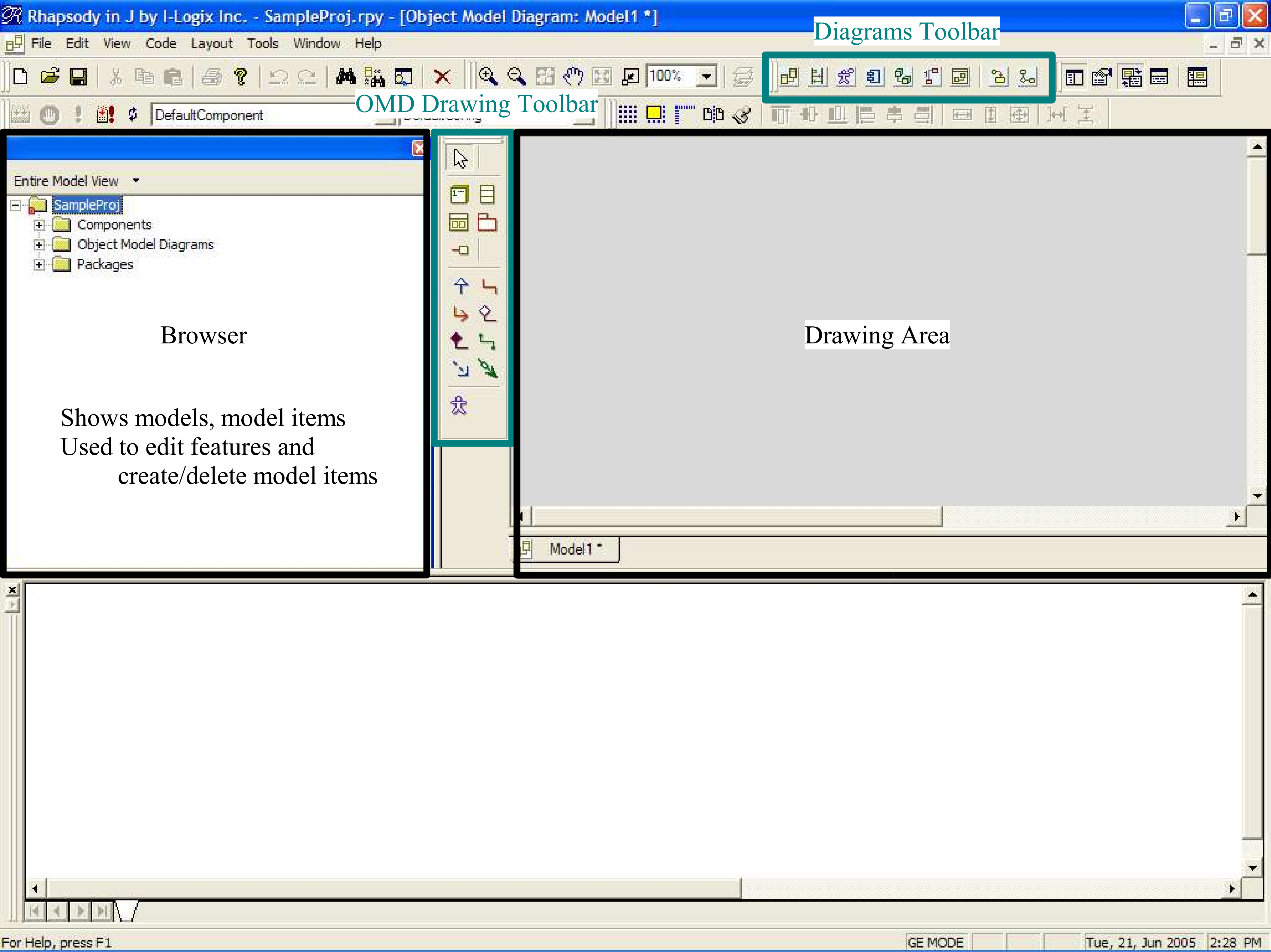
1. Introduction to Rhapsody
2. Basic Usage of Rhapsody
3. Important Points
4. ROPES
5. Example: Answering Machine
6. Personal Experiences using Rhapsody
7. Conclusion

