Cadena:

Using Cadena to Define a CCM Development Environment

SAnToS Laboratory, Kansas State University, USA

http://cadena.projects.cis.ksu.edu

Adam Childs
Jesse Greenwald
John Hatcliff
Matt Hoosier
Georg Jung

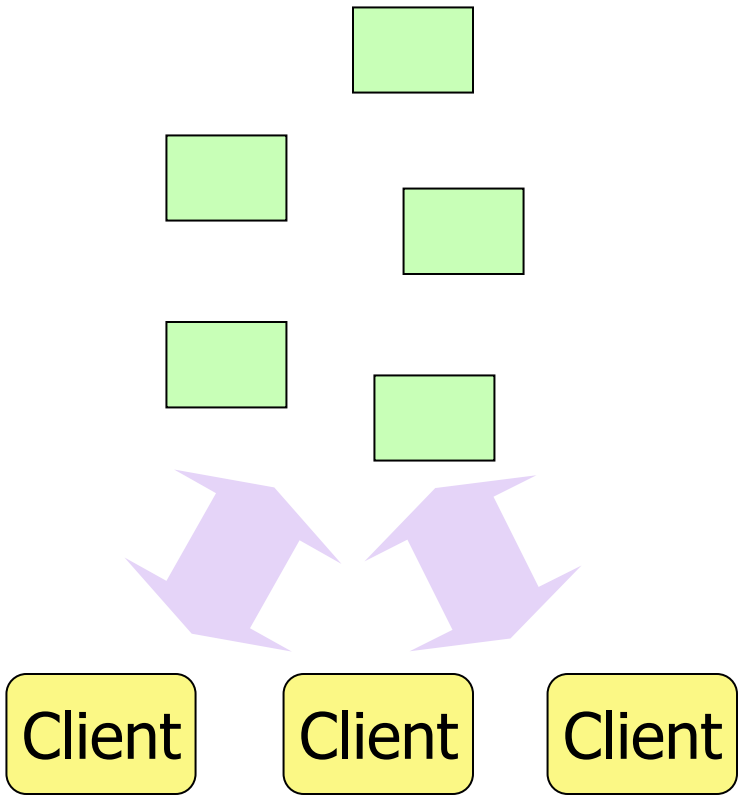
Support

US Army Research Office (ARO)
US National Science Foundation (NSF)
US Department of Defense
Advanced Research Projects Agency (DARPA)

IBM Eclipse

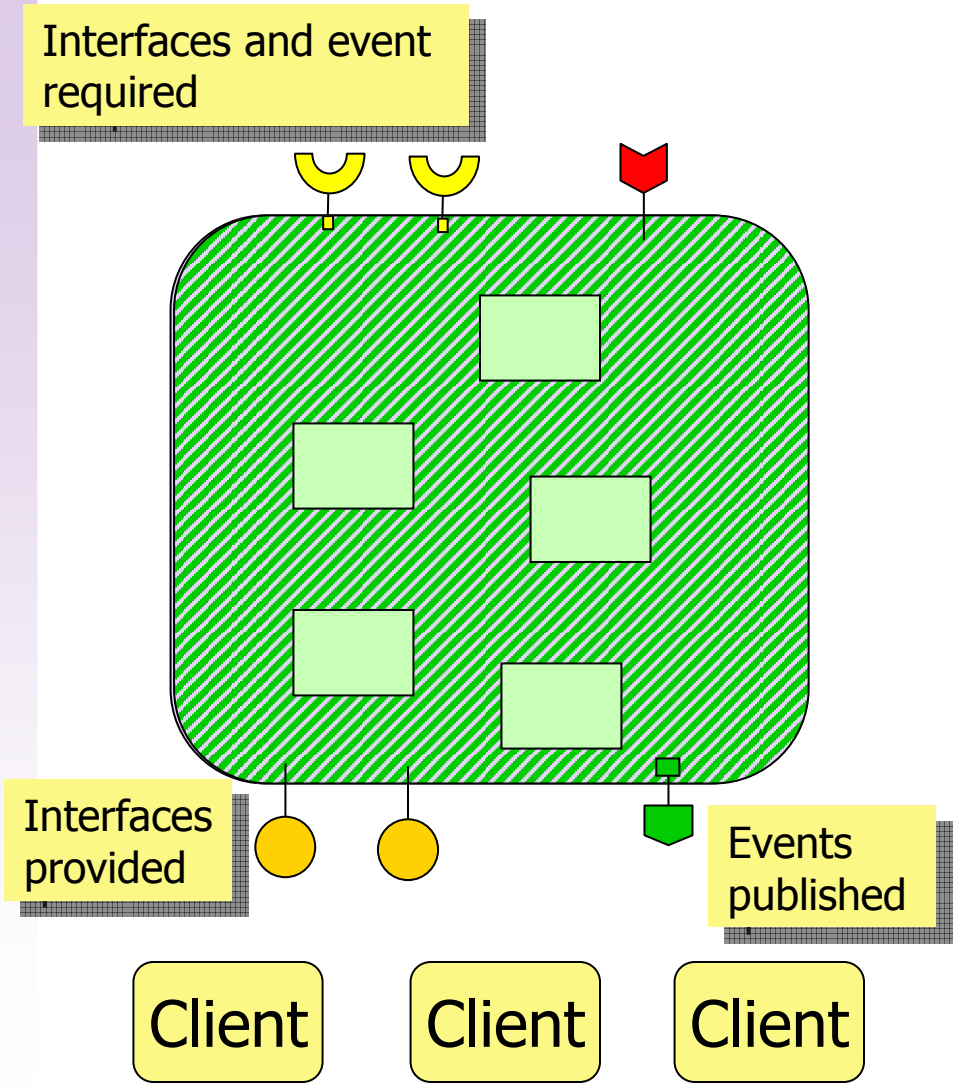
Lockheed Martin

Objects To Components



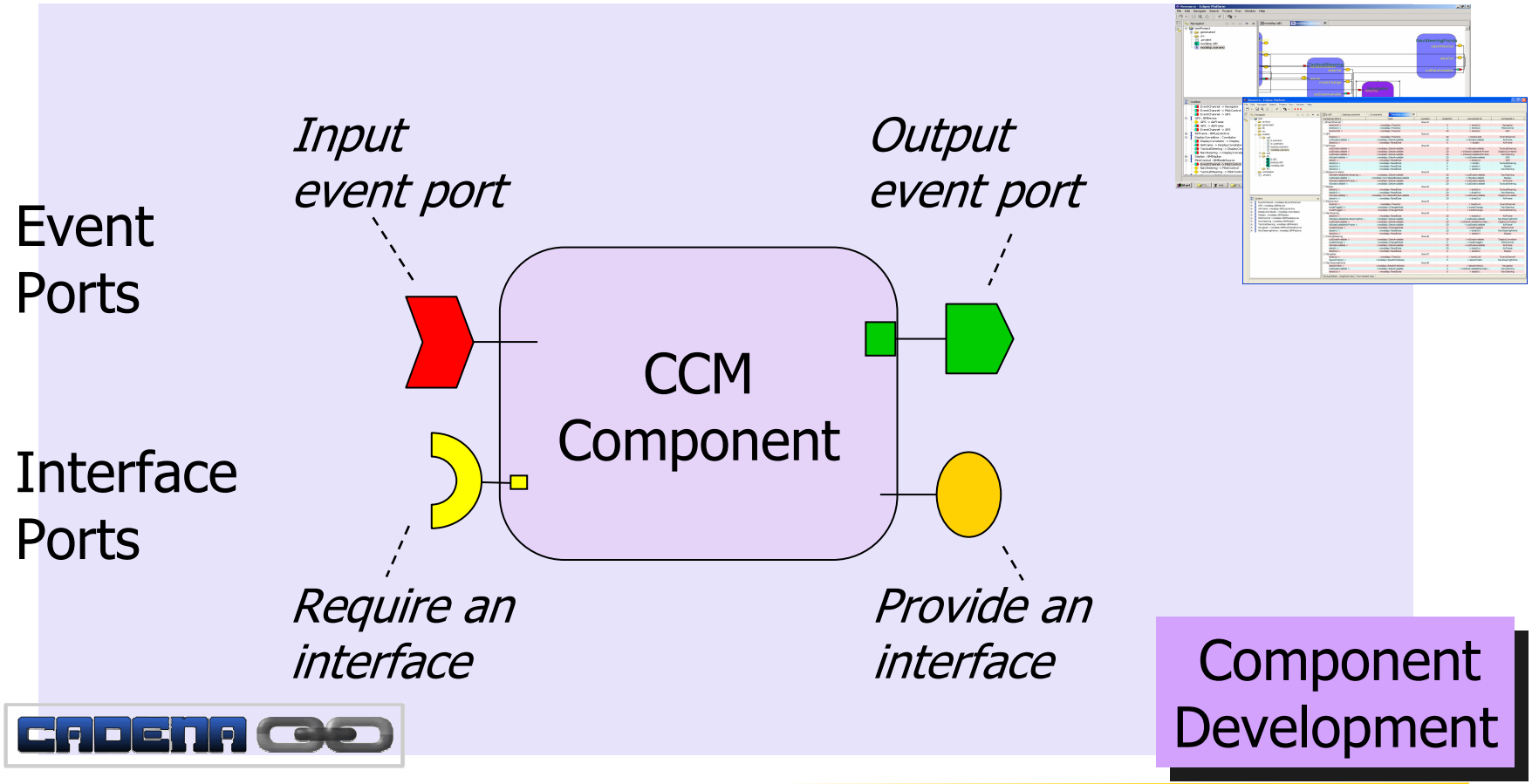
- *Consider*: group of objects working together to provide a service to clients
- Objects are meant to be used "as a team"
  - unit of composition
- No language mechanism to
  - identify components as a single group
  - explicitly define interfaces
  - explicitly define dependences on other 'groups'
- Harder for 3rd parties to reuse and assemble

Objects To Components



- Components collects related classes together to form a coarser-grain composable unit
- Components explicitly define interfaces they provide to their clients
- Components indicate the other interfaces/events they depend on
- Considerable auto-coding functionality provide

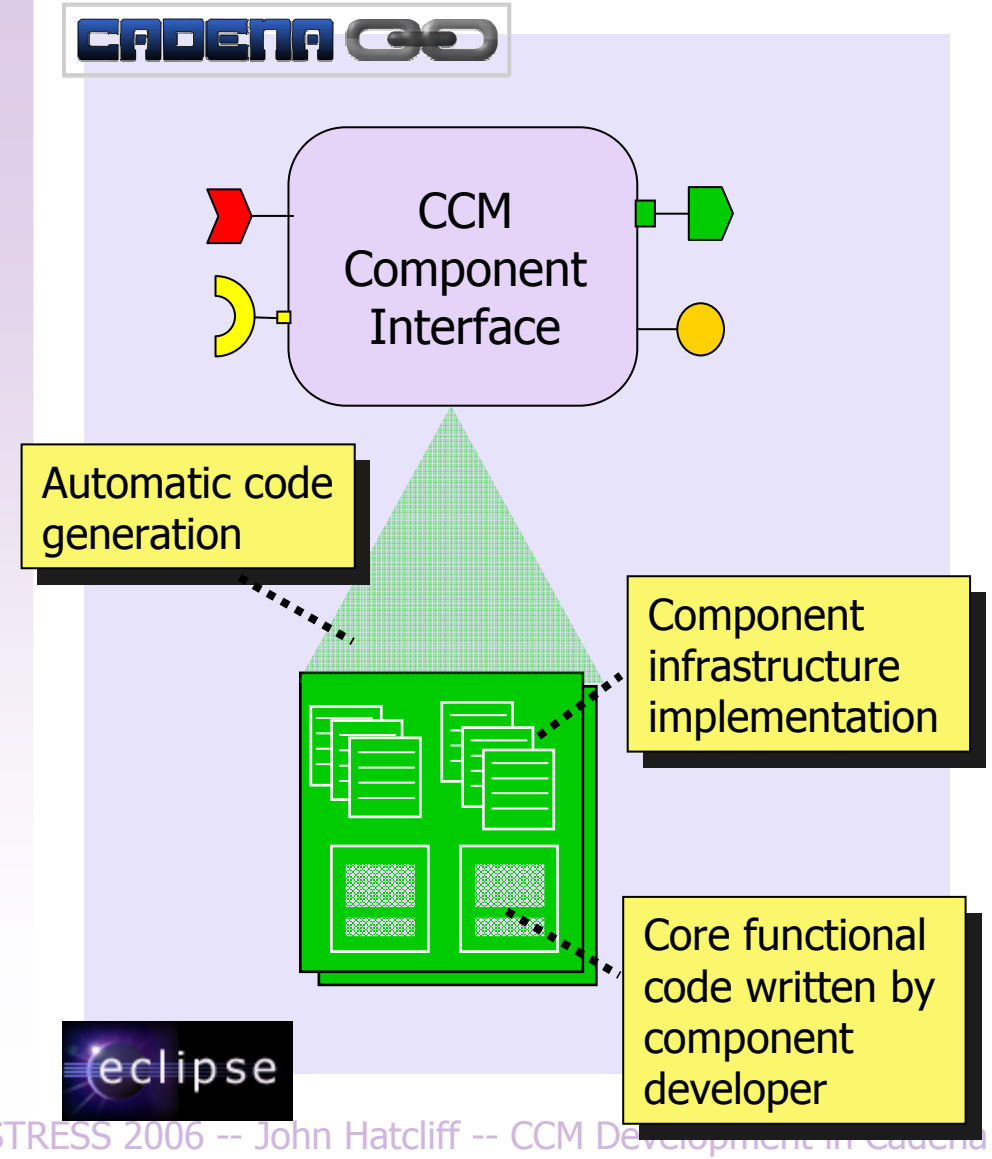
Component-based Design



Component Development

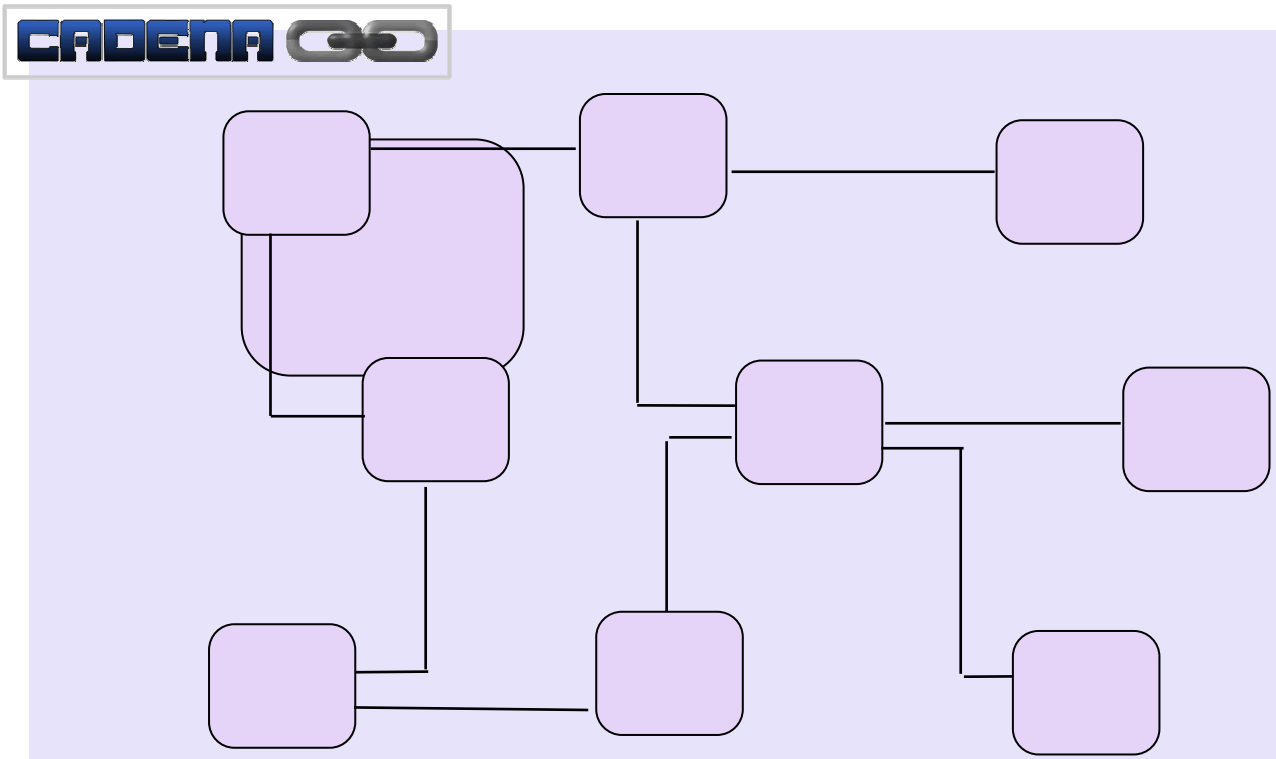
Cadena development environment enables model-based development of applications using frameworks such as CORBA Component Model (CCM)

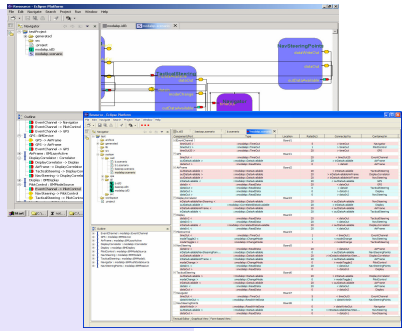
Component Development



- Development of component interfaces using CCM *Interface Definition Language*
- Automatic generation of component infrastructure code using CCM IDL compilers
- Development of core functional code (business logic) using Eclipse Java facilities

Component Integration

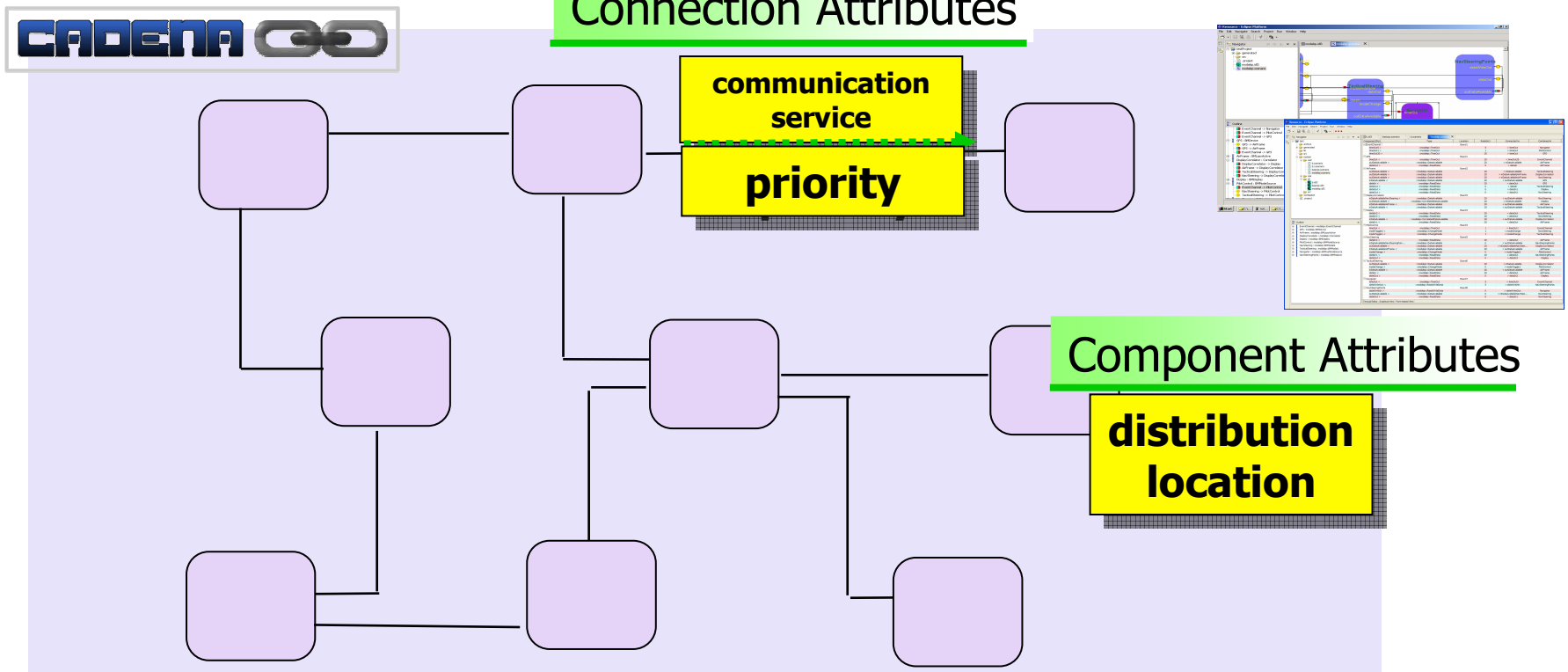




Component Integration

Multiple views for allocating component instances and connecting components together to form a *system assembly*