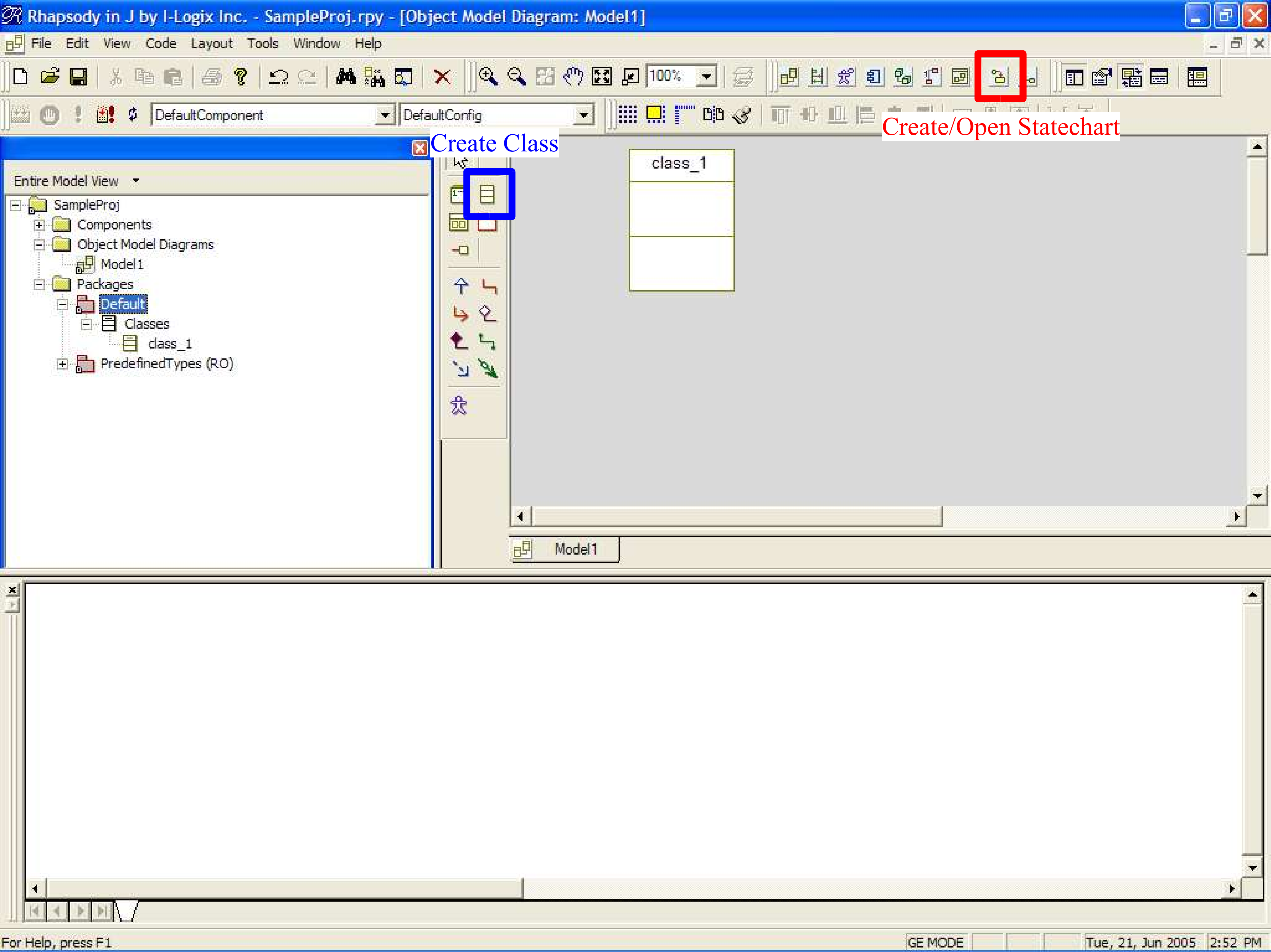
Introduction to Rhapsody

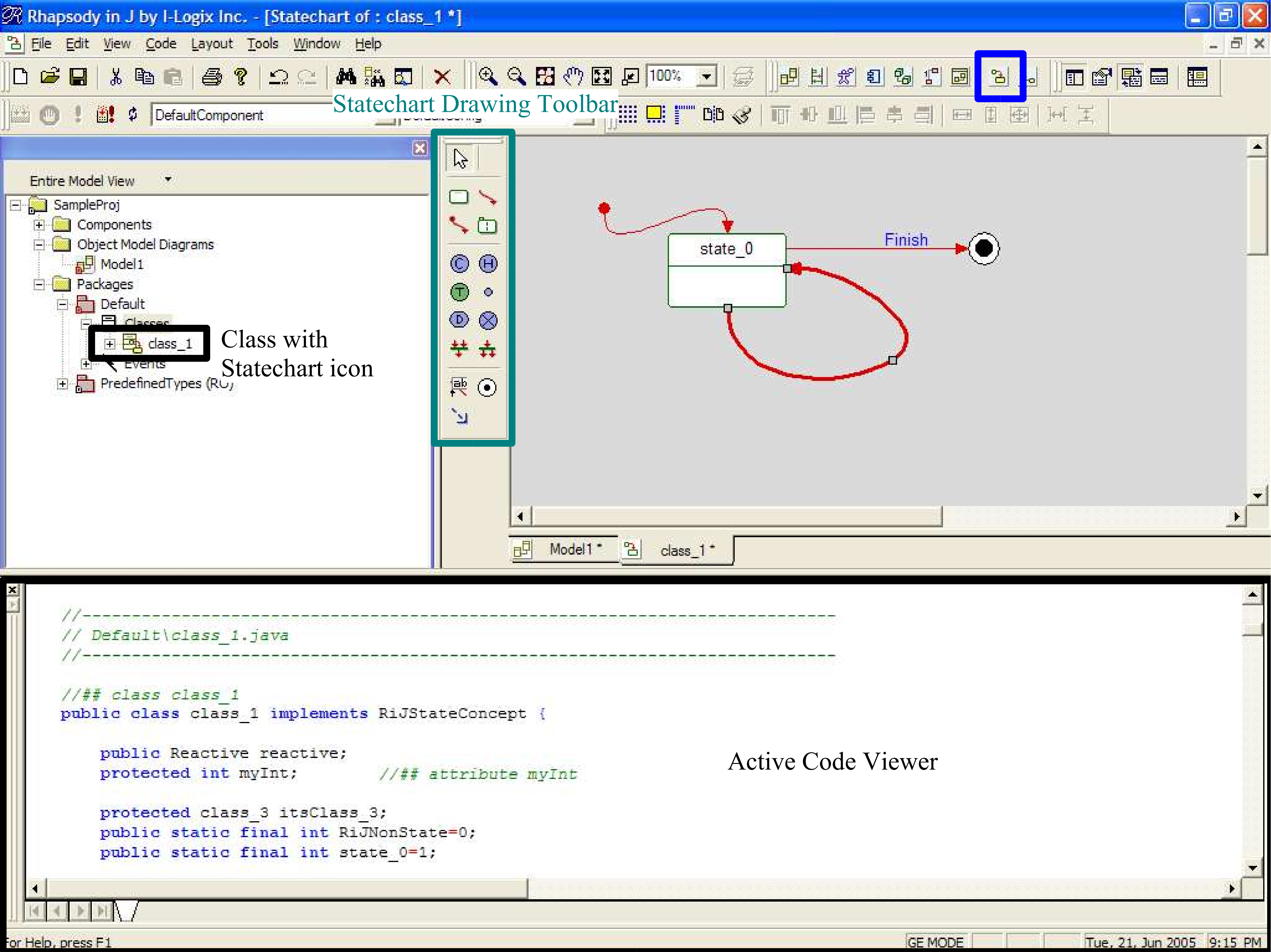
- Rhapsody is software for UML-based design and simulation
  - Activity Diagrams
  - Collaboration Diagrams
  - Component Diagrams
  - Deployment Diagrams
  - Sequence Diagrams
  - Statecharts
  - Structure Diagrams
  - Object Model Diagrams
  - Use Case Diagrams

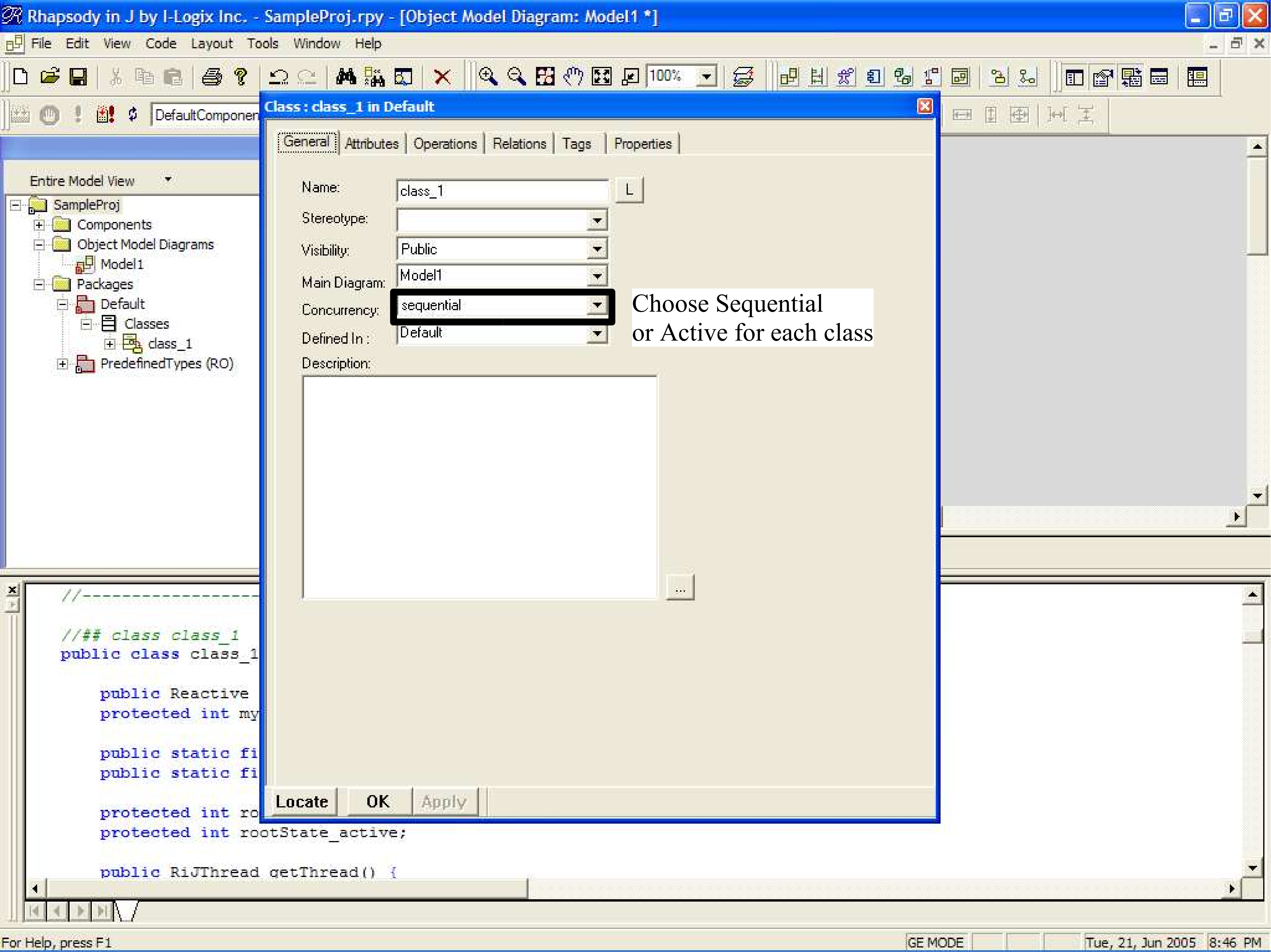
Introduction to Rhapsody

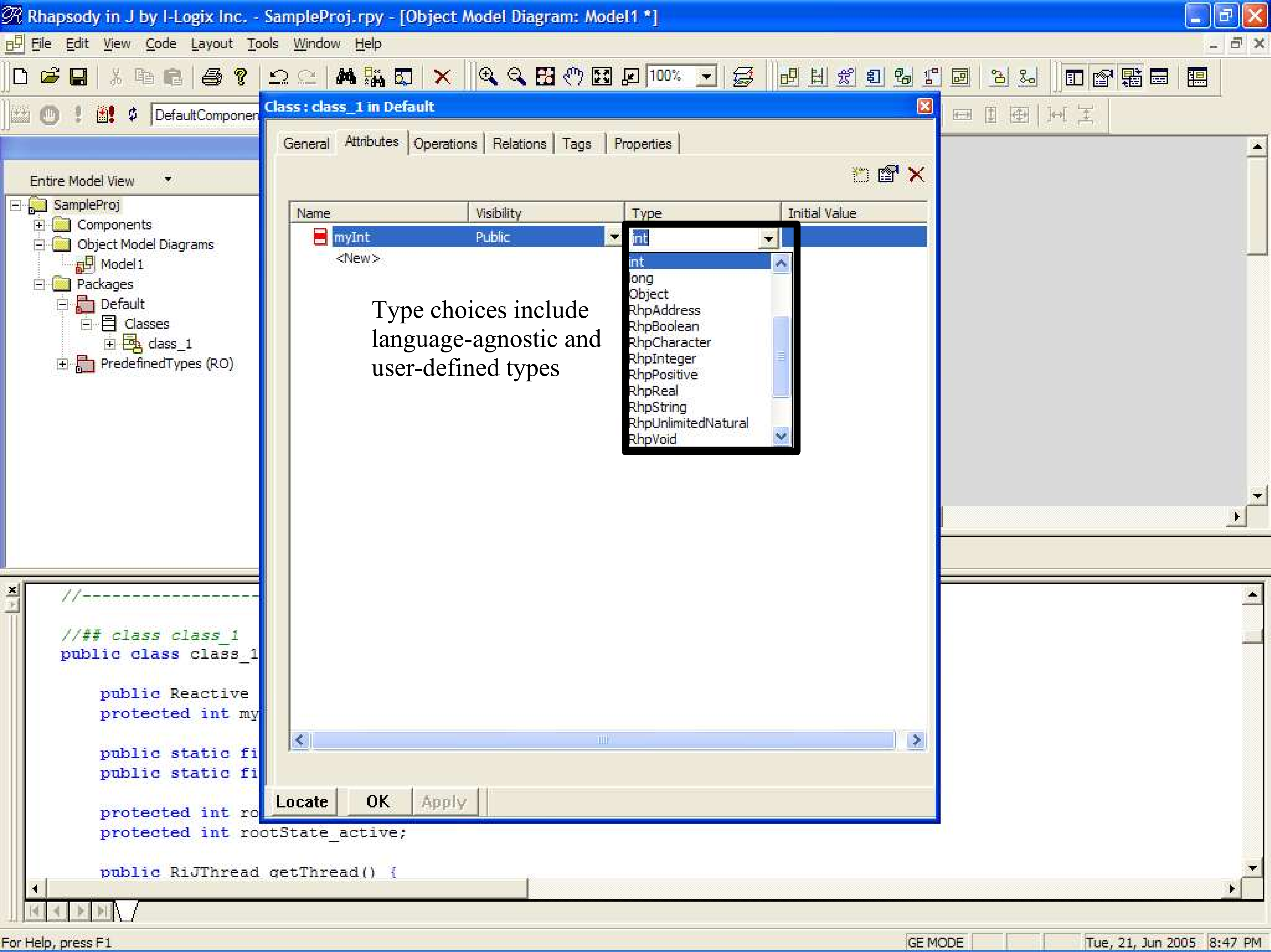
- Generates C, C++, Ada, or Java code
- Allows for reactivity, multiple threads, real-time environments
- Allows user to “roundtrip”, i.e. alter code directly and then update visual model
- For this presentation, “Rhapsody in J” version 6.0 for Windows used (“Rhapsody in C++” examined)

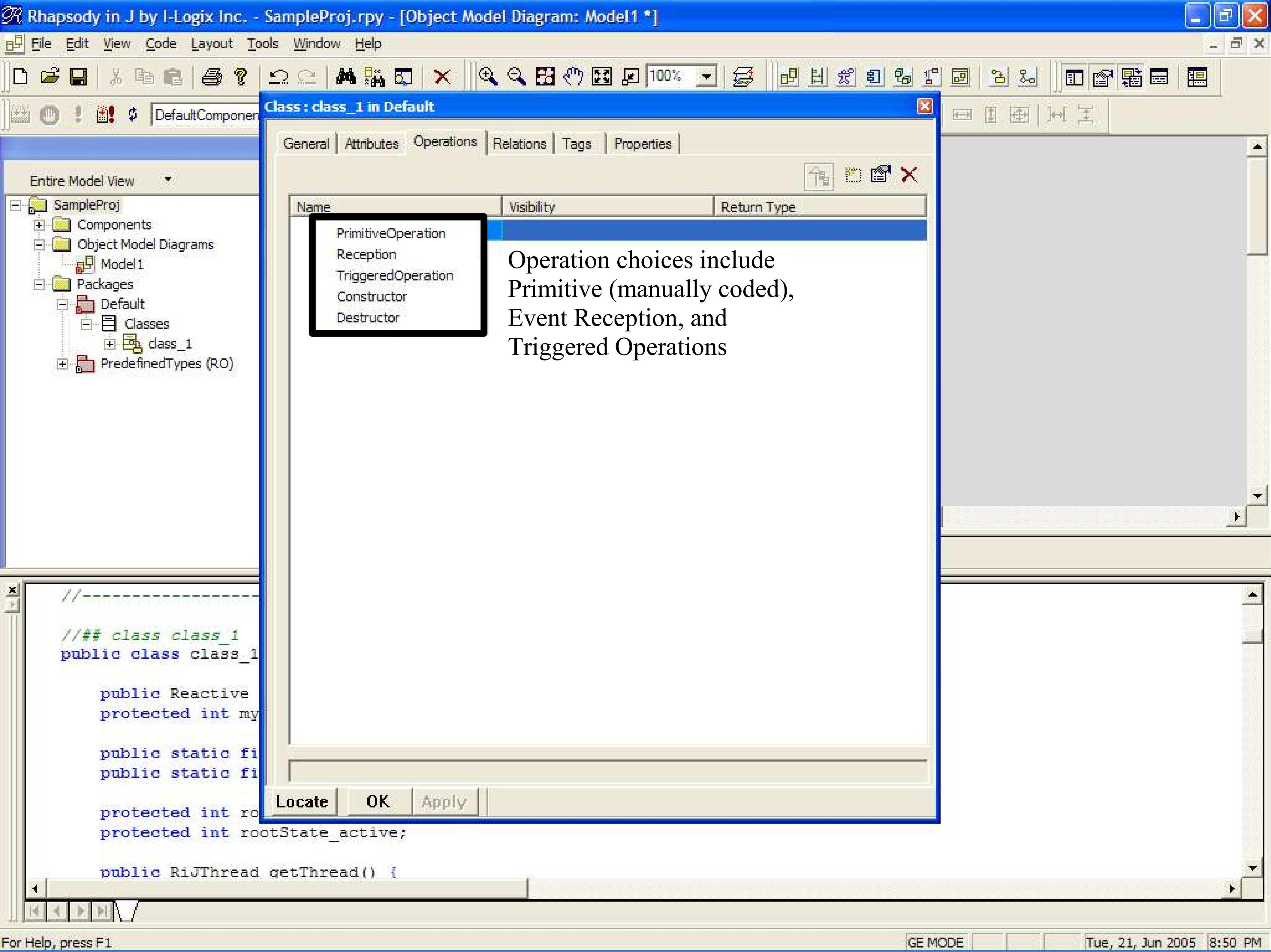

Rhapsody in J by I-Logix Inc. - SampleProj.rpy - [Object Model Diagram: Model1 *]
File Edit View Code Layout Tools Window Help
Diagrams Toolbar
100%
OMD Drawing Toolbar
DefaultComponent
Entire Model View
SampleProj
Components
Object Model Diagrams
Packages
Browser
Shows models, model items
Used to edit features and
create/delete model items
Drawing Area
Model1 *
For Help, press F1
GE MODE
Tue, 21, Jun 2005
2:28 PM


Rhapsody in J by I-Logix Inc. - SampleProj.rpy - [Object Model Diagram: Model1]
File Edit View Code Layout Tools Window Help
100%
DefaultComponent
DefaultConfig
Create Class
Create/Open Statechart
Entire Model View
SampleProj
Components
Object Model Diagrams
Model1
Packages
Default
Classes
class_1
PredefinedTypes (RO)
class_1
Model1
For Help, press F1
GE MODE
Tue, 21, Jun 2005
2:52 PM


Rhapsody in J by I-Logix Inc. - [Statechart of : class_1 *]
File Edit View Code Layout Tools Window Help
DefaultComponent
Statechart Drawing Toolbar
Entire Model View
SampleProj
Components
Object Model Diagrams
Model1
Packages
Default
class_1
PredefinedTypes
Class with Statechart icon
state_0
Finish
Model1 *
class_1 *
//--
// Default\class_1.java
//--
//## class class_1
public class class_1 implements RiJStateConcept {
public Reactive reactive;
protected int myInt; //## attribute myInt
protected class_3 itsClass_3;
public static final int RiJNonState=0;
public static final int state_0=1;
Active Code Viewer
For Help, press F1
Tue, 21, Jun 2005
9:15 PM