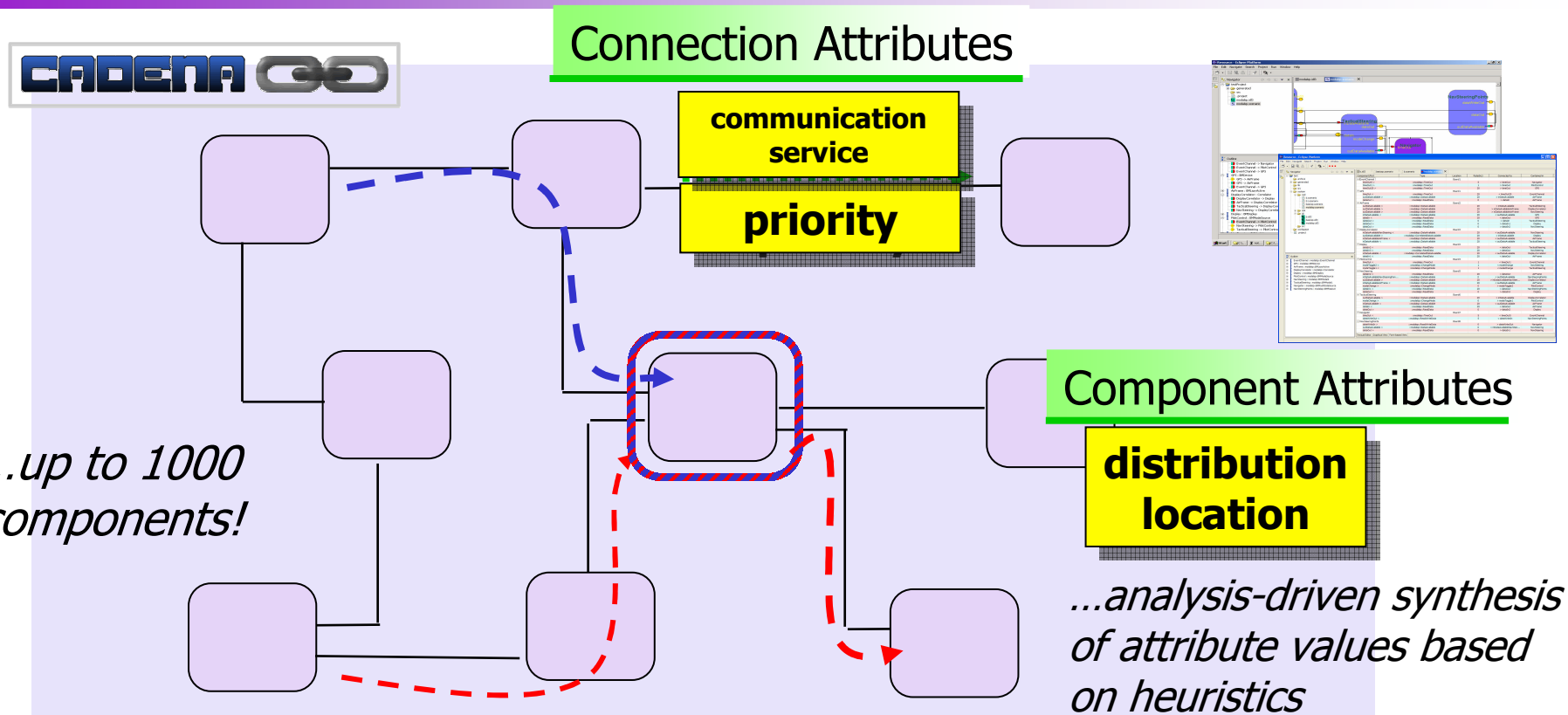
Model-based Programming



Programming at a higher level of abstraction…

Many system elements – configuration of communication services, setting of QoS properties, etc. – are *programmed* by selecting particular attribute values at the model level.

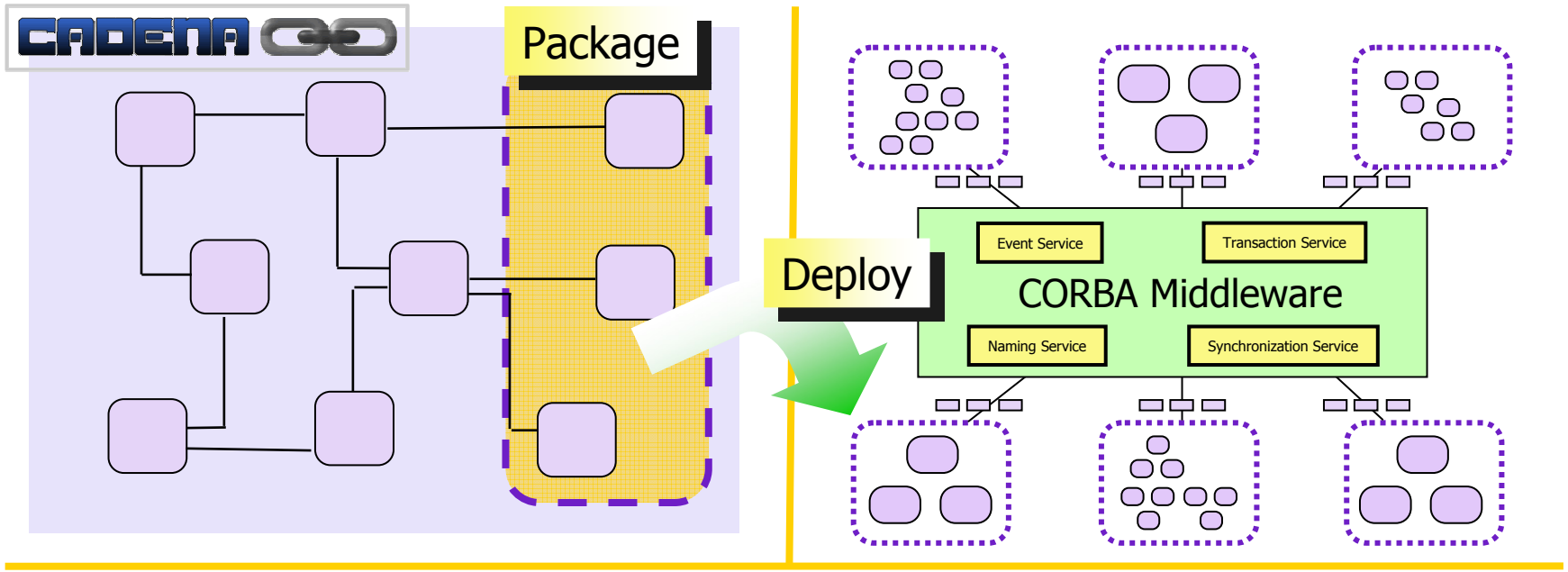
Model-level Analysis



Various analyses guide system development…

Analysis facilities provides multiple forms of a design-level slicing, chopping, etc. and model-checking of global temporal properties.

Packaging & Deployment



automatic generation

```
<CONFIGURATION_PASS>
  <HOME>          <...>
    <COMPONENT>
    <ID> <...></ID>
    <EVENT_SUPPLIER>
    <...events this component supplies...>
    </EVENT_SUPPLIER>
    </COMPONENT>
    </HOME>
</CONFIGURATION_PASS>
```

XML-based Configuration and Deployment information

CCM Deployment Infrastructure

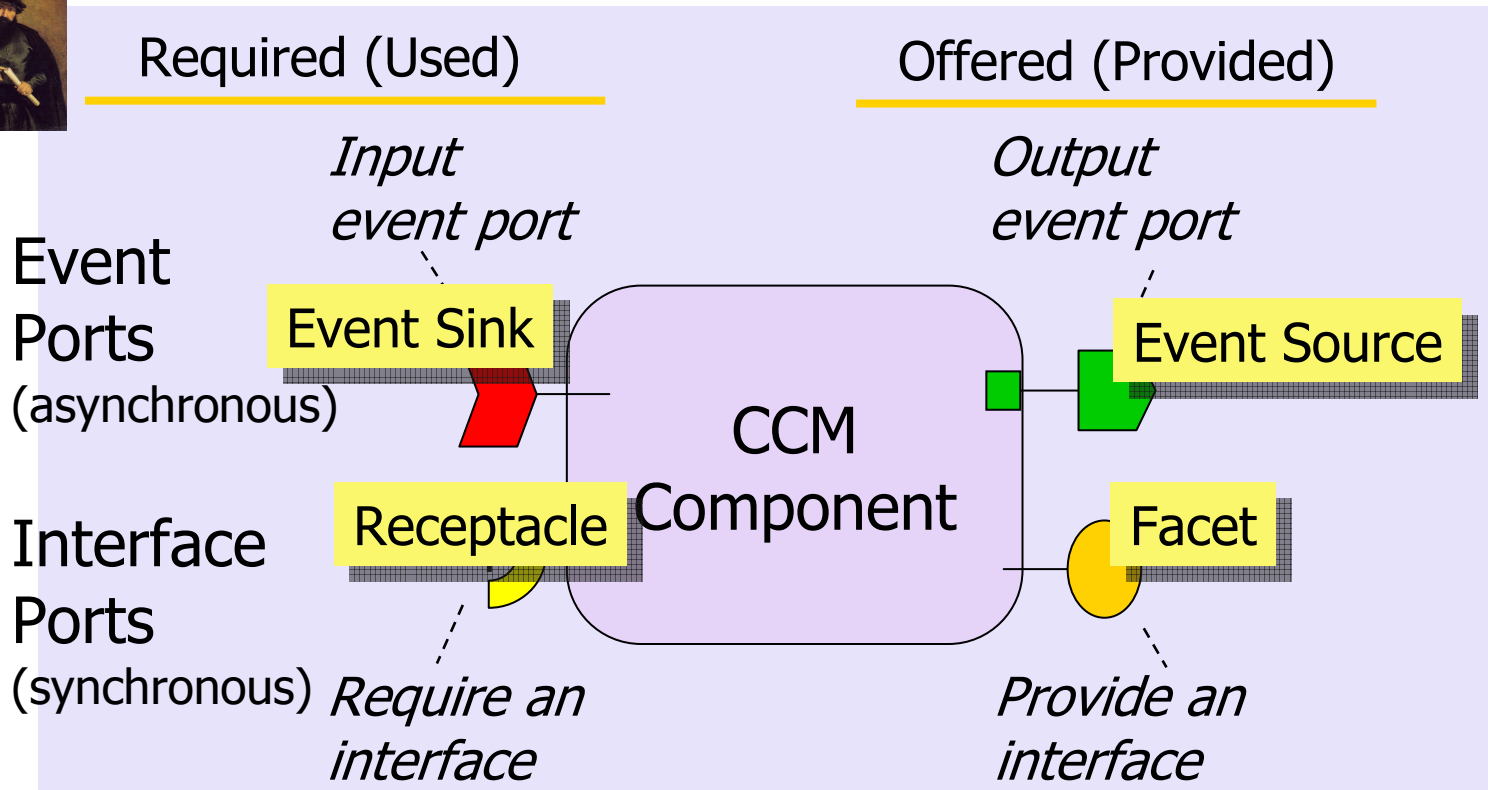
Warning!

- The CCM Specification is over 1000 pages long
- The official OMG CCM tutorial (which by no means covers all topics from the spec) is 215 slides long
- There are many aspects of CCM that we will not address
  - container model, component implementation descriptor language, packaging descriptors, details of the deployment framework, etc.

Outline

- Preparing the CCM Development Environment
  - Defining CCM Style
  - Defining Plug-ins
  - Product-line Architect also sets development process or expected work flow
- Using the environment to build a small implementation

Domain Analysis -- CCM

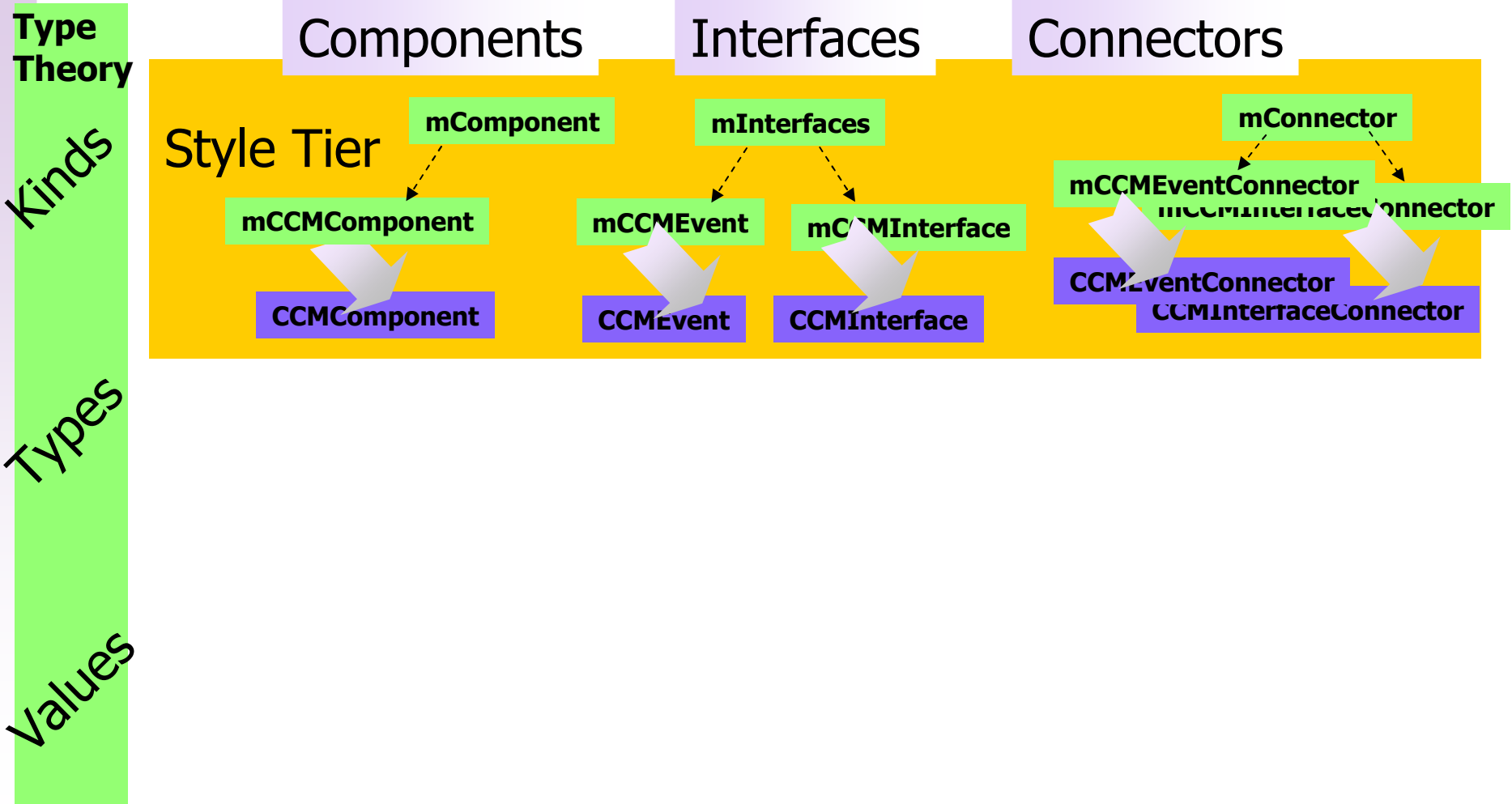


Key: *CCM Terminology*

Shapes of a CCM Component Interface

Declaring Kinds to Define CCM

Define an architectural style that captures CCM

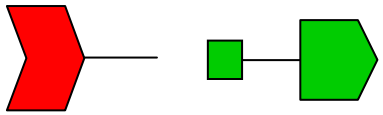


Modeling: CCM Interfaces

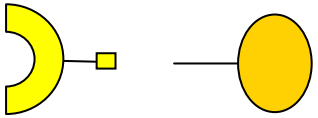
CCM has two classes of ports (asynchronous, synchronous), so create two distinct Cadena interface kinds from which interfaces for CCM ports will be drawn.

CCM

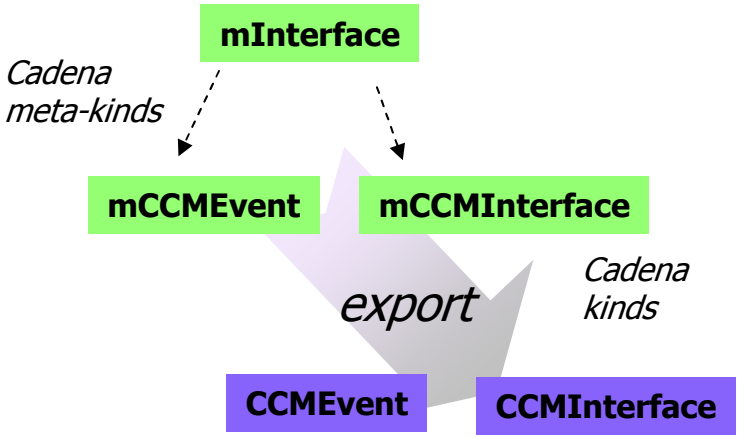
Events (asynchronous)



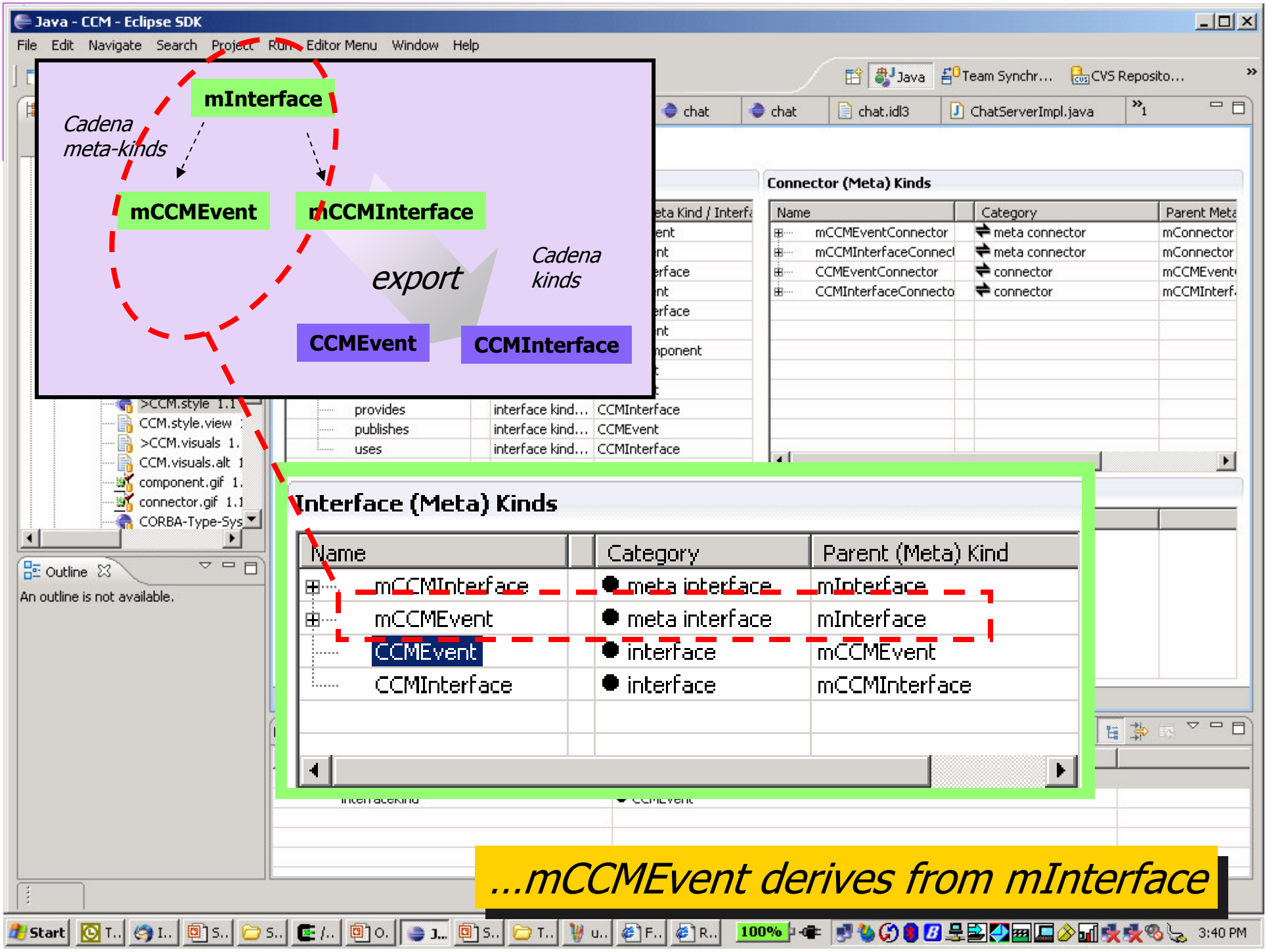
Interfaces (synchronous)

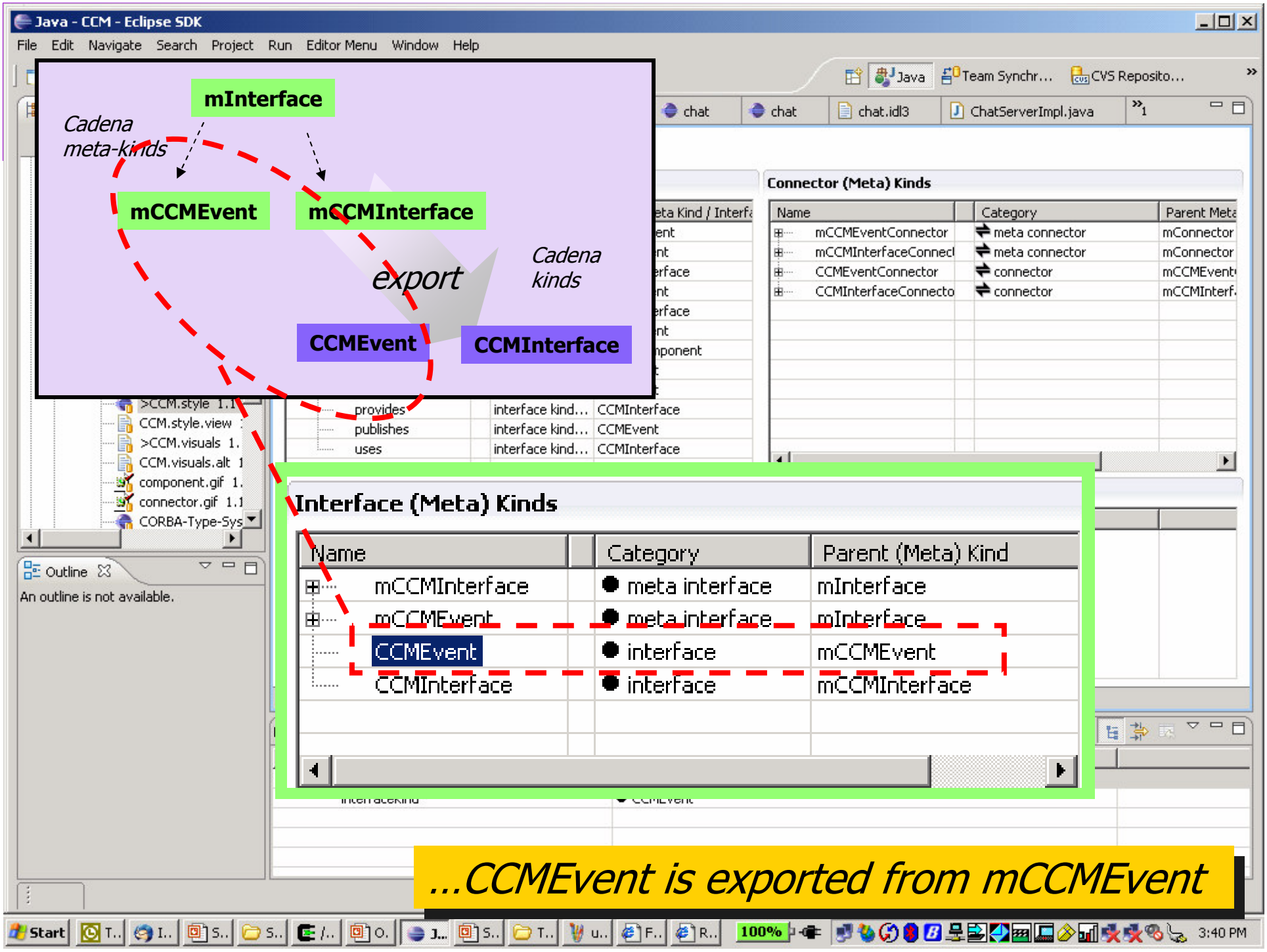


Modeling in Cadena CCM Style

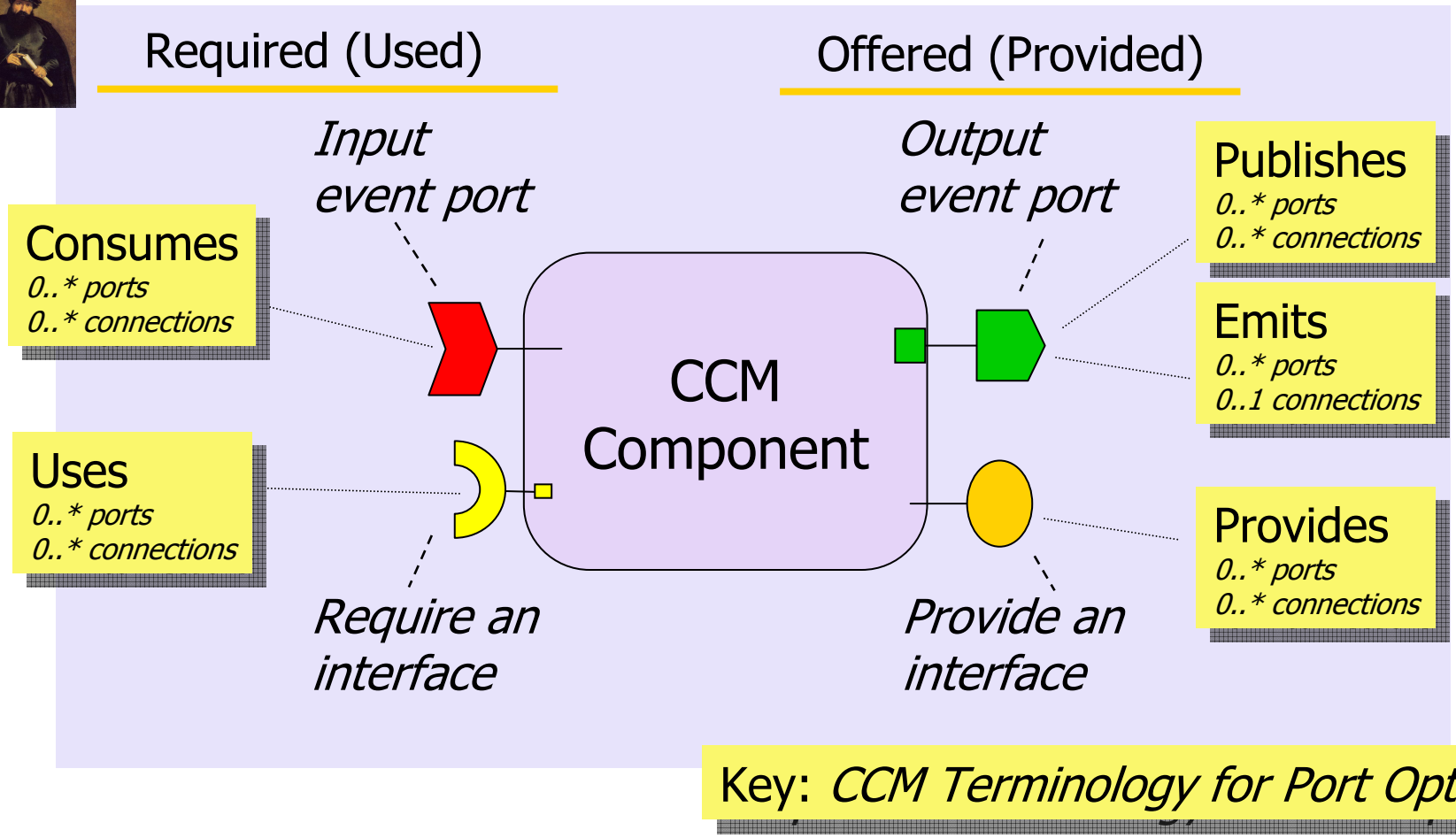


...illustrate this using the event interface


Java - CCM - Eclipse SDK
File Edit Navigate Search Project Run Editor Menu Window Help
mInterface
Cadena meta-kinds
mCCMEvent
mCCMInterface
export
Cadena kinds
CCMEvent
CCMInterface
Connector (Meta) Kinds
Name
Category
Parent Meta
mCCMEventConnector
meta connector
mConnector
mCCMInterfaceConnecl
meta connector
mConnector
CCMEventConnector
connector
mCCMEvent
CCMInterfaceConnecto
connector
mCCMInterf
provides
interface kind...
CCMInterface
publishes
interface kind...
CCMEvent
uses
interface kind...
CCMInterface
Interface (Meta) Kinds
Name
Category
Parent (Meta) Kind
mCCMInterface
meta interface
mInterface
mCCMEvent
meta interface
mInterface
CCMEvent
interface
mCCMEvent
CCMInterface
interface
mCCMInterface
Outline
An outline is not available.
...mCCMEvent derives from mInterface
Start
100%
3:40 PM


Java - CCM - Eclipse SDK
File Edit Navigate Search Project Run Editor Menu Window Help
mInterface
Cadena meta-kinds
mCCMEvent
mCCMInterface
export
Cadena kinds
CCMEvent
CCMInterface
Connector (Meta) Kinds
Name
Category
Parent Meta
mCCMEventConnector
meta connector
mConnector
mCCMInterfaceConnecl
meta connector
mConnector
CCMEventConnector
connector
mCCMEvent
CCMInterfaceConnecto
connector
mCCMInterf
provides
interface kind...
CCMInterface
publishes
interface kind...
CCMEvent
uses
interface kind...
CCMInterface
Interface (Meta) Kinds
Name
Category
Parent (Meta) Kind
mCCMInterface
meta interface
mInterface
mCCMEvent
meta interface
mInterface
CCMEvent
interface
mCCMEvent
CCMInterface
interface
mCCMInterface
Outline
An outline is not available.
...CCMEvent is exported from mCCMEvent

Modeling: CCM Components

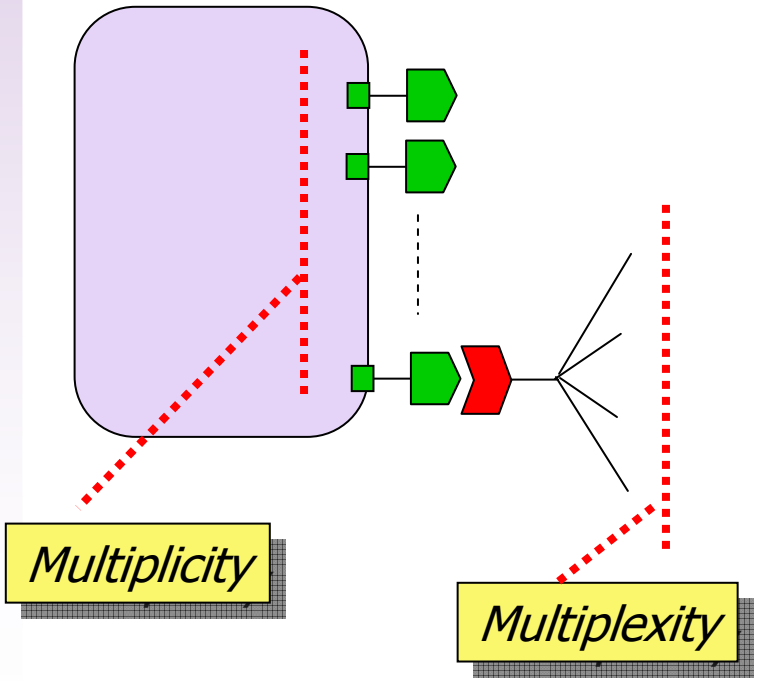


Shapes of a CCM Component Interface

Modeling: Port Options

A *Port Option* in a Cadena component style declaration represents a capacity for declaring ports that adhere to certain constraints

CCM



Modeling in Cadena CCM Style

Port Option

Name: *<name of port option>*
e.g., emits

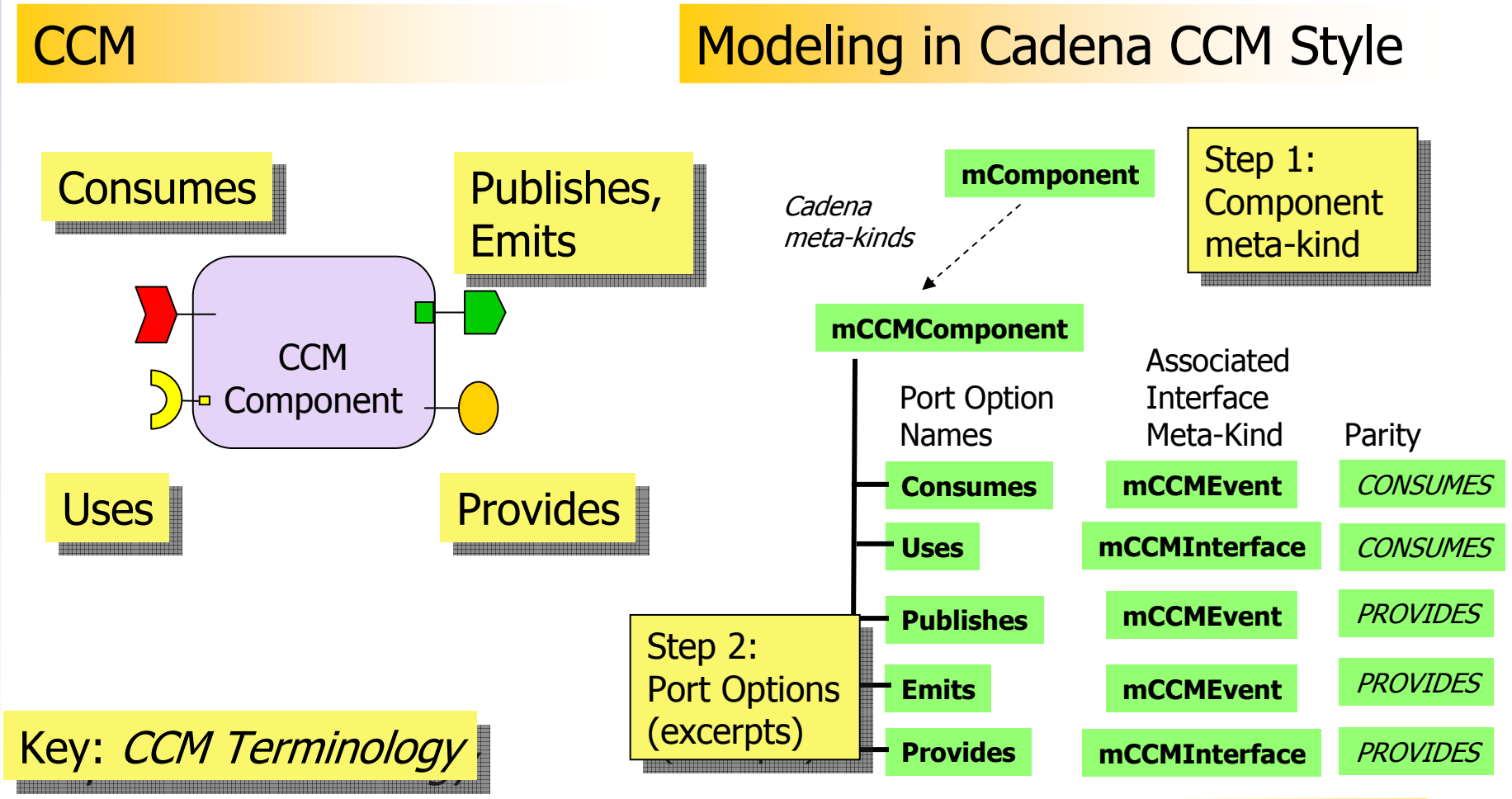
Interface Meta Kind: *<what kind of interfaces can be bound>*
e.g., mCCMEvent