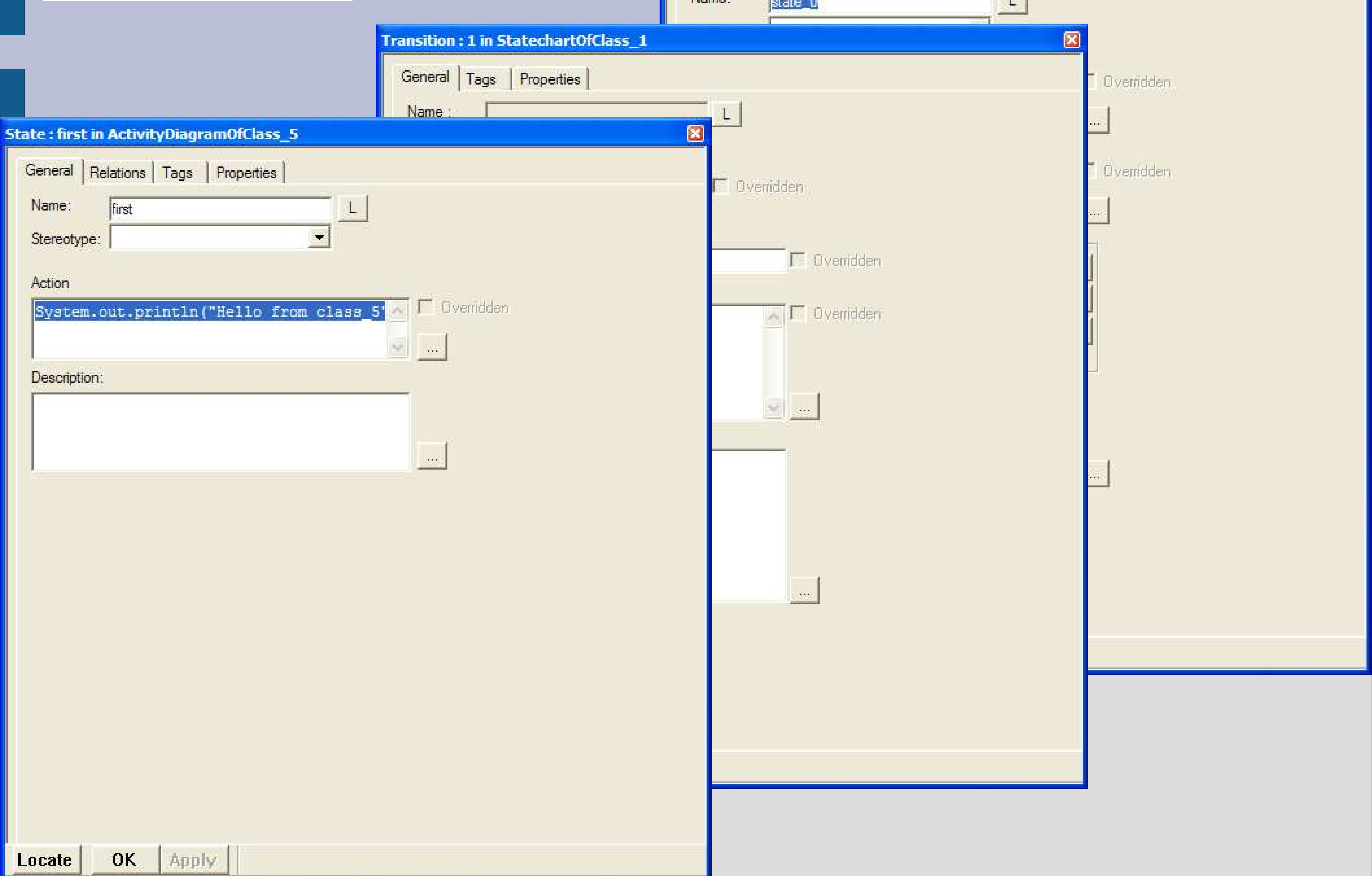

Rhapsody in J by I-Logix Inc. - SampleProj.rpy - [Object Model Diagram: Model1 *]
File Edit View Code Layout Tools Window Help
100%
DefaultComponen
Entire Model View
SampleProj
Components
Object Model Diagrams
Model1
Packages
Default
Classes
class_1
PredefinedTypes (RO)
Class : class_1 in Default
General
Attributes
Operations
Relations
Tags
Properties
Name:
class_1
L
Stereotype:
Visibility:
Public
Main Diagram:
Model1
Concurrency:
sequential
Defined In :
Default
Description:
...
Locate
OK
Apply
Choose Sequential
or Active for each class
//## class class_1
public class class_1
public Reactive
protected int my
public static fi
public static fi
protected int ro
protected int rootState_active;
public RiJThread getThread() {
For Help, press F1
GE MODE
Tue, 21, Jun 2005
8:46 PM

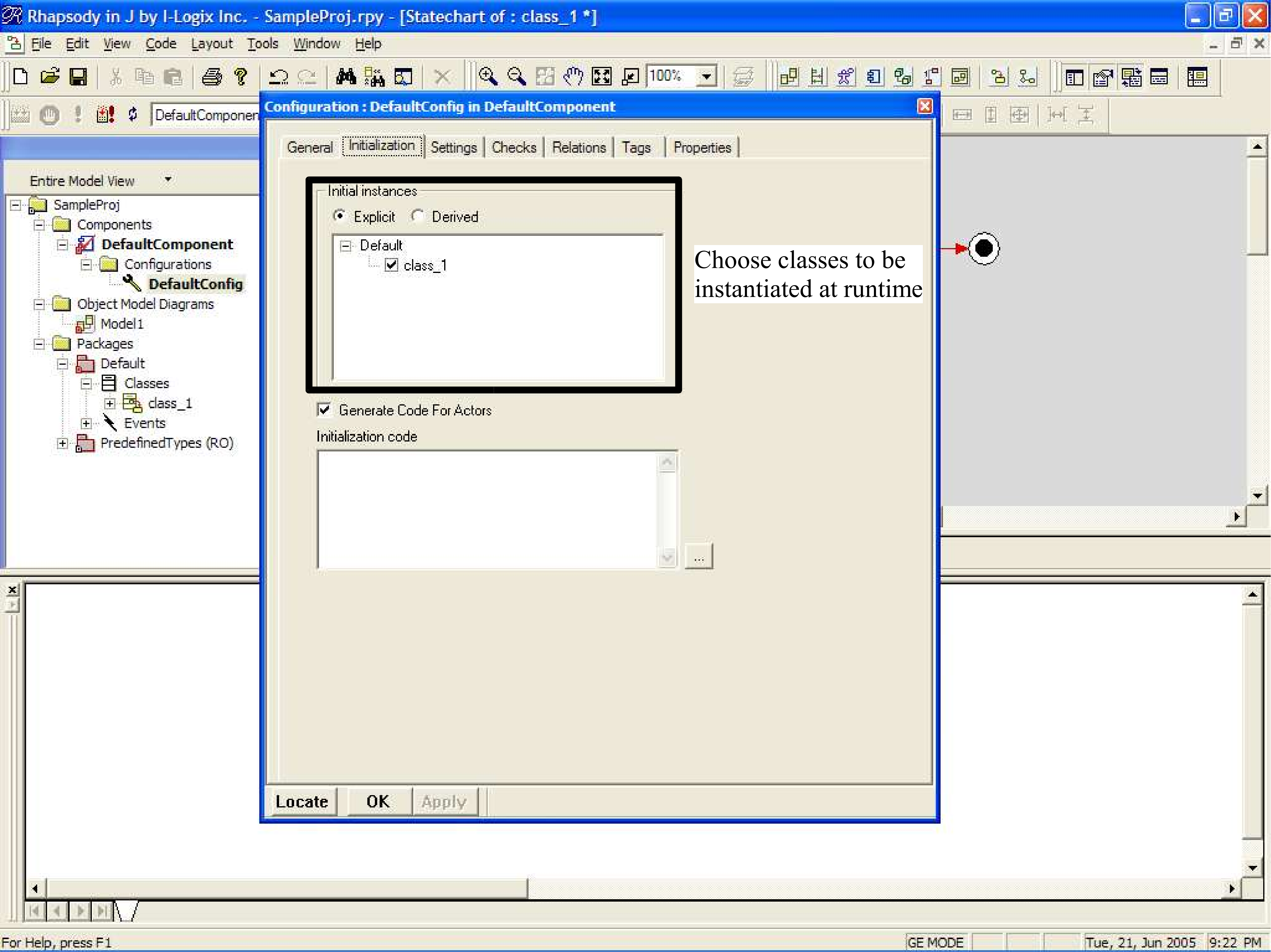

Rhapsody in J by I-Logix Inc. - SampleProj.rpy - [Object Model Diagram: Model1 *]
File Edit View Code Layout Tools Window Help
100%
DefaultComponen
Entire Model View
SampleProj
Components
Object Model Diagrams
Model1
Packages
Default
Classes
class_1
PredefinedTypes (RO)
Class : class_1 in Default
General
Attributes
Operations
Relations
Tags
Properties
Name
Visibility
Type
Initial Value
myInt
Public
int
<New>
int
long
Object
RhpAddress
RhpBoolean
RhpCharacter
RhpInteger
RhpPositive
RhpReal
RhpString
RhpUnlimitedNatural
RhpVoid
Type choices include language-agnostic and user-defined types
Locate
OK
Apply
//## class class_1
public class class_1
public Reactive
protected int my
public static fi
public static fi
protected int ro
protected int rootState_active;
public RiJThread getThread() {
For Help, press F1
GE MODE
Tue, 21, Jun 2005
8:47 PM