Parity: *<PROVIDES or USES>*
e.g., PROVIDES

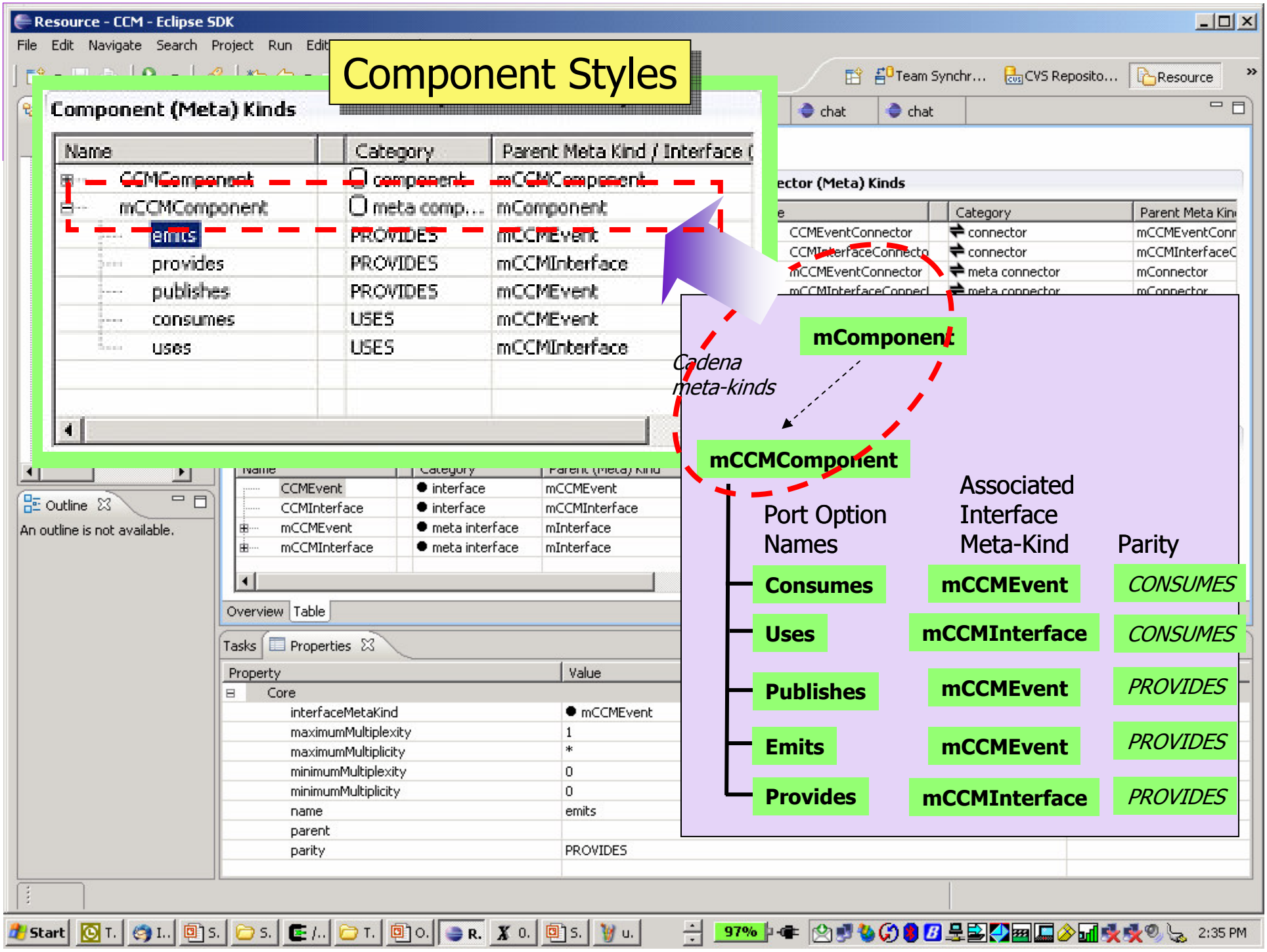
Multiplicity: *<# ports allowed>*
e.g., 0..*

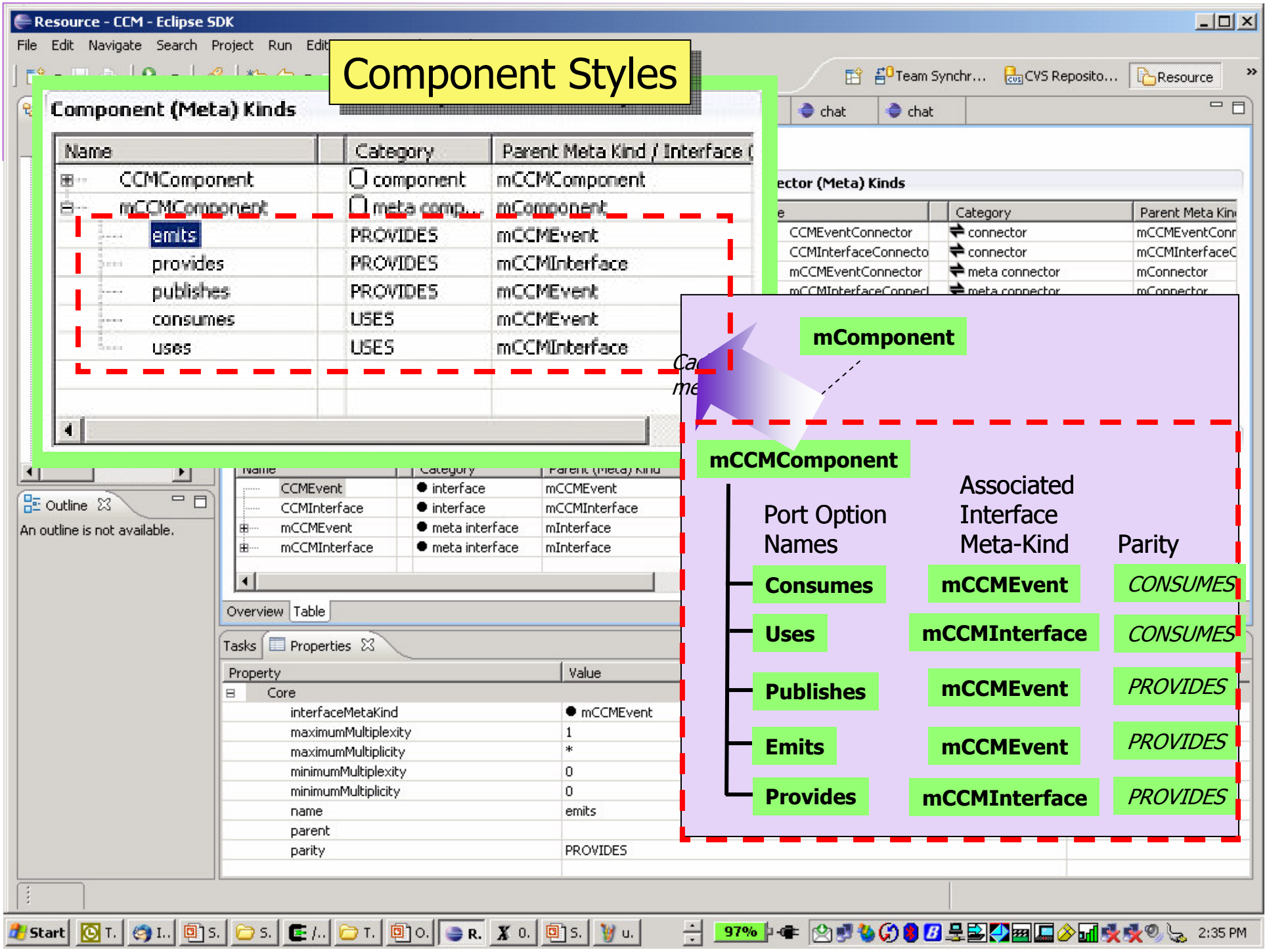
Multiplexity: *<# connections allowed on a particular port>*
e.g., 0..1

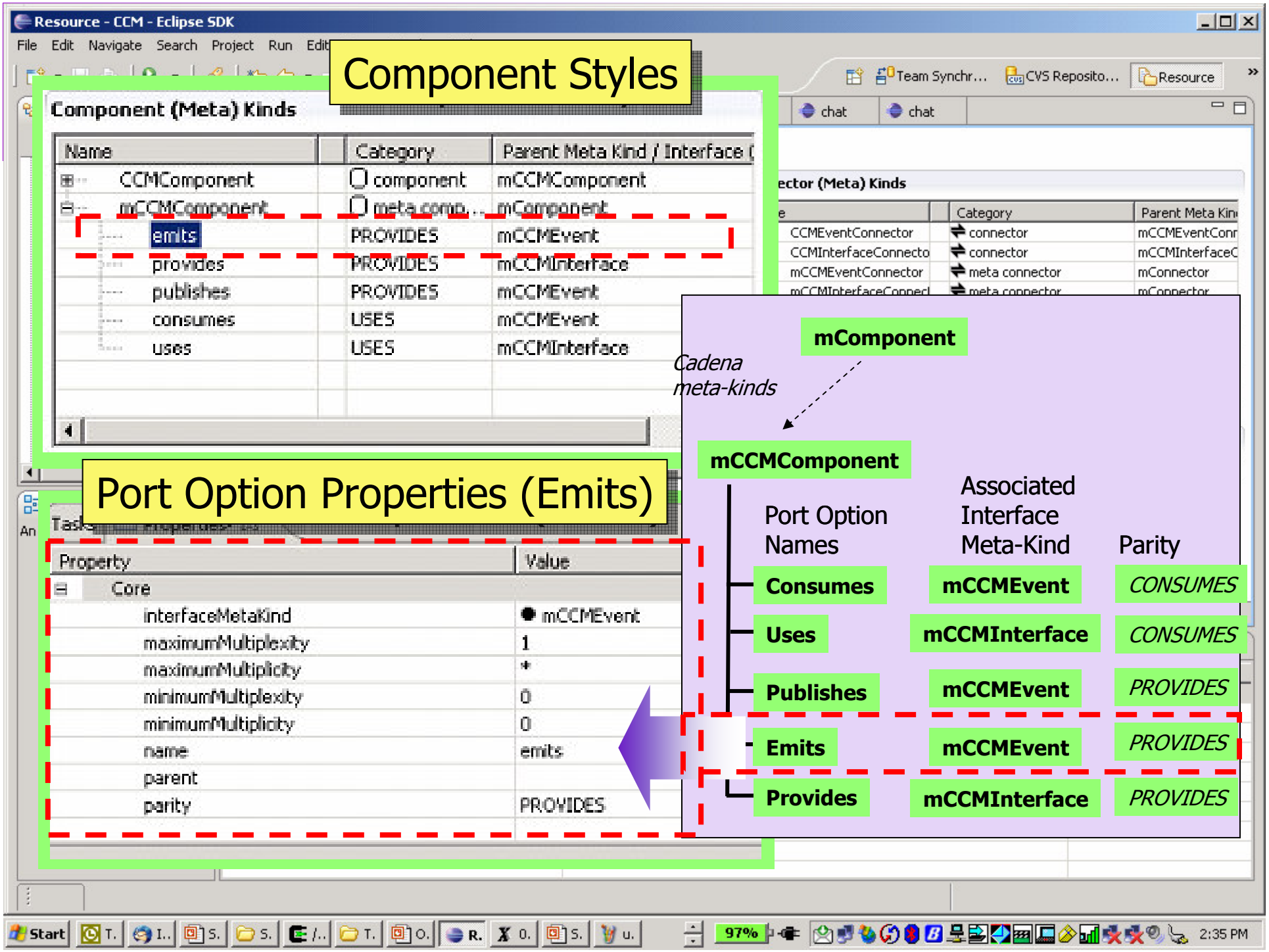
Modeling: Components & Ports



Shapes of a CCM Component Interface

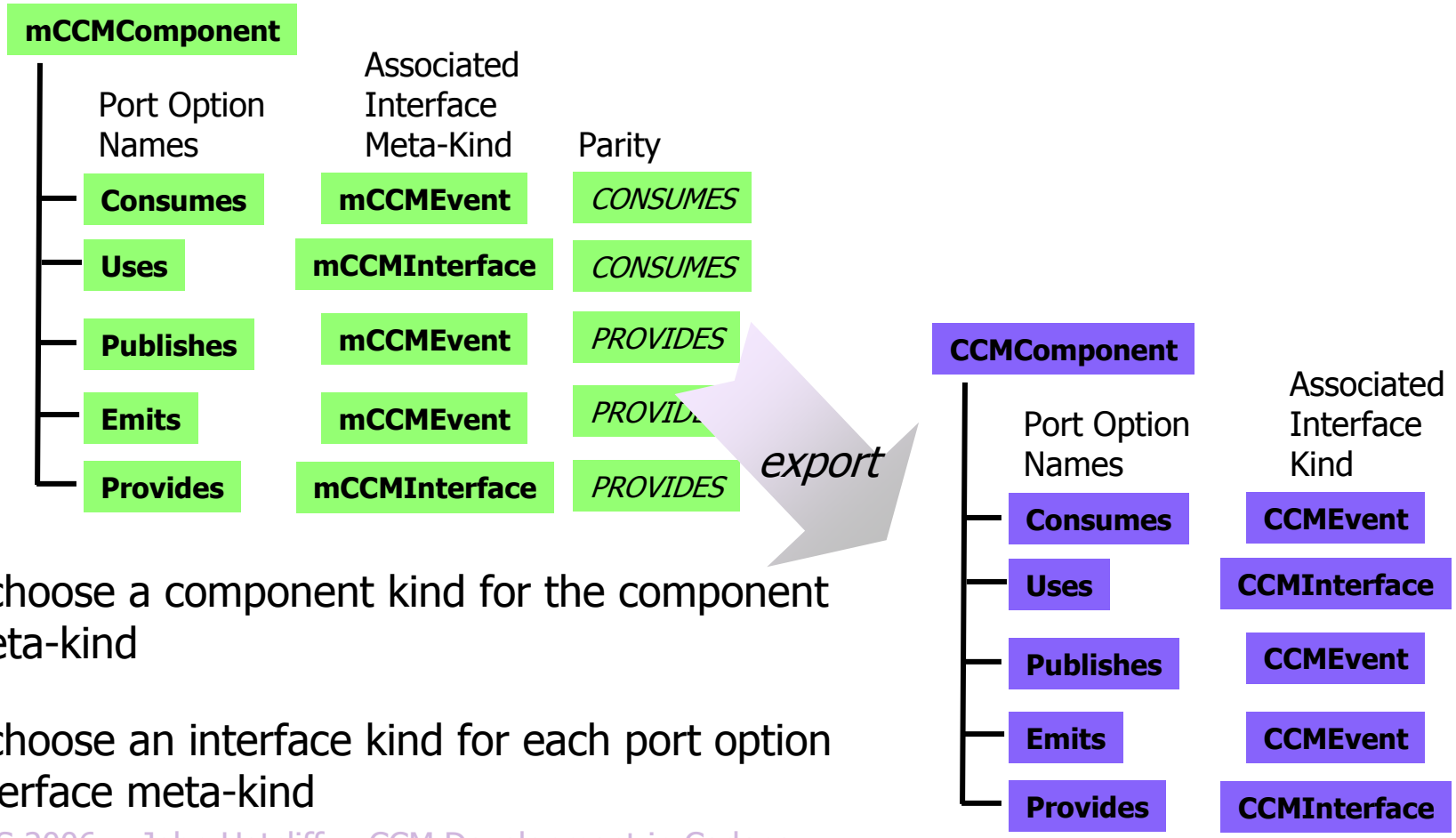

Component Styles
Component (Meta) Kinds
Name
Category
Parent Meta Kind / Interface
CCMComponent
component
mCCMComponent
meta comp...
mComponent
emits
PROVIDES
mCCMEvent
provides
PROVIDES
mCCMInterface
publishes
PROVIDES
mCCMEvent
consumes
USES
mCCMEvent
uses
USES
mCCMInterface
Cadena
meta-kinds
mComponent
mCCMComponent
Port Option
Names
Associated
Interface
Meta-Kind
Parity
Consumes
mCCMEvent
CONSUMES
Uses
mCCMInterface
CONSUMES
Publishes
mCCMEvent
PROVIDES
Emits
mCCMEvent
PROVIDES
Provides
mCCMInterface
PROVIDES


Component Styles
Component (Meta) Kinds
Name
Category
Parent Meta Kind / Interface
CCMComponent
component
mCCMComponent
mCCMComponent
meta comp...
mComponent
emits
PROVIDES
mCCMEvent
provides
PROVIDES
mCCMInterface
publishes
PROVIDES
mCCMEvent
consumes
USES
mCCMEvent
uses
USES
mCCMInterface
mComponent
mCCMComponent
Port Option Names
Associated Interface Meta-Kind
Parity
Consumes
mCCMEvent
CONSUMES
Uses
mCCMInterface
CONSUMES
Publishes
mCCMEvent
PROVIDES
Emits
mCCMEvent
PROVIDES
Provides
mCCMInterface
PROVIDES


Resource - CCM - Eclipse SDK
File Edit Navigate Search Project Run Edit
Component Styles
Component (Meta) Kinds
Name
Category
Parent Meta Kind / Interface
CCMComponent
component
mCCMComponent
mCCMComponent
meta comp...
mComponent
emits
PROVIDES
mCCMEvent
provides
PROVIDES
mCCMInterface
publishes
PROVIDES
mCCMEvent
consumes
USES
mCCMEvent
uses
USES
mCCMInterface
Port Option Properties (Emits)
Property
Value
Core
interfaceMetaKind
mCCMEvent
maximumMultiplexity
1
maximumMultiplicity
*
minimumMultiplexity
0
minimumMultiplicity
0
name
emits
parent
parity
PROVIDES
mComponent
Cadena
meta-kinds
mCCMComponent
Port Option Names
Associated Interface Meta-Kind
Parity
Consumes
mCCMEvent
CONSUMES
Uses
mCCMInterface
CONSUMES
Publishes
mCCMEvent
PROVIDES
Emits
mCCMEvent
PROVIDES
Provides
mCCMInterface
PROVIDES
Team Synchr...
CVS Reposito...
Resource
chat
chat
Start
97%
2:35 PM

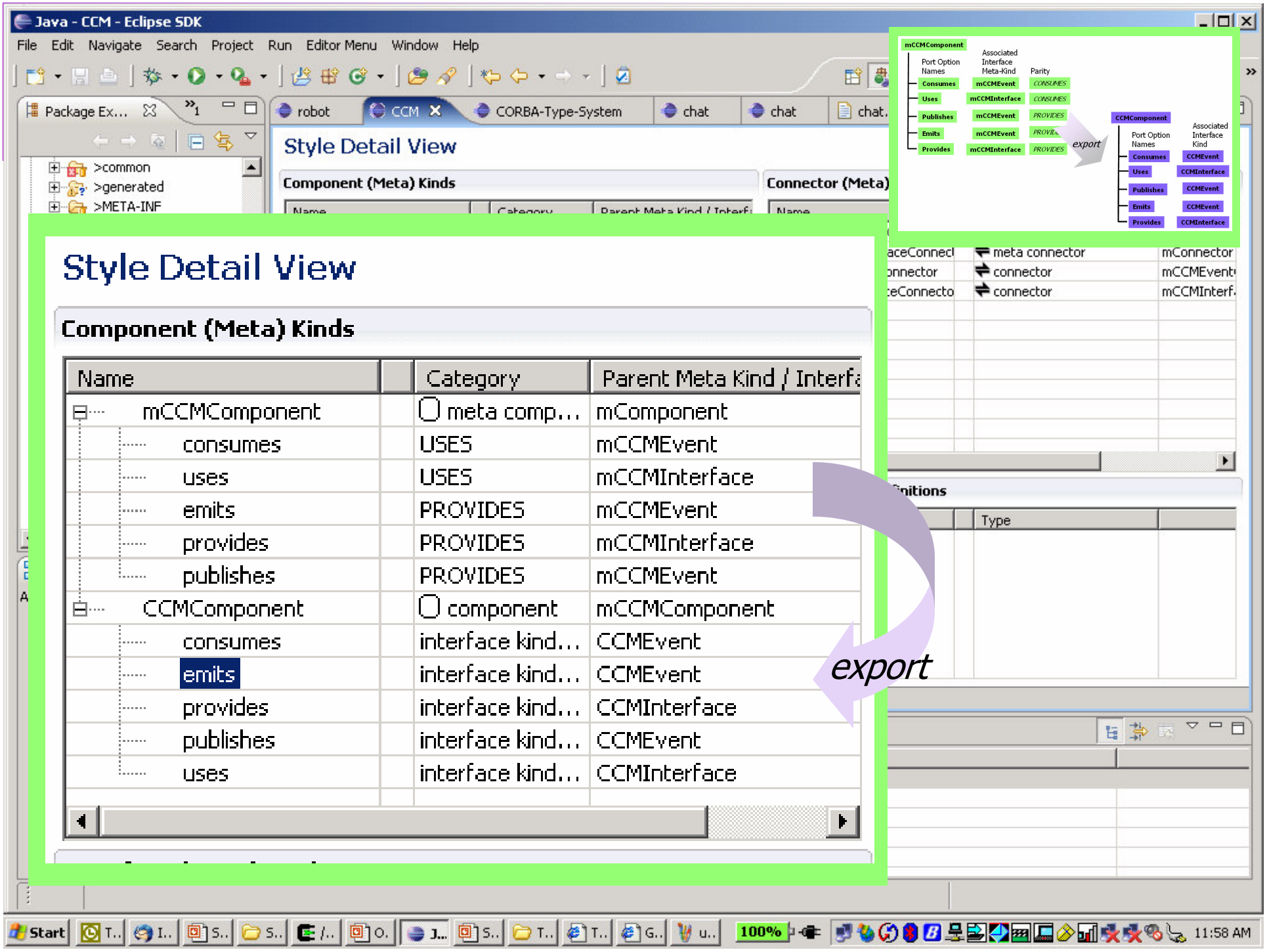
Modeling: Exporting Components

Exporting a *component meta-kind* to a *kind* involves exporting each port option *interface meta-kind* to a *kind.*



...choose a component kind for the component meta-kind

...choose an interface kind for each port option interface meta-kind

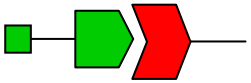

Java - CCM - Eclipse SDK
File Edit Navigate Search Project Run Editor Menu Window Help
Package Ex...
>common
>generated
>META-INF
robot
CCM
CORBA-Type-System
chat
chat
Style Detail View
Component (Meta) Kinds
Connector (Meta)
mCCMComponent
Port Option Names
Associated Interface Meta-Kind
Parity
Consumes
Uses
Publishes
Emits
Provides
mCCMEvent
mCCMInterface
CONSUMES
PROVIDES
export
CCMComponent
Associated Interface Kind
CCMEvent
CCMInterface
meta connector
connector
mConnector
mCCMEvent
mCCMInterf
Type
Style Detail View
Component (Meta) Kinds
Name
Category
Parent Meta Kind / Interfa
mCCMComponent
meta comp...
mComponent
consumes
USES
mCCMEvent
uses
USES
mCCMInterface
emits
PROVIDES
mCCMEvent
provides
PROVIDES
mCCMInterface
publishes
PROVIDES
mCCMEvent
CCMComponent
component
mCCMComponent
consumes
interface kind...
CCMEvent
emits
interface kind...
CCMEvent
provides
interface kind...
CCMInterface
publishes
interface kind...
CCMEvent
uses
interface kind...
CCMInterface
export
Start
100%
11:58 AM

Modeling: Connectors

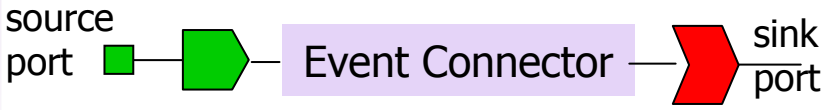
In CCM, event sources connect to event sinks and facets connect to receptacles. We model this by declaring an *event connector* and an *interface connector*

CCM

Event source/sink



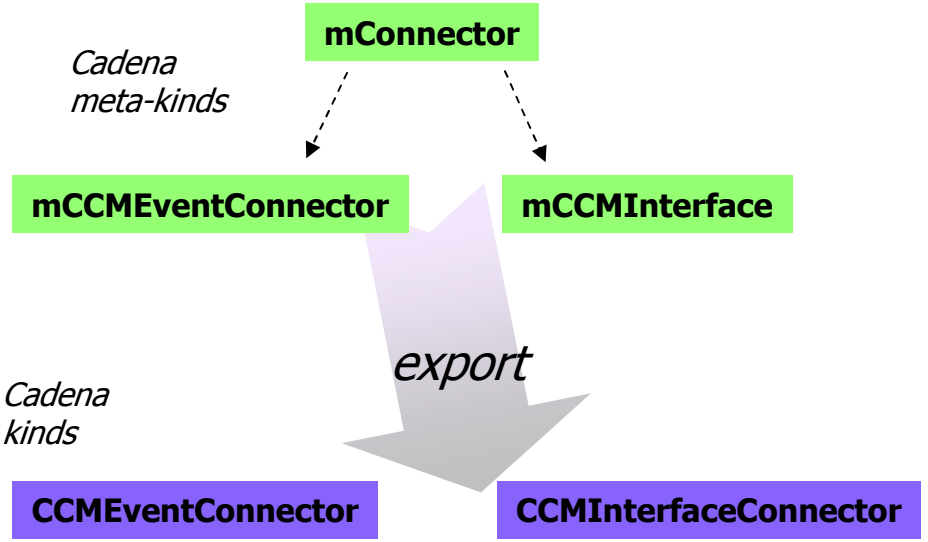
...modeled in Cadena as...



Interface facet/receptacle

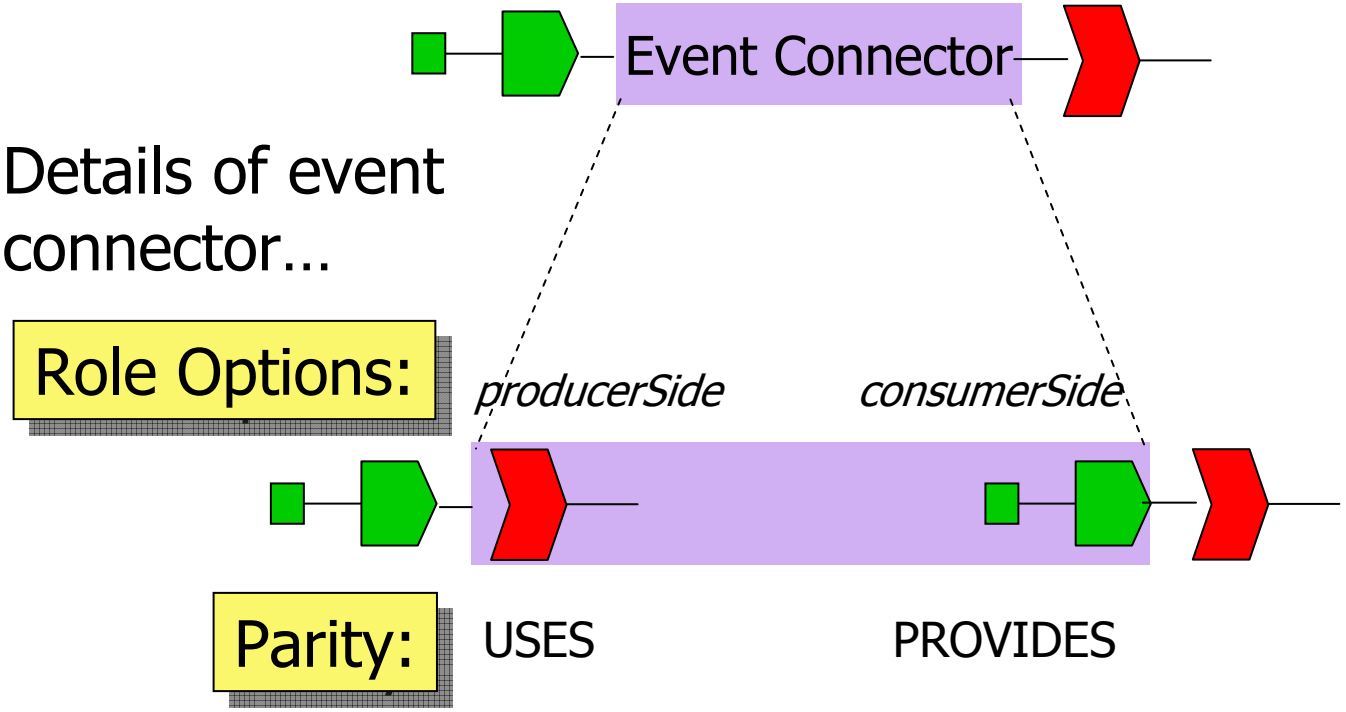
...modeled in Cadena as...

Modeling in Cadena CCM Style



Modeling: Connector with Roles

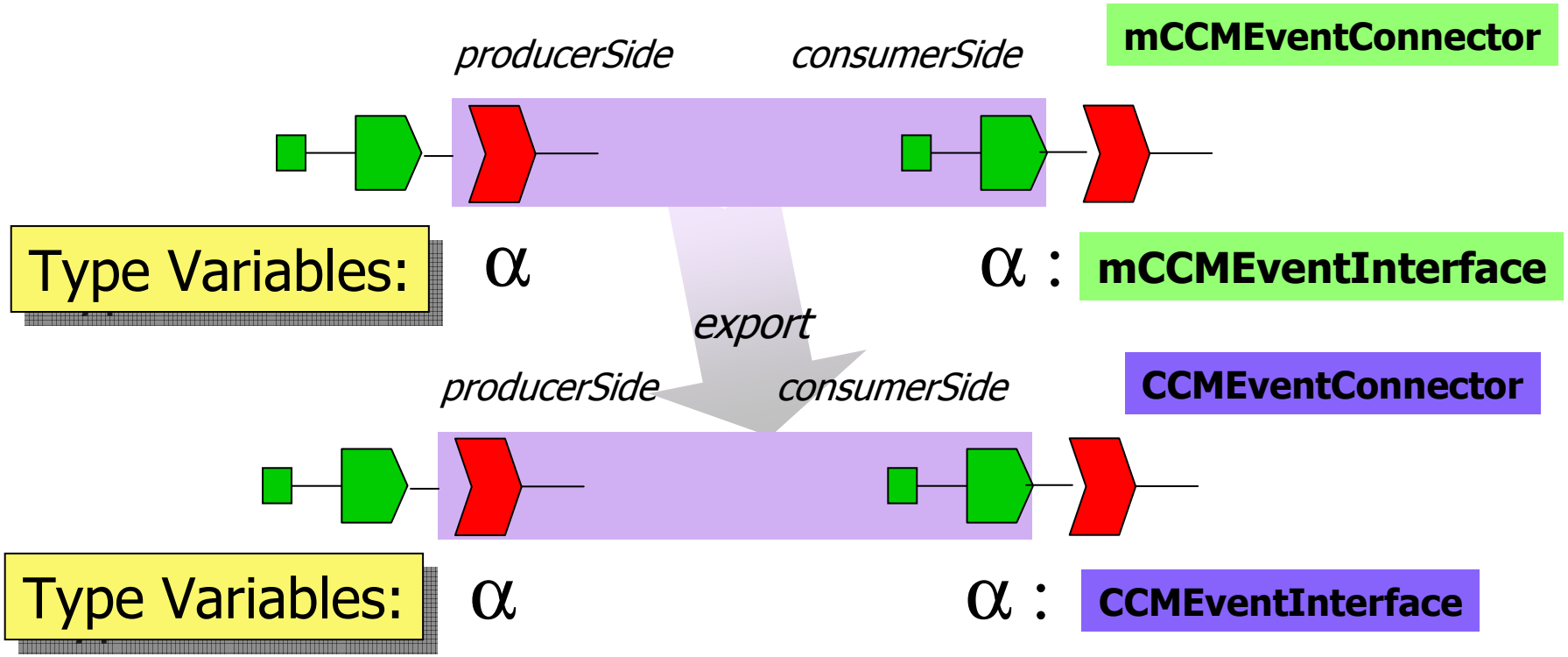
A CCM event connector will connect an event source (producer) to an event sink (consumer)



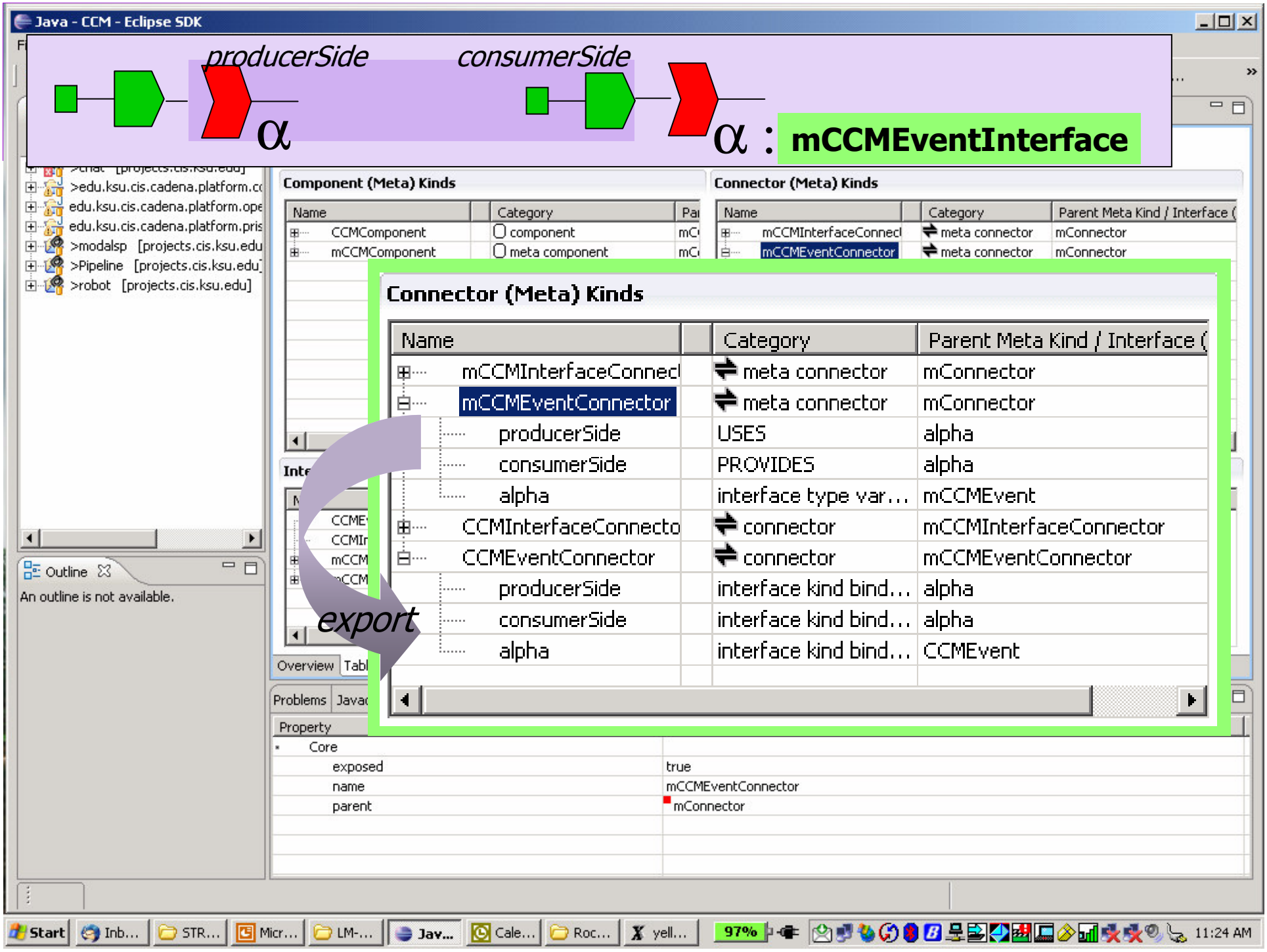
Recall that a Cadena connector has connection points called *roles* (analogous to component *ports)*

Modeling: Connection Constraints

Use *typing capability* to ensure that connections on either side of the connector are compatible.

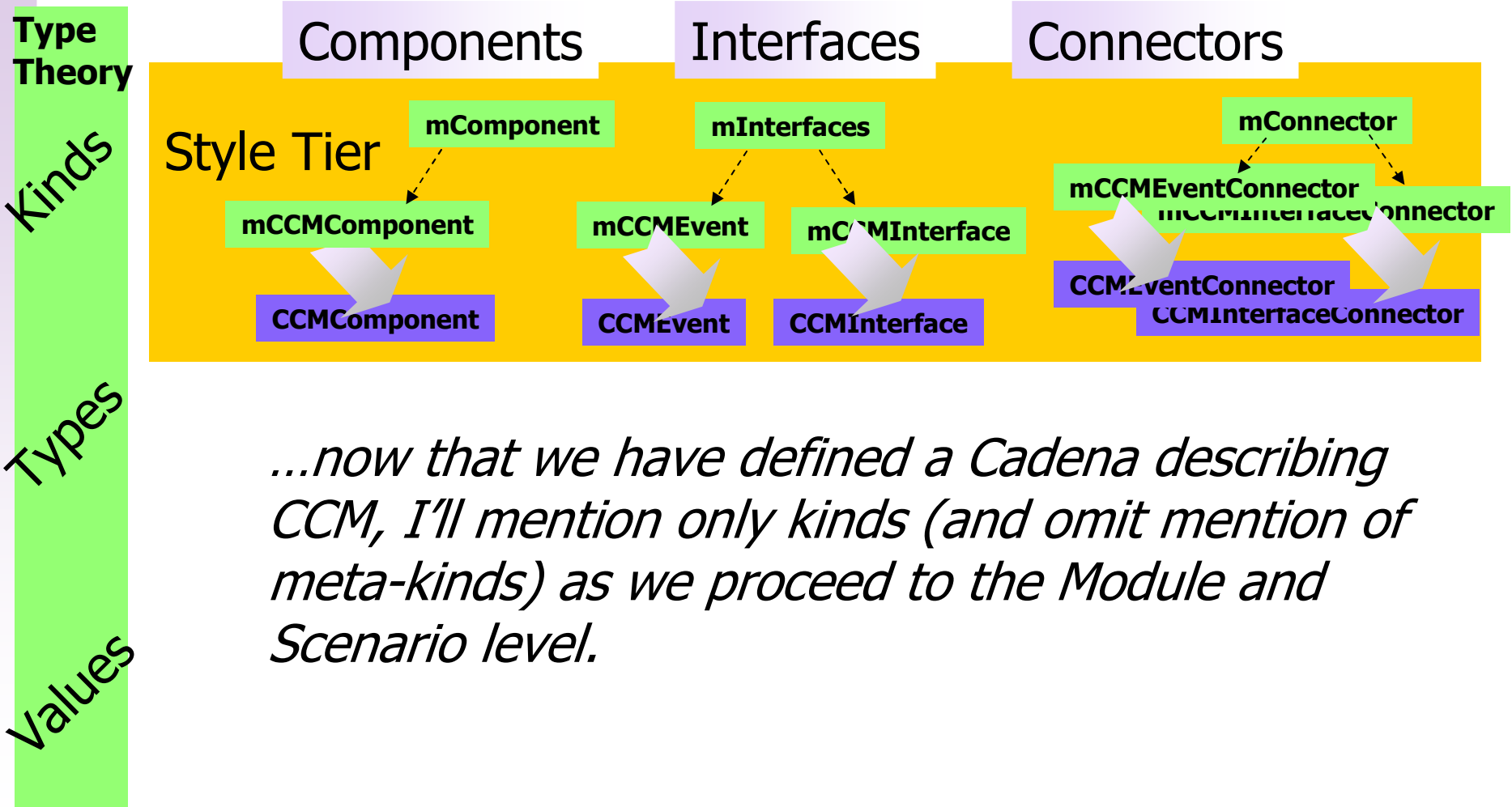


For CCMEventConnector, the typing tells us that the connector can bind on both sides to some CCMEventInterface type T.


producerSide
consumerSide
α
α : mCCMEventInterface
Java - CCM - Eclipse SDK
Component (Meta) Kinds
Name
Category
CCMComponent
component
mCCMComponent
meta component
Connector (Meta) Kinds
Name
Category
Parent Meta Kind / Interface (
mCCMInterfaceConnecl
meta connector
mConnector
mCCMEventConnector
meta connector
mConnector
producerSide
USES
alpha
consumerSide
PROVIDES
alpha
alpha
interface type var...
mCCMEvent
CCMInterfaceConnecto
connector
mCCMInterfaceConnector
CCMEventConnector
connector
mCCMEventConnector
producerSide
interface kind bind...
alpha
consumerSide
interface kind bind...
alpha
alpha
interface kind bind...
CCMEvent
export
Outline
An outline is not available.
Overview
Problems
Property
Core
exposed
true
name
mCCMEventConnector
parent
mConnector
Start
11:24 AM

Declaring Kinds to Define CCM

Define an architectural style that captures CCM



…now that we have defined a Cadena describing CCM, I'll mention only kinds (and omit mention of meta-kinds) as we proceed to the Module and Scenario level.