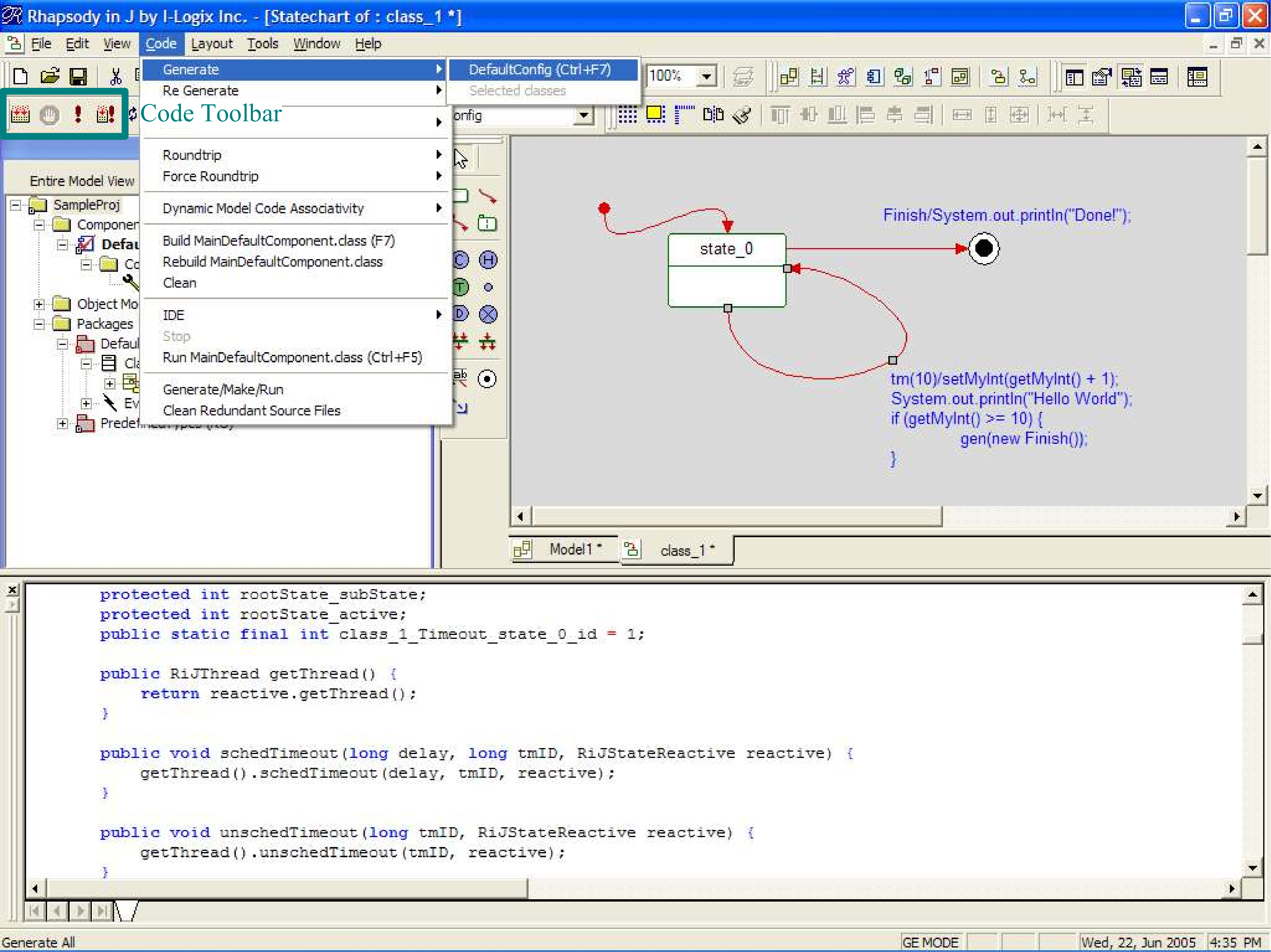


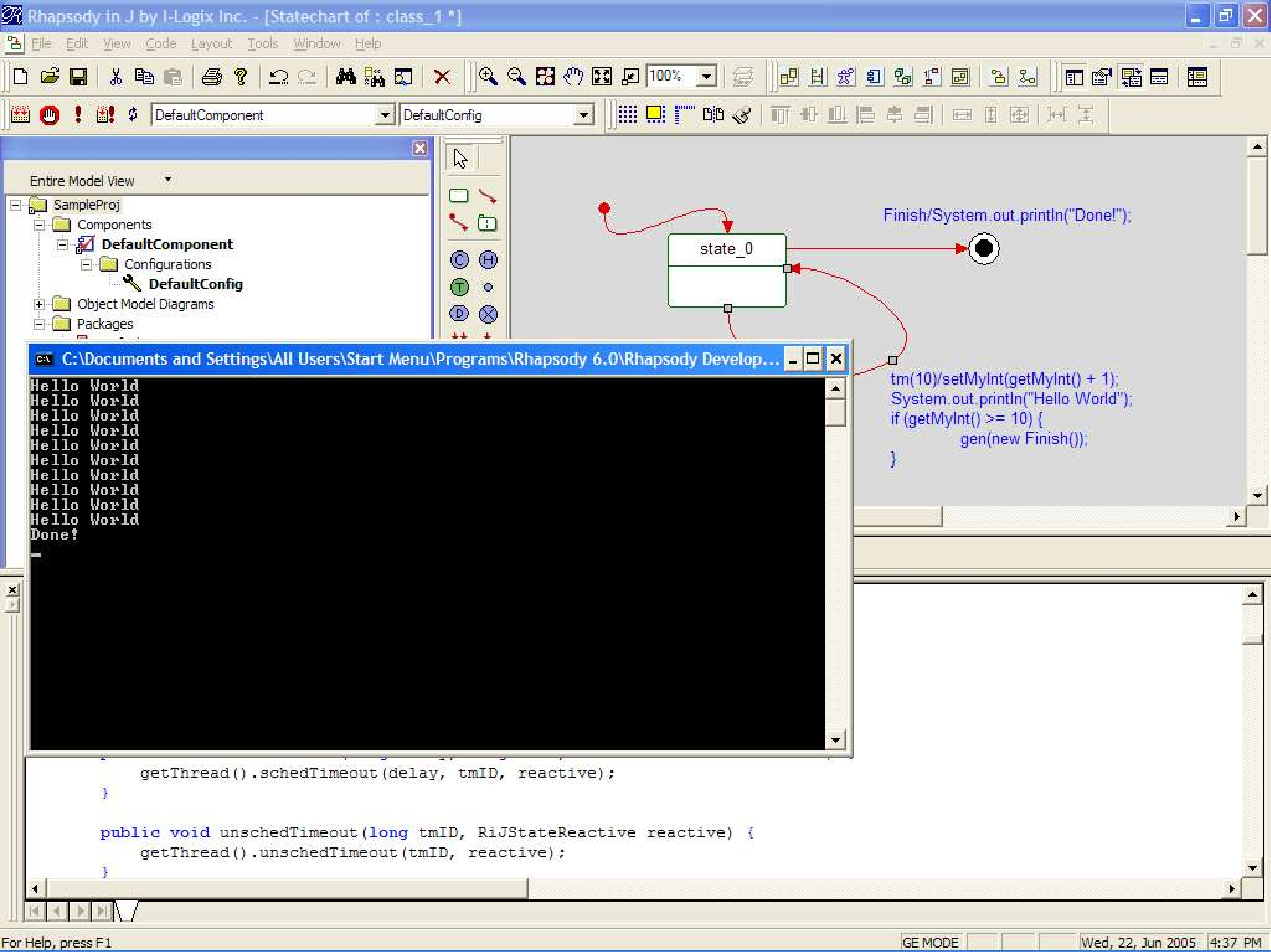
Rhapsody in J by I-Logix Inc. - SampleProj.rpy - [Object Model Diagram: Model1 *]
File Edit View Code Layout Tools Window Help
100%
DefaultComponen
Entire Model View
SampleProj
Components
Object Model Diagrams
Model1
Packages
Default
Classes
class_1
PredefinedTypes (RO)
Class : class_1 in Default
General Attributes Operations Relations Tags Properties
Name Visibility Return Type
PrimitiveOperation
Reception
TriggeredOperation
Constructor
Destructor
Operation choices include Primitive (manually coded), Event Reception, and Triggered Operations
Locate OK Apply
//## class class_1
public class class_1
public Reactive
protected int my
public static fi
public static fi
protected int ro
protected int rootState_active;
public RiJThread getThread() {
For Help, press F1
GE MODE
Tue, 21, Jun 2005
8:50 PM

Features can vary greatly for different model items




Rhapsody in J by I-Logix Inc. - SampleProj.rpy - [Statechart of : class_1 *]
File Edit View Code Layout Tools Window Help
100%
DefaultComponen
Entire Model View
SampleProj
Components
DefaultComponent
Configurations
DefaultConfig
Object Model Diagrams
Model1
Packages
Default
Classes
class_1
Events
PredefinedTypes (RO)
Configuration : DefaultConfig in DefaultComponent
General
Initialization
Settings
Checks
Relations
Tags
Properties
Initial instances
Explicit
Derived
Default
class_1
Generate Code For Actors
Initialization code
...
Locate
OK
Apply
Choose classes to be instantiated at runtime
For Help, press F1
GE MODE
Tue, 21, Jun 2005
9:22 PM


Rhapsody in J by I-Logix Inc. - [Statechart of : class_1 *]
File Edit View Code Layout Tools Window Help
Code Toolbar
Generate
DefaultConfig (Ctrl+F7)
Selected classes
Re Generate
Roundtrip
Force Roundtrip
Dynamic Model Code Associativity
Build MainDefaultComponent.class (F7)
Rebuild MainDefaultComponent.class
Clean
IDE
Stop
Run MainDefaultComponent.class (Ctrl+F5)
Generate/Make/Run
Clean Redundant Source Files
Entire Model View
SampleProj
Object Mo
Packages
state_0
Finish/System.out.println("Done!");
tm(10)/setMyInt(getMyInt() + 1);
System.out.println("Hello World");
if (getMyInt() >= 10) {
gen(new Finish());
}
Model1 *
class_1 *
protected int rootState_subState;
protected int rootState_active;
public static final int class_1_Timeout_state_0_id = 1;
public RiJThread getThread() {
return reactive.getThread();
}
public void schedTimeout(long delay, long tmID, RiJStateReactive reactive) {
getThread().schedTimeout(delay, tmID, reactive);
}
public void unschedTimeout(long tmID, RiJStateReactive reactive) {
getThread().unschedTimeout(tmID, reactive);
}
Generate All
GE MODE
Wed, 22, Jun 2005
4:35 PM