A CCM Development Environment

Develop plug-ins to create complete environment that supports specific workflows

- Module level plug-in to generate component infrastructure and templates
- Scenario level plug-in to generate deployment and configuration code

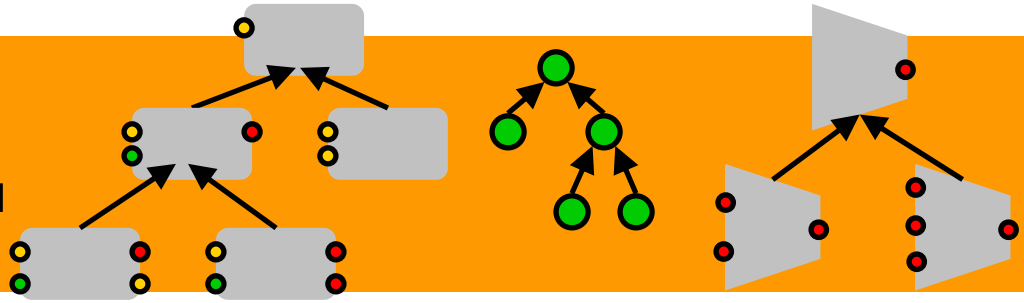
Component Systems Development

Component developer

…builds reusable components, refines general components to product-specific ones

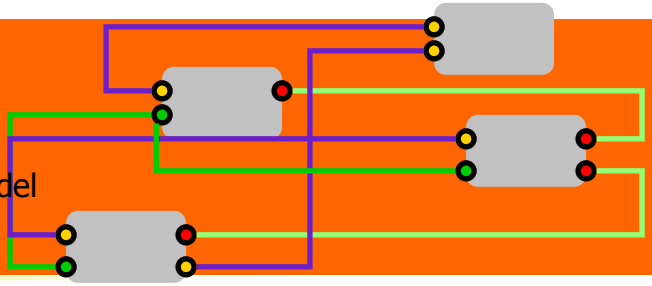
Module Tier

Define Component and Interface Types within an Architecture



Scenario Tier

Allocate component instances, connect, model programming within an architecture



…connects components, configures infrastructure according set strategies, achieves global functional non-functional properties

Component Systems Workflow

Idealized!

Component developer

I. Interface Specification

- Design interfaces
- Design component interfaces

Component integrator

III. Component Integration

- Allocate component instances
- Connect components

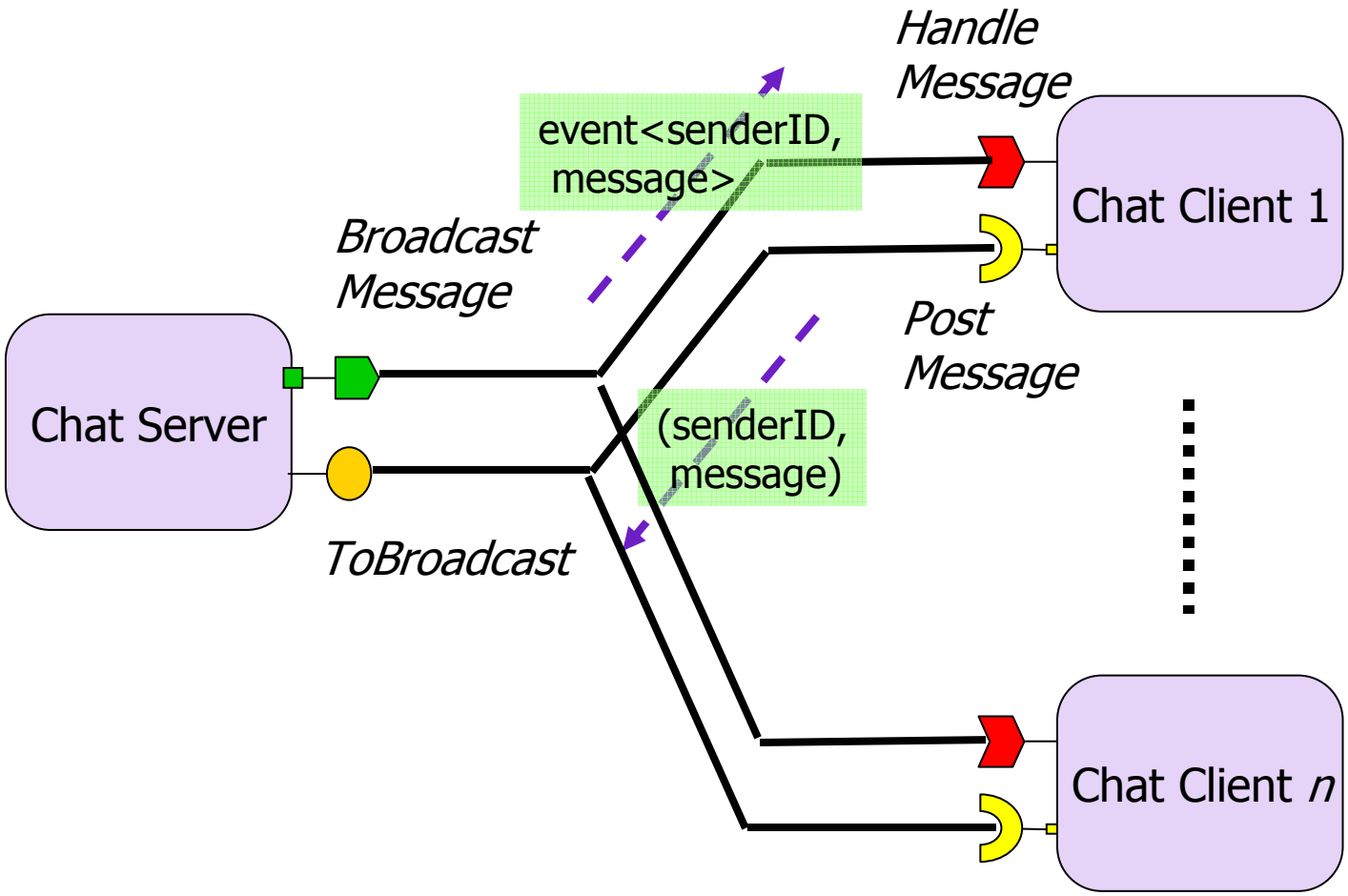
II. Component Implementation

- Generate Interface Definition Language descriptions
- Generate infrastructure code and executer skeletons
- Implement business logic
- Unit level testing

IV. System Deployment

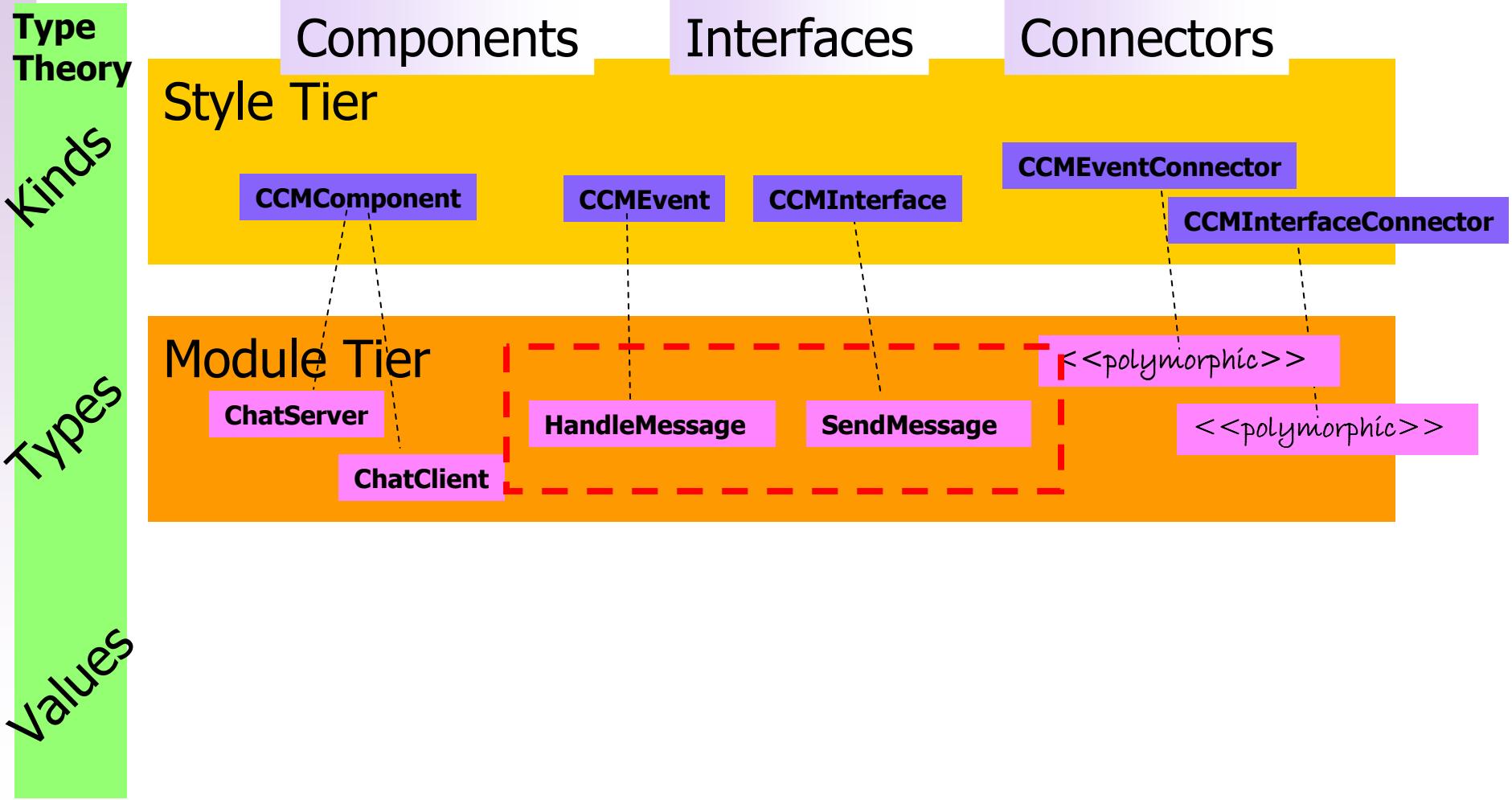
- Generate deployment code
- Execute system