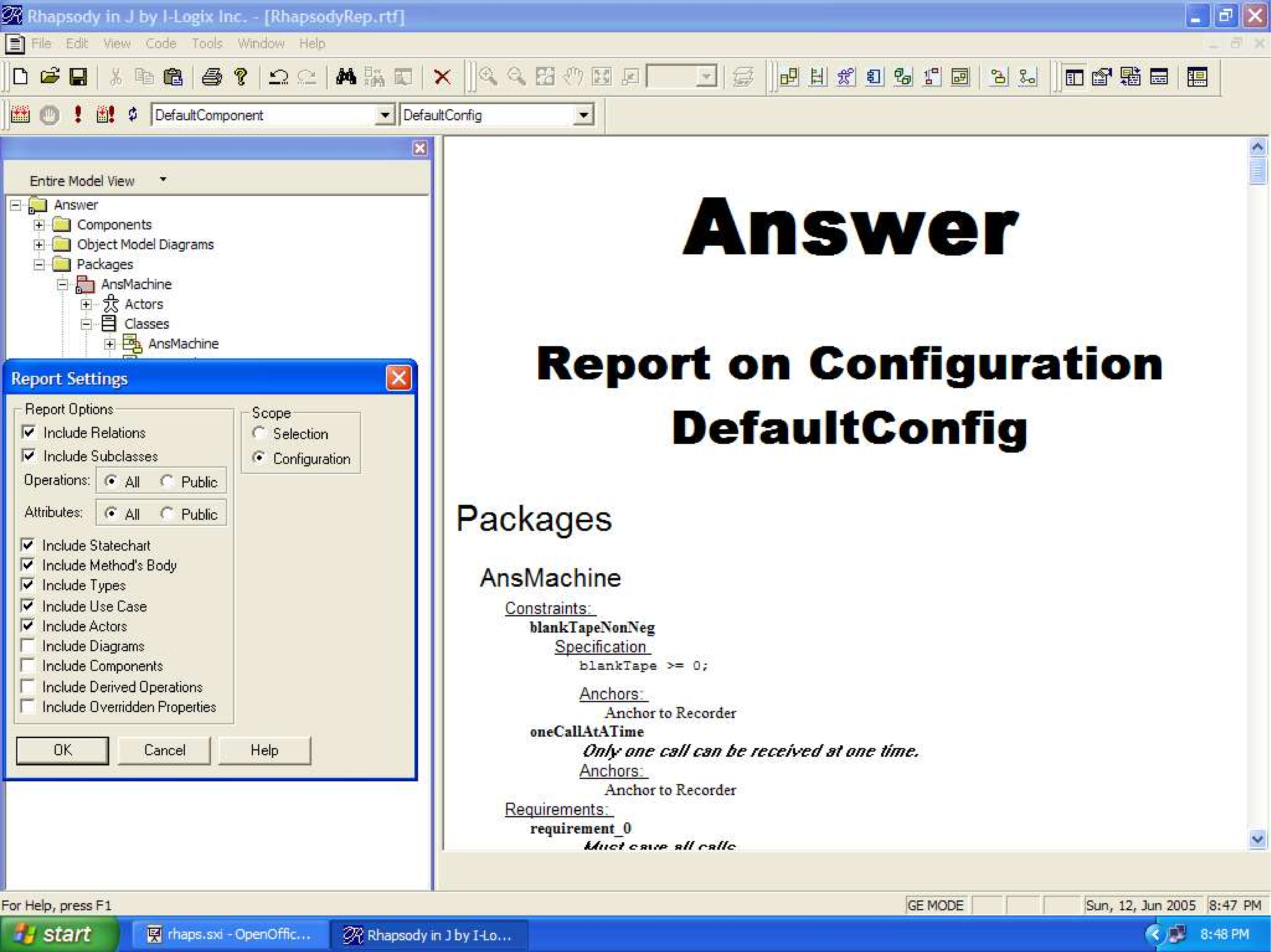

Rhapsody in J by I-Logix Inc. - [Statechart of : class_1 *]
File Edit View Code Layout Tools Window Help
100%
DefaultComponent
DefaultConfig
Entire Model View
SampleProj
Components
DefaultComponent
Configurations
DefaultConfig
Object Model Diagrams
Packages
state_0
Finish/System.out.println("Done!");
tm(10)/setMyInt(getMyInt() + 1);
System.out.println("Hello World");
if (getMyInt() >= 10) {
gen(new Finish());
}
C:\Documents and Settings\All Users\Start Menu\Programs\Rhapsody 6.0\Rhapsody Develop...
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Done!
getThread().schedTimeout(delay, tmID, reactive);
}
public void unschedTimeout(long tmID, RiJStateReactive reactive) {
getThread().unschedTimeout(tmID, reactive);
}
For Help, press F1
GE MODE
Wed, 22, Jun 2005
4:37 PM

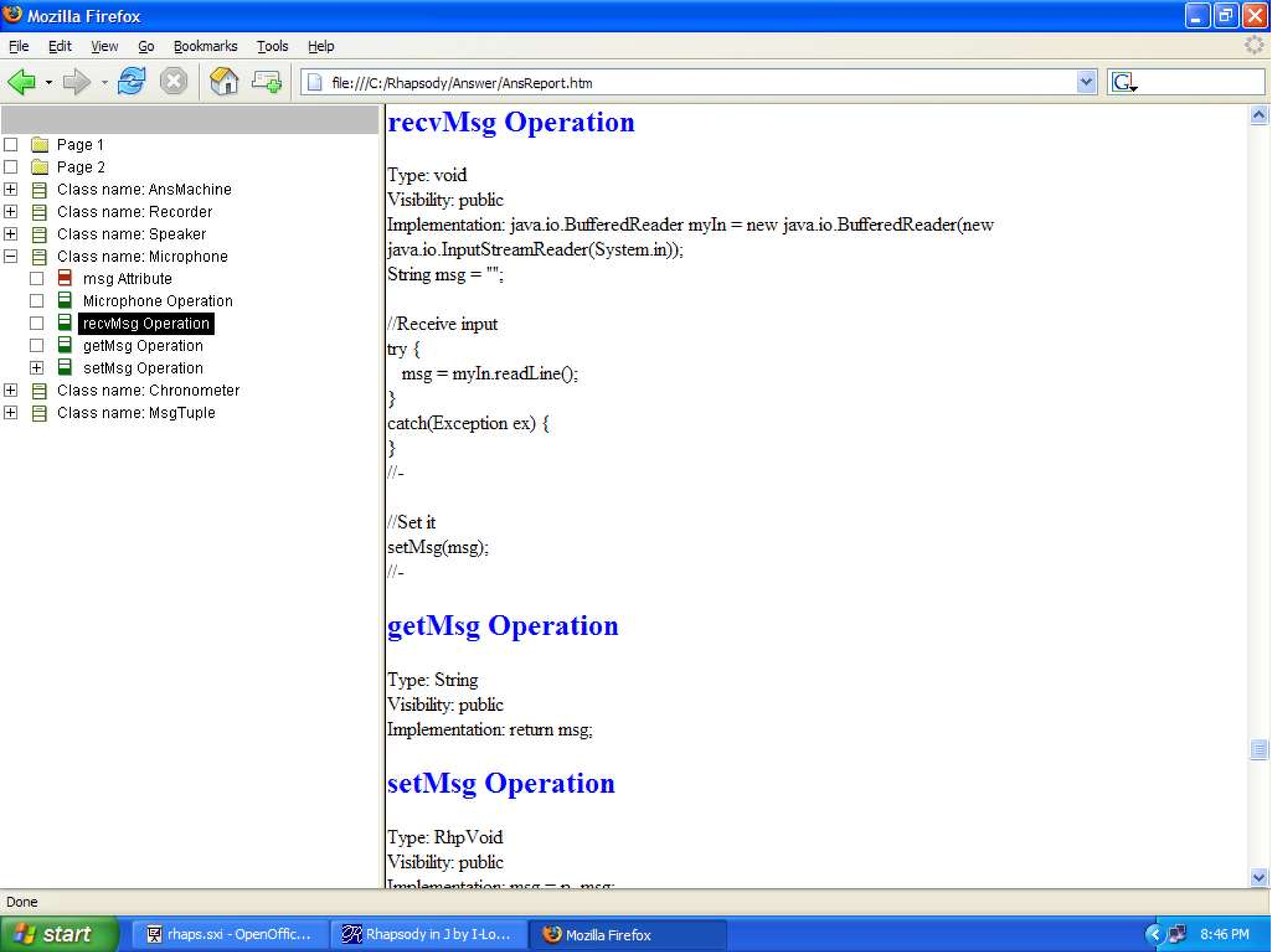
Basic Usage of Rhapsody

1. Create classes (in Browser, OMD, Sequence Diagram, etc.)
2. Create Statechart or Activity Diagram of one or more classes
3. Set up default Component and active Configuration
4. Generate code for active configuration
5. Compile and run active configuration

Important Points

- Model-Based Design
  - Sequence Diagrams are created in Analysis or Design Mode
    - Design: Messages, classes realized on insertion into diagram. Messages deleted from diagram on deletion from model
  - “Roundtripping” is only allowed at code locations between special comment markers
    - “--+[<Type> <Name>” and “--+]” in Ada
    - “/*#[<Type> <Name> */” and “/*#]*/” in C
    - “//#[<Type> <Name>” and “//#]” in C++, Java
  - Rhapsody's Internal Reporter and ReporterPLUS can generate reports in RTF, DOC, HTML, PPT, etc. (examples on slides)