Example: Chat System

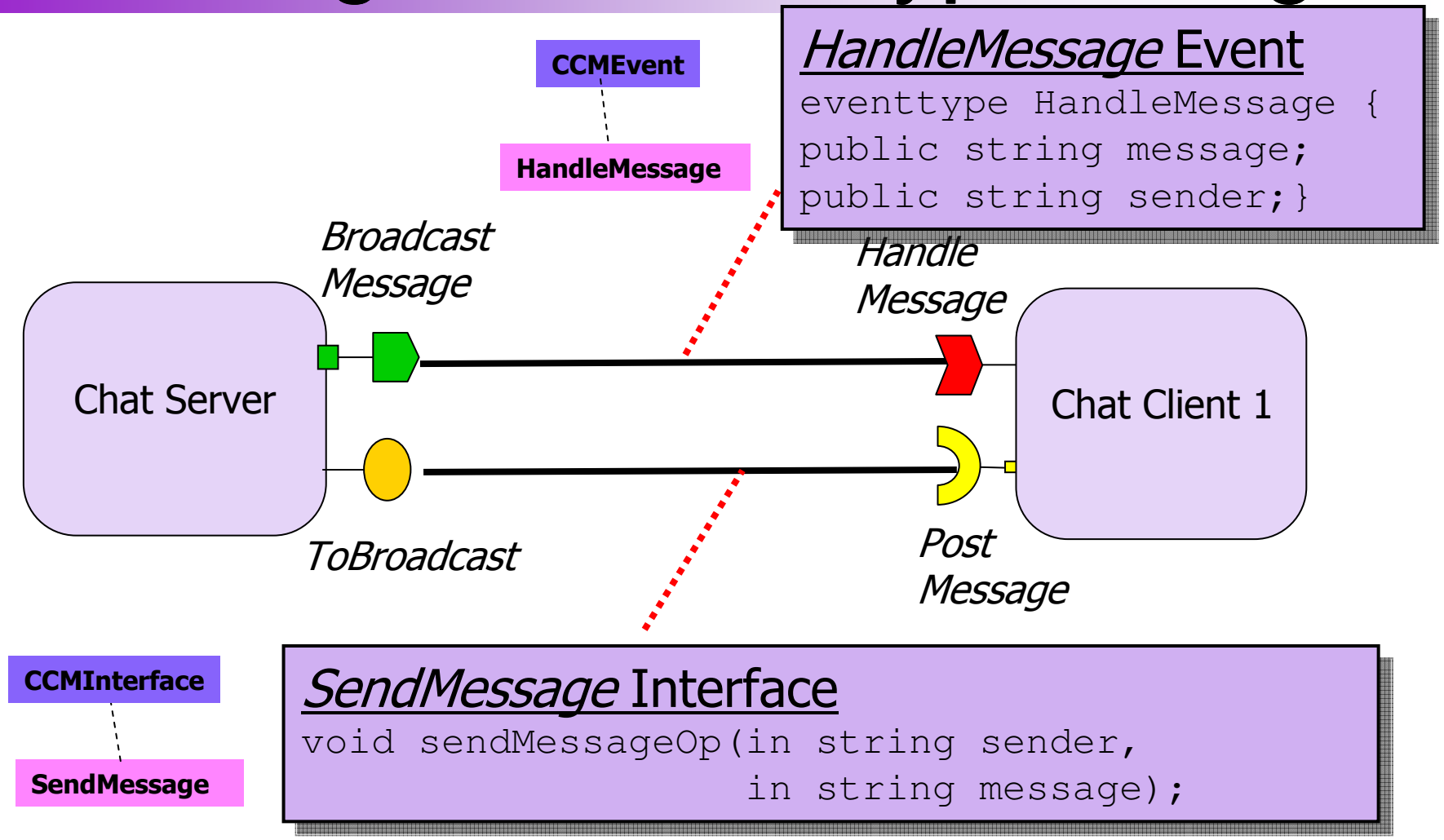


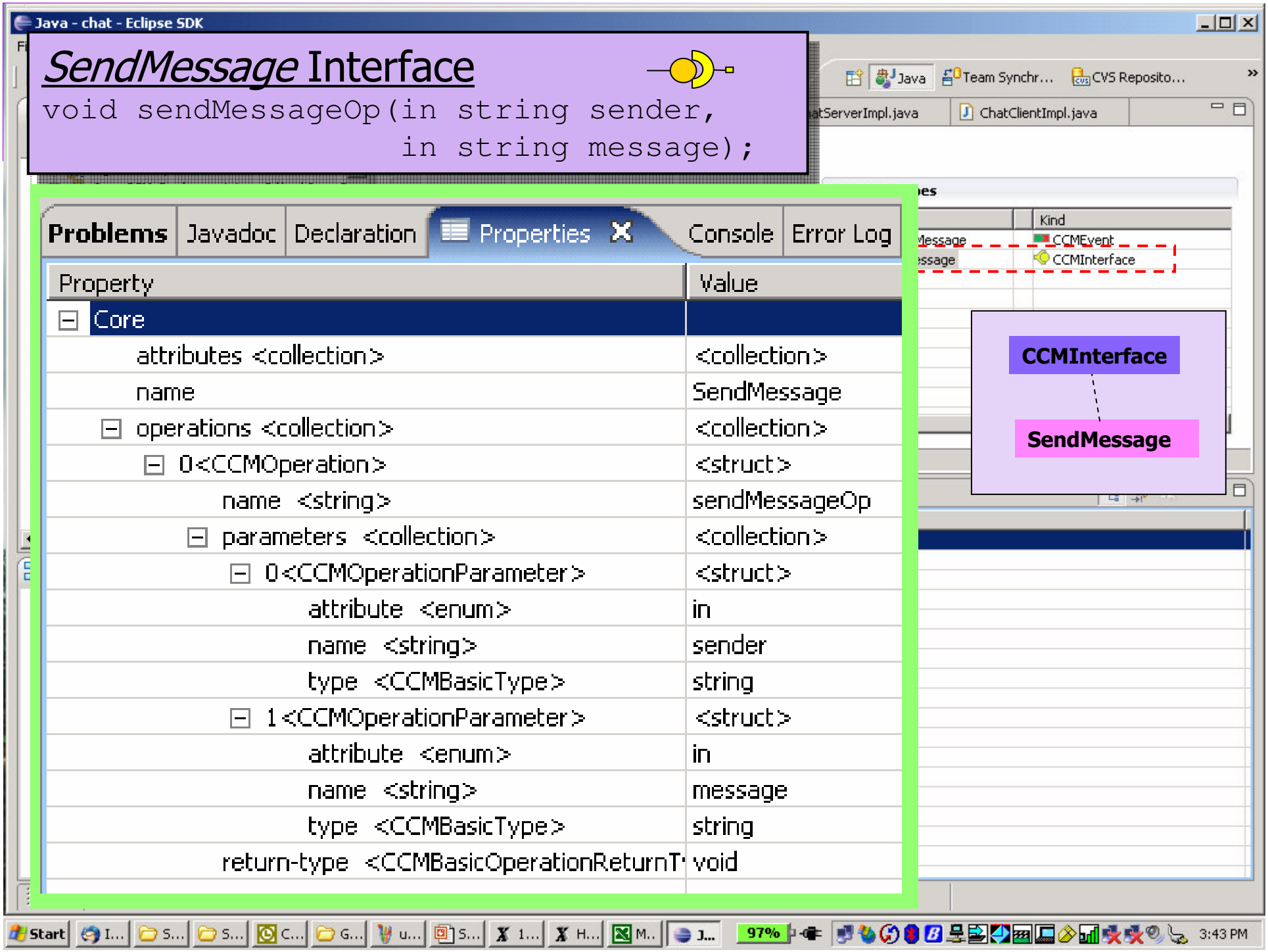
Types from Kinds

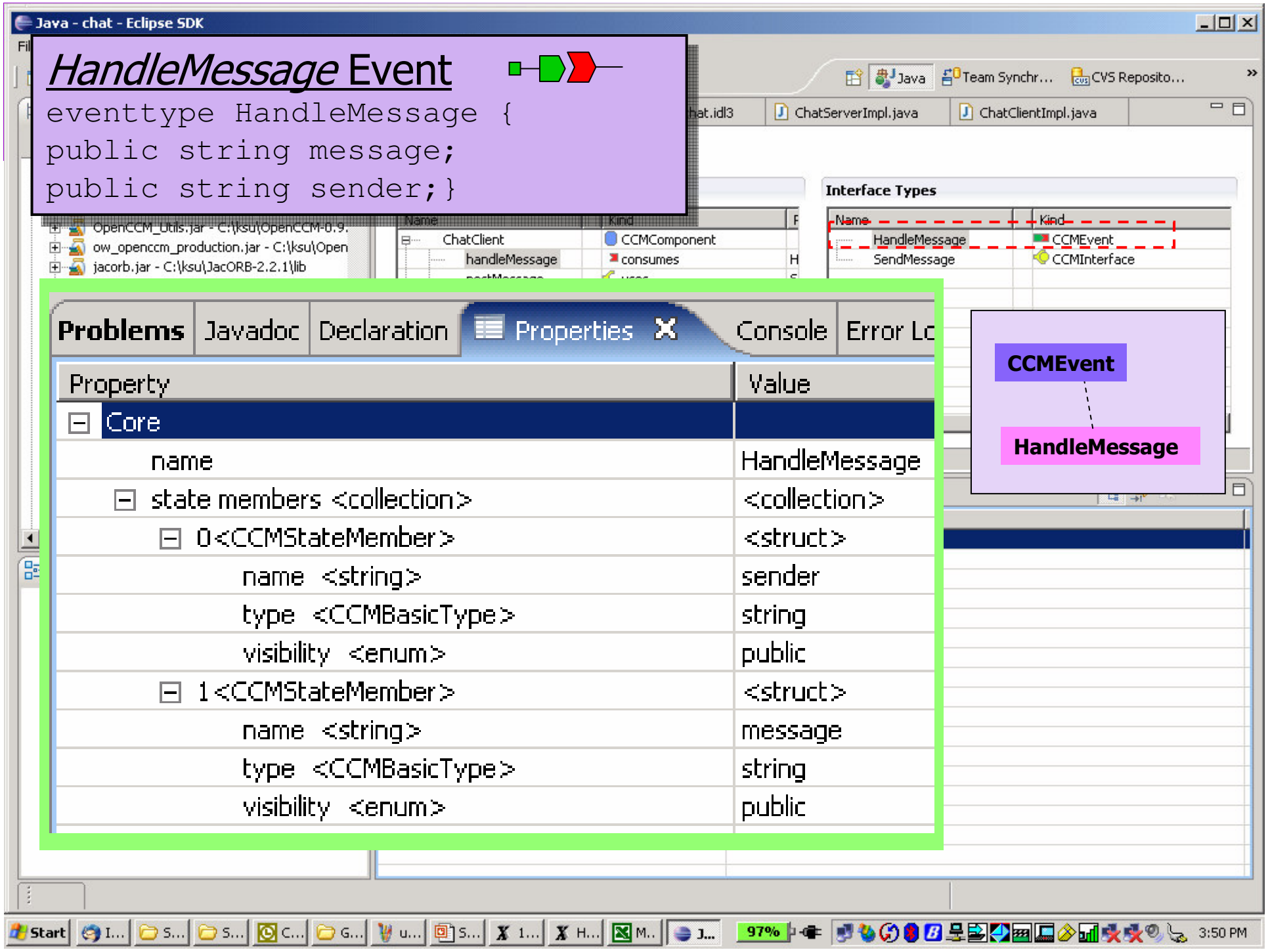
Declaring Types use vocabulary defined by Kinds



Modeling: Interface Type Design

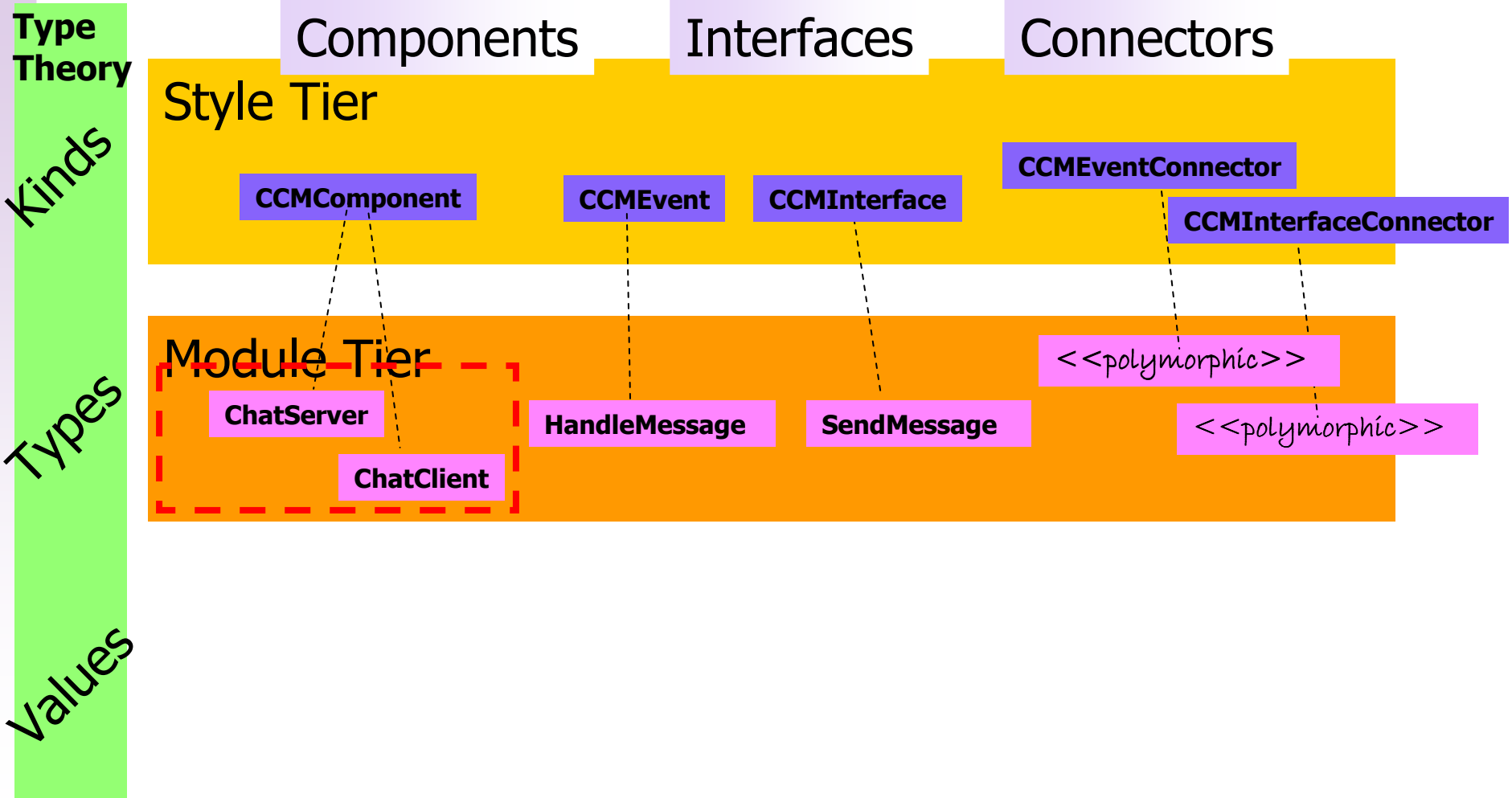



SendMessage Interface
void sendMessageOp(in string sender,
in string message);
Problems
Javadoc
Declaration
Properties
Console
Error Log
Property
Value
Core
attributes <collection>
<collection>
name
SendMessage
operations <collection>
<collection>
0<CCMOperation>
<struct>
name <string>
sendMessageOp
parameters <collection>
<collection>
0<CCMOperationParameter>
<struct>
attribute <enum>
in
name <string>
sender
type <CCMBasicType>
string
1<CCMOperationParameter>
<struct>
attribute <enum>
in
name <string>
message
type <CCMBasicType>
string
return-type <CCMBasicOperationReturnT
void
Java
Team Synchr...
CVS Reposito...
ChatClientImpl.java
Kind
CCMEvent
CCMInterface
CCMInterface
SendMessage
Start
97%
3:43 PM

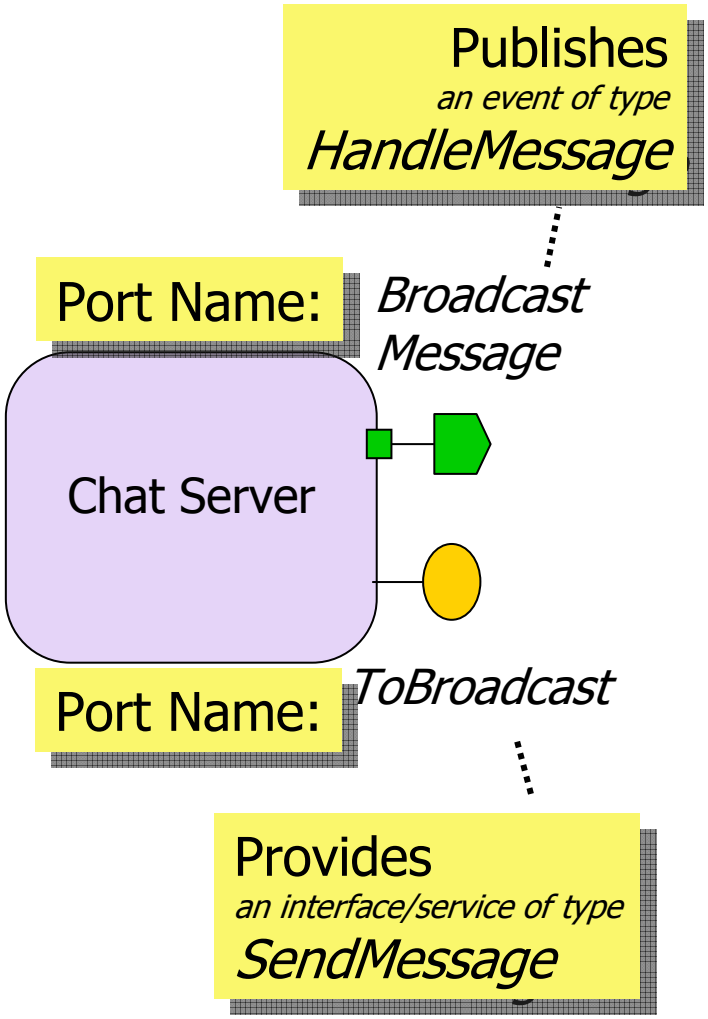

HandleMessage Event
eventtype HandleMessage {
public string message;
public string sender;}
Java - chat - Eclipse SDK
Interface Types
Name
Kind
HandleMessage
CCMEvent
SendMessage
CCMInterface
ChatClient
CCMComponent
handleMessage
consumes
Problems
Javadoc
Declaration
Properties
Console
Error Lc
Property
Value
Core
name
HandleMessage
state members <collection>
<collection>
0<CCMStateMember>
<struct>
name <string>
sender
type <CCMBasicType>
string
visibility <enum>
public
1<CCMStateMember>
<struct>
name <string>
message
type <CCMBasicType>
string
visibility <enum>
public
CCMEvent
HandleMessage
Start
97%
3:50 PM