Rhapsody in J by I-Logix Inc. - [RhapsodyRep.rtf]
File Edit View Code Tools Window Help
DefaultComponent
DefaultConfig
Entire Model View
Answer
Components
Object Model Diagrams
Packages
AnsMachine
Actors
Classes
AnsMachine
Report Settings
Report Options
Include Relations
Include Subclasses
Operations: All Public
Attributes: All Public
Include Statechart
Include Method's Body
Include Types
Include Use Case
Include Actors
Include Diagrams
Include Components
Include Derived Operations
Include Overridden Properties
Scope
Selection
Configuration
OK
Cancel
Help
Answer
Report on Configuration
DefaultConfig
Packages
AnsMachine
Constraints:
blankTapeNonNeg
Specification
blankTape >= 0;
Anchors:
Anchor to Recorder
oneCallAtATime
Only one call can be received at one time.
Anchors:
Anchor to Recorder
Requirements:
requirement_0
For Help, press F1
GE MODE
Sun, 12, Jun 2005
8:47 PM
start
rhaps.sxi - OpenOffic...
Rhapsody in J by I-Lo...
8:48 PM



recvMsg Operation

Type: void
Visibility: public
Implementation: java.io.BufferedReader myIn = new java.io.BufferedReader(new java.io.InputStreamReader(System.in));

```
String msg = "";

//Receive input
try {
  msg = myIn.readLine();
}
catch(Exception ex) {
}
//-

//Set it
setMsg(msg);
//-
```

getMsg Operation

Type: String
Visibility: public
Implementation: return msg;

setMsg Operation

Type: RhpVoid
Visibility: public
Implementation: msg = p_msg;

Important Points

- Object/Instance Behavior
  - Each class may have one “state” diagram
    - Activity Diagram
    - Statechart
  - Classes can have one of two thread behaviors
    - Sequential, running in main thread
    - Active, running in own thread
  - Code added to states/actions must be written in the language of the active program (“Rhapsody in C”, “C++”, “Ada”, “Java”)
  - Configurations determine which classes will be instantiated into objects at runtime (at least one class required).

Important Points

- Simulation
  - Runs can be viewed through any number and combination of the following animated diagrams
    - Activity Diagrams
    - Sequence Diagrams
    - Statecharts
  - User can use Rhapsody's built-in, text-only tracer to advance through runs
  - During runs, user can manually send events to system through Event Generator

- Third-Party Software
  - Rational Rose models can be imported
  - Rhapsody models can be exported to DOORS

ROPES Development Process

- ROPES = "Rapid Object-oriented Process for Embedded Systems"

  by Bruce Powel Douglass (of I-Logix)
  - Iterative development process
  - Works for both elaborative and translative development, better with translative

- Outline
  1. Analysis
  2. Design
  3. Translation
  4. Testing

ROPES: Analysis

- Requirements Analysis
  - Create use cases, scenarios
  - Discover necessary constraints
  - Discover external factors that affect system
  - Discover possible system hazards, risks
    - Sequence Diagrams
    - Statecharts
    - Use Case Diagrams
- Systems Analysis
  - Separate system into functional segments
  - Design high-level algorithms for these segments

ROPES: Analysis

- Categorize system functions as software, electronics, or mechanics
- Test the segments
  - Activity Diagrams
  - Component Diagrams
  - Sequence Diagrams
  - Statecharts

- Object Analysis
  - Design required classes and objects for system
  - Test the classes and objects
    - Activity Diagrams
    - Collaboration Diagrams

ROPES: Analysis

- Component Diagrams
- Object Model Diagrams
- Sequence Diagrams
- Statecharts

ROPES: Design

- Architectural Design
  - Determine number and usage of threads
  - Utilize design patterns for error handling, safety, fault tolerance
    - Activity Diagrams
    - Collaboration Diagrams
    - Component Diagrams
    - Object Model Diagrams
    - Sequence Diagrams
    - Statecharts

- Mechanistic Design
  - Utilize design patterns to facilitate collaboration

ROPES: Design

  - Collaboration Diagrams
  - Component Diagrams
  - Object Model Diagrams
  - Sequence Diagrams

- Detailed Design
  - Specifically design internals of classes, associations with other classes
    - Activity Diagrams
    - Object Model Diagrams
    - Statecharts

ROPES: Translation & Testing

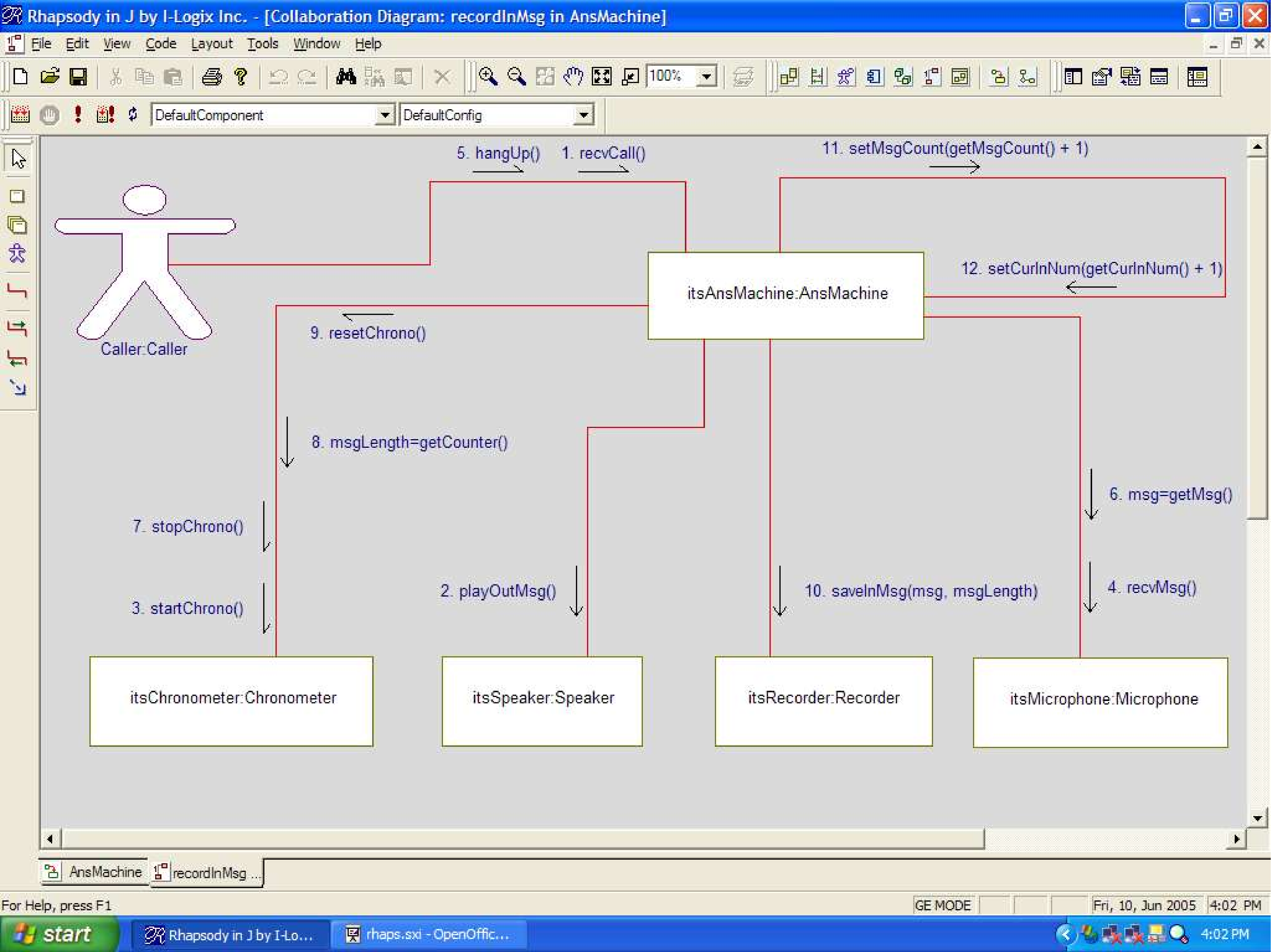
- Translation
  - Transform model information into source code
    - Rhapsody Code Generation
- Testing
  - Follow a planned testing document
  - Add one component at a time during integration testing
  - Run validation tests (black box)
  - Run safety tests (white box)

Example: Answering Machine

- Requirements Analysis
  - Use cases
    - Recording/Playing back messages
    - Recording outgoing message
    - Displaying number of recorded messages
    - Recording incoming messages
    - *(2nd Iter.)* Keep track of message lengths, blank tape
  - External factors
    - Length of tape/Amount of memory in answering machine

Example: Answering Machine

- Systems Analysis
  - Split Answering Machine components into groups
    - AnsMachine (Software)
    - Hardware
    - *(2nd Iter.)* Caller & Owner Agents
- Object Analysis
  - Design classes and objects, interactions between them
    - (Collaboration Diagram on next slide)


Rhapsody in J by I-Logix Inc. - [Collaboration Diagram: recordInMsg in AnsMachine]
File Edit View Code Layout Tools Window Help
100%
DefaultComponent
DefaultConfig
5. hangUp()
1. recvCall()
11. setMsgCount(getMsgCount() + 1)
12. setCurlnNum(getCurlnNum() + 1)
itsAnsMachine:AnsMachine
Caller:Caller
9. resetChrono()
8. msgLength=getCounter()
6. msg=getMsg()
7. stopChrono()
3. startChrono()
2. playOutMsg()
10. saveInMsg(msg, msgLength)
4. recvMsg()
itsChronometer:Chronometer
itsSpeaker:Speaker
itsRecorder:Recorder
itsMicrophone:Microphone
AnsMachine
recordInMsg ...
For Help, press F1
GE MODE
Fri, 10, Jun 2005
4:02 PM
start
Rhapsody in J by I-Lo...
rhaps.sxi - OpenOffic...
4:02 PM

Example: Answering Machine

- Architectural Design
  - Design threads
    1. Main system thread
    2. *(2nd Iter.)* Caller agent thread
    3. *(2nd Iter.)* Owner agent thread
  - Use Design Patterns for error handling, safety, fault tolerance
    - No safety or fault tolerance concerns
    - Found patterns generally not applicable to example

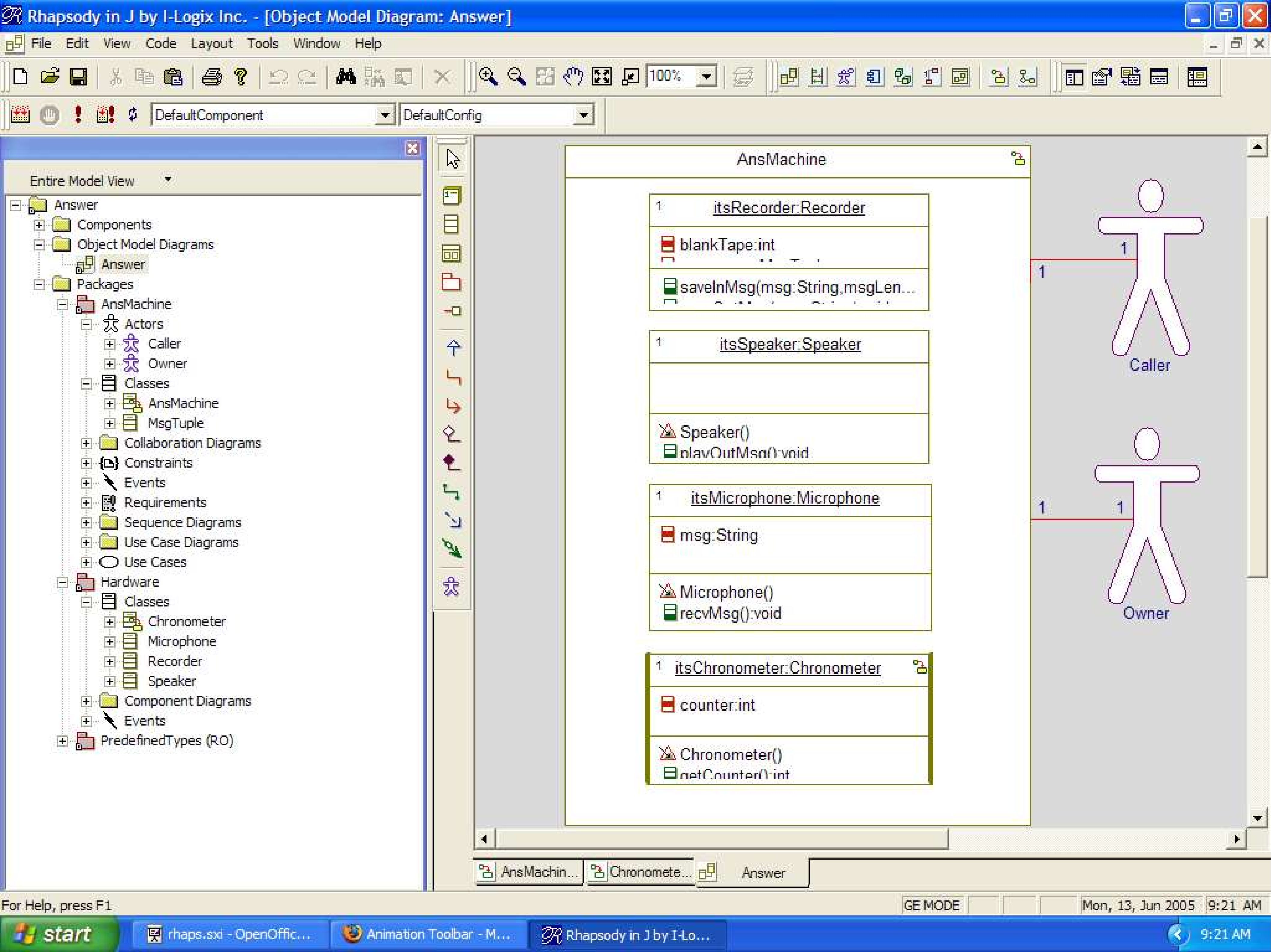
- Mechanistic Design
  - Use Design Patterns to aid in collaboration
    - Found patterns generally not applicable to example

Example: Answering Machine

- Detailed Design (OMD in two slides)
  - AnsMachine
    - Contains one instance each of Chronometer, Microphone, Recorder, Speaker
    - Takes events from Caller and Owner
  - Microphone
    - Receives incoming messages from Caller and outgoing message from Owner
  - Recorder
    - Saves message data
    - Discards data if “blankTape” is less than message length
  - Speaker
    - Plays outgoing message to Caller
    - Plays incoming messages to Owner

Example: Answering Machine

    - Plays informational messages
- *(2nd Iter.)* Caller
    - Makes calls and leaves incoming messages by sending events to AnsMachine
- *(2nd Iter.)* Owner
    - Sets outgoing message
    - Hears, deletes incoming messages by sending events to AnsMachine
- *(2nd Iter.)* Chronometer
    - Tracks lengths of messages as Caller “speaks” into microphone
- *(3rd Iter.)* MsgTuple
    - Contains string data “msg”
    - Contains integer data “msgLength”


Rhapsody in J by I-Logix Inc. - [Object Model Diagram: Answer]
File Edit View Code Layout Tools Window Help
DefaultComponent
DefaultConfig
Entire Model View
Answer
Components
Object Model Diagrams
Answer
Packages
AnsMachine
Actors
Caller
Owner
Classes
AnsMachine
MsgTuple
Collaboration Diagrams
Constraints
Events
Requirements
Sequence Diagrams
Use Case Diagrams
Use Cases
Hardware
Classes
Chronometer
Microphone
Recorder
Speaker
Component Diagrams
Events
PredefinedTypes (RO)
AnsMachine
itsRecorder:Recorder
blankTape:int
saveInMsg(msg:String,msgLen...
itsSpeaker:Speaker
Speaker()
playOutMsg():void
itsMicrophone:Microphone
msg:String
Microphone()
recvMsg():void
itsChronometer:Chronometer
counter:int
Chronometer()
getCounter():int
Caller
Owner
AnsMachin...
Chronomete...
Answer
For Help, press F1
GE MODE
Mon, 13, Jun 2005
9:21 AM
start
rhaps.sxi - OpenOffic...
Animation Toolbar - M...
Rhapsody in J by I-Lo...
9:21 AM

Example: Answering Machine

Demonstration

Personal Experiences using Rhapsody

- Ease of Use
  - Appears intuitive, but surprises can confuse new users
    - Some models and model items generate code while others do not
    - Setting up a Default Configuration incorrectly can cause compilation errors
  - New user will probably consult Rhapsody manual often, but it is often lacking
    - Manual is C++-centric
    - Manual does not discuss model to code translation
  - Few Java example projects, many C++ examples; but very few use Activity Diagrams
  - Auto-realization of operations, events very useful

Personal Experiences using Rhapsody

- Extent of Model-Driven Design
  - Depends on user
    - Statecharts cannot have sub-activity diagrams and vice versa. This limits extent of model-driven design
    - Round-tripping may allow user to greatly ignore model
- Stability of Rhapsody
  - Glitches
    - Animated Sequence Charts must be opened from menu to animate properly
    - Collaboration Diagrams can improperly number messages if new messages are inserted
  - Occasionally, .rpy files corrupted while saving
  - Occasional, inexplicable crashes occur

Conclusion

- I-Logix Rhapsody
  - has
    - simple model-drawing, model-defining tools
    - useful thread control mechanisms
    - a variety of report generation styles
    - powerful animation/simulation and debugging tools
    - the ability to generate code in several programming languages
  - but lacks
    - a language-agnostic User Guide that answers the questions new users will have
    - the stability it should have (at least, in my experience)
    - language-agnosticism as its focus, a little idealism(?). Rhapsody is strictly utilitarian

References

Douglass, Bruce Powel. “ROPES: Rapid Object-oriented Process for Embedded Systems”. 1999.

I-Logix. “Getting Started with Rhapsody”.

I-Logix. “Rhapsody Tutorial in C++”. Release 5.2. 2004.

I-Logix. “Rhapsody Tutorial in Java”. Release 4.1 MR2. 2003.

I-Logix. “Rhapsody User Guide”.

I-Logix. “Properties Reference Guide”.

I-Logix. “Rhapsody List of Books”.

I-Logix. “Using Third-Party Tools with Rhapsody”.