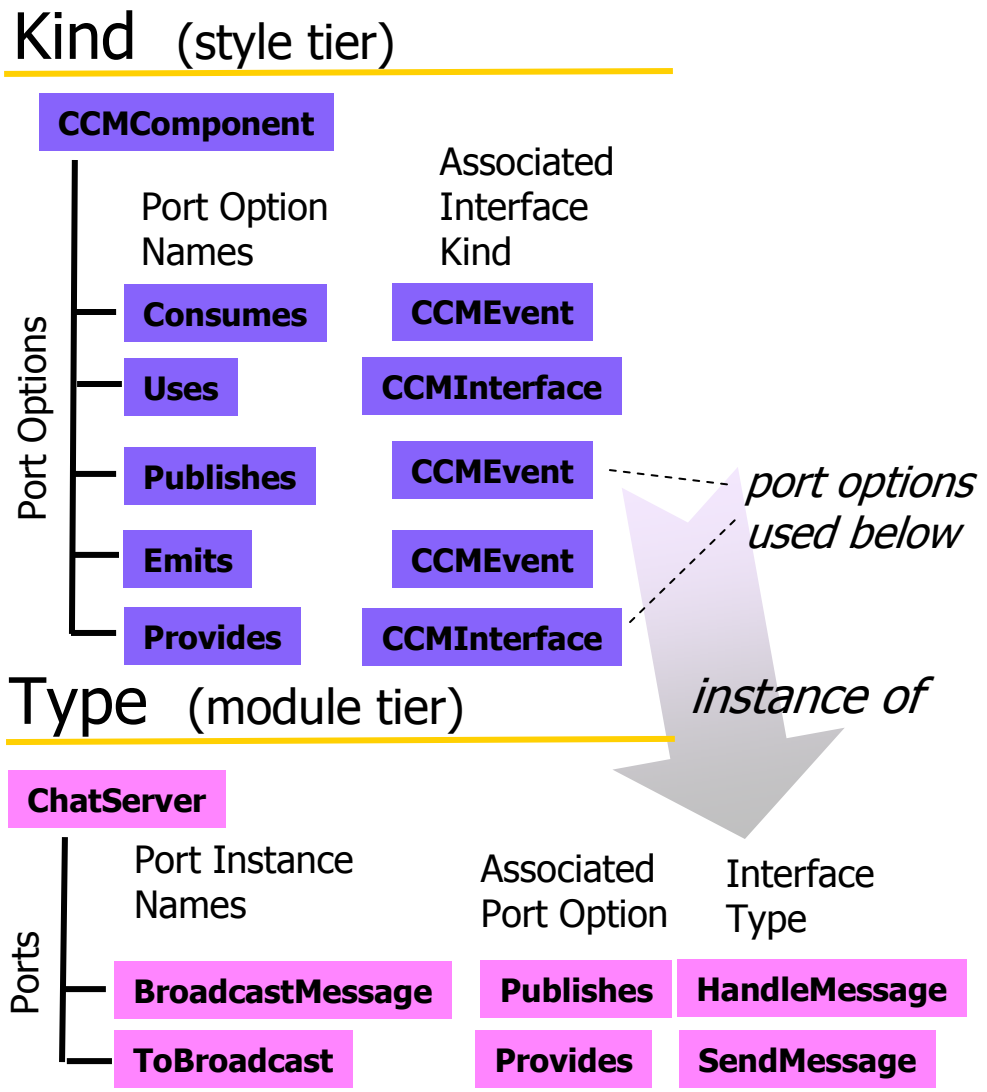
Types from Kinds

Declaring Types use vocabulary defined by Kinds

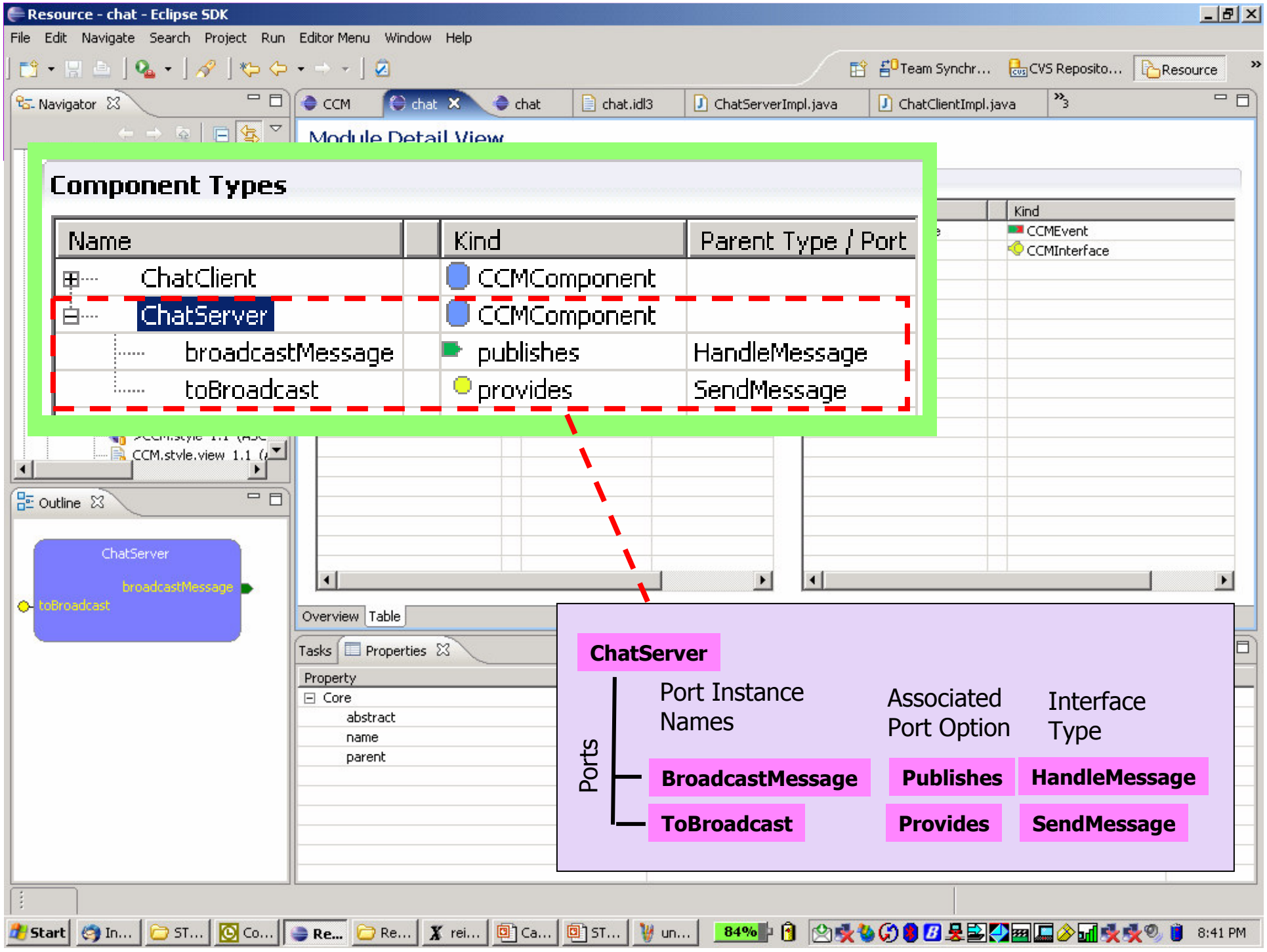


Modeling: Chat Server Component

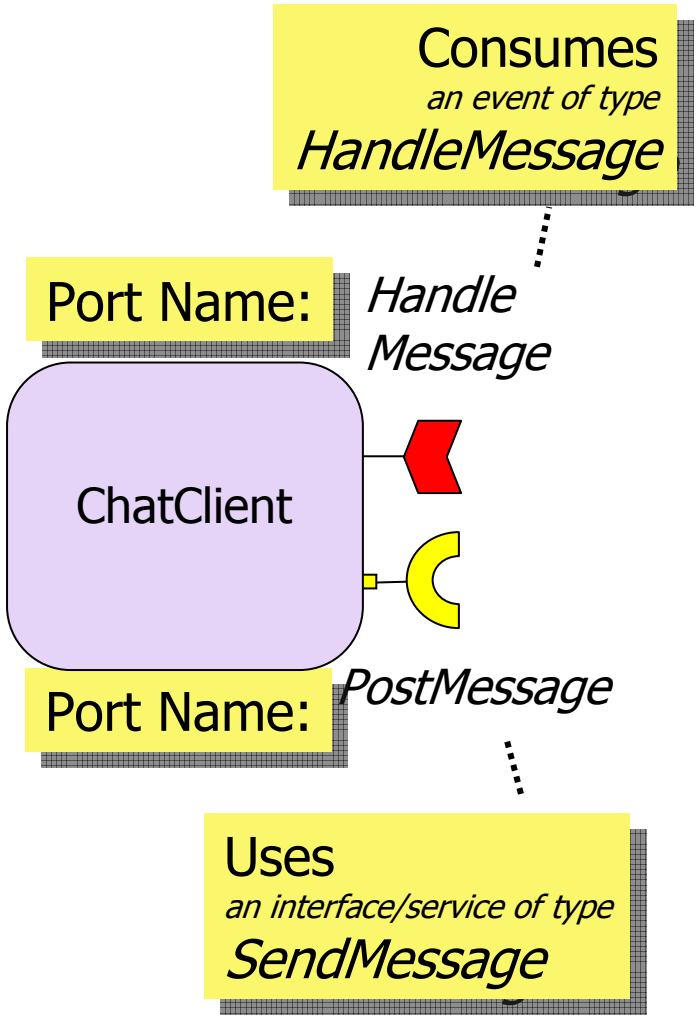


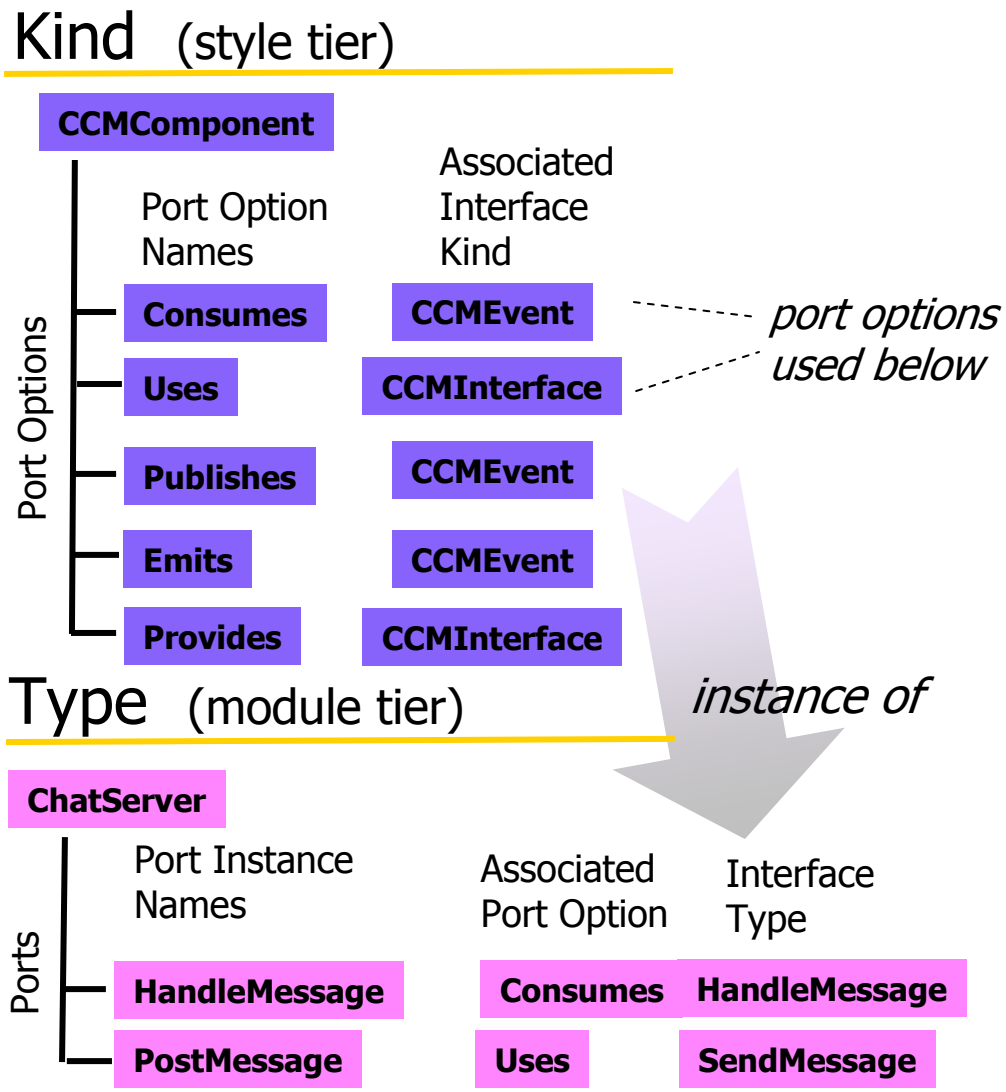


...with multiplicity constraints satisfied

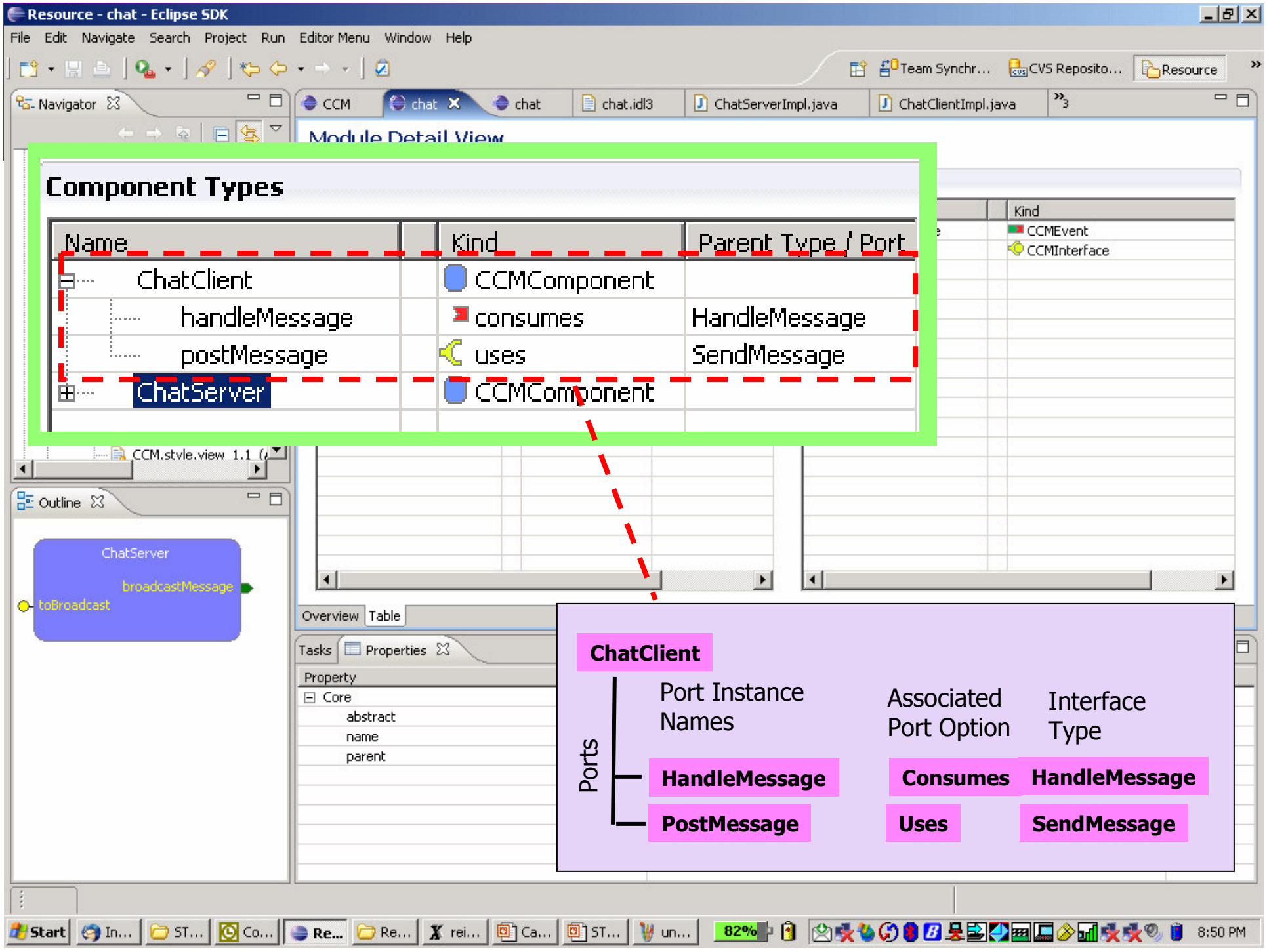

Resource - chat - Eclipse SDK
File Edit Navigate Search Project Run Editor Menu Window Help
Navigator
CCM
chat
chat.idl3
ChatServerImpl.java
ChatClientImpl.java
Module Detail View
Component Types
Name
Kind
Parent Type / Port
ChatClient
CCMComponent
ChatServer
CCMComponent
broadcastMessage
publishes
HandleMessage
toBroadcast
provides
SendMessage
Kind
CCMEvent
CCMInterface
CCM.style.view 1.1
Outline
ChatServer
broadcastMessage
toBroadcast
Overview
Table
Tasks
Properties
Property
Core
abstract
name
parent
ChatServer
Ports
Port Instance Names
Associated Port Option
Interface Type
BroadcastMessage
Publishes
HandleMessage
ToBroadcast
Provides
SendMessage
Start
84%
8:41 PM

Modeling: Chat Client Component



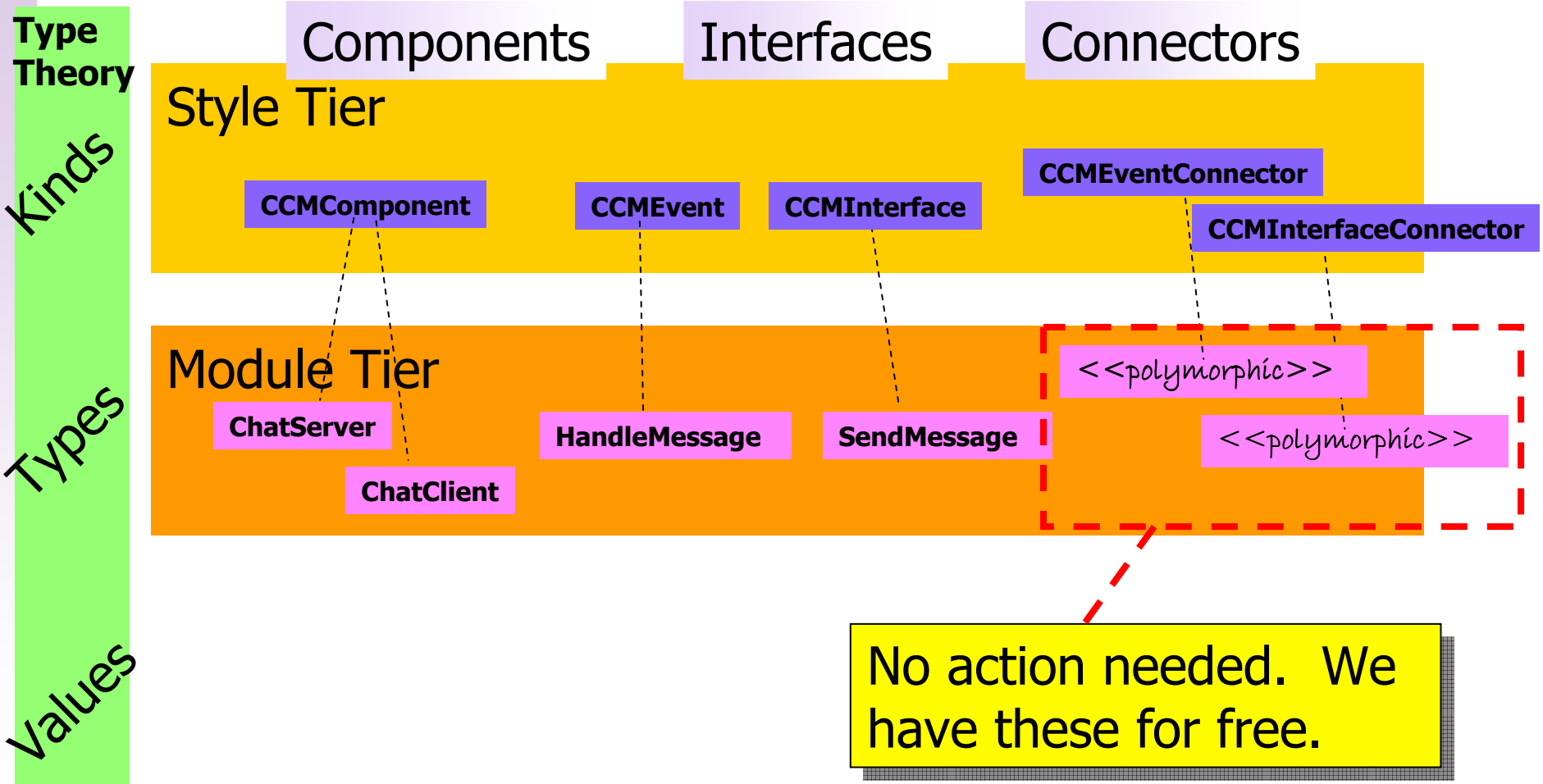


...with multiplicity constraints satisfied


Resource - chat - Eclipse SDK
File Edit Navigate Search Project Run Editor Menu Window Help
Navigator
CCM
chat
chat.idl3
ChatServerImpl.java
ChatClientImpl.java
Module Detail View
Component Types
Name
Kind
Parent Type / Port
ChatClient
CCMComponent
handleMessage
consumes
HandleMessage
postMessage
uses
SendMessage
ChatServer
CCMComponent
Kind
CCMEvent
CCMInterface
Outline
ChatServer
broadcastMessage
toBroadcast
Overview
Table
Tasks
Properties
Property
Core
abstract
name
parent
ChatClient
Ports
Port Instance Names
Associated Port Option
Interface Type
HandleMessage
Consumes
HandleMessage
PostMessage
Uses
SendMessage
Start
82%
8:50 PM

Types from Kinds

Declaring Types use vocabulary defined by Kinds



Component Systems Workflow

Idealized!

Component developer

I. Interface Specification

- Design interfaces
- Design component interfaces

Component integrator

III. Component Integration

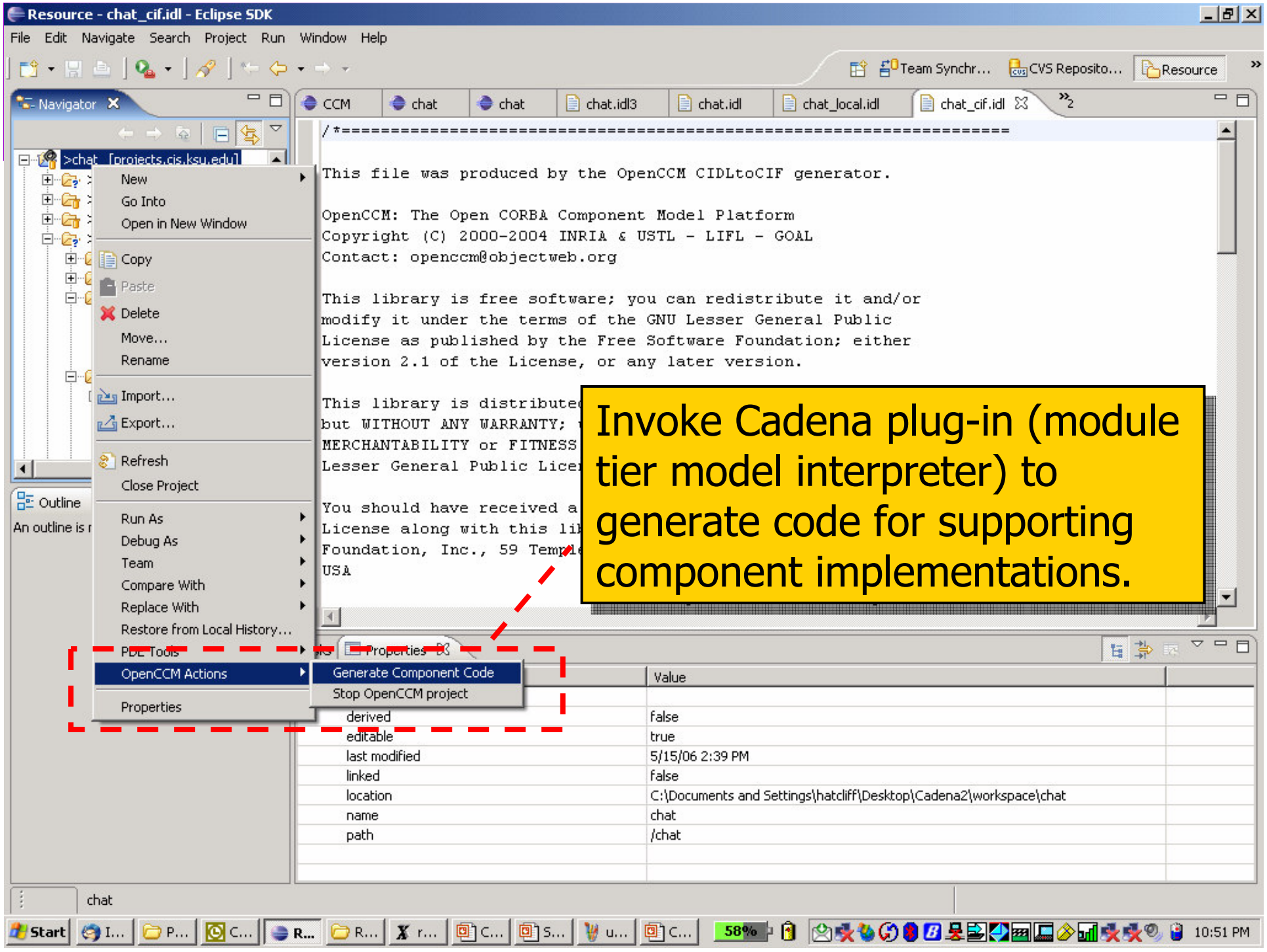
- Allocate component instances
- Connect components

II. Component Implementation

- Generate Interface Definition Language descriptions
- Generate infrastructure code and executer skeletons
- Implement business logic
- Unit level testing

IV. System Deployment

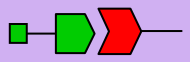
- Generate deployment code
- Execute system


Resource - chat_cif.idl - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Navigator
CCM
chat
chat
chat.idl3
chat.idl
chat_local.idl
chat_cif.idl
/*==========
This file was produced by the OpenCCM CIDLtoCIF generator.
OpenCCM: The Open CORBA Component Model Platform
Copyright (C) 2000-2004 INRIA & USTL - LIFL - GOAL
Contact: openccm@objectweb.org
This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or any later version.
New
Go Into
Open in New Window
Copy
Paste
Delete
Move...
Rename
Import...
Export...
Refresh
Close Project
Run As
Debug As
Team
Compare With
Replace With
Restore from Local History...
PDE Tools
OpenCCM Actions
Properties
Generate Component Code
Stop OpenCCM project
Invoke Cadena plug-in (module tier model interpreter) to generate code for supporting component implementations.
Outline
Properties
Value
derived
false
editable
true
last modified
5/15/06 2:39 PM
linked
false
location
C:\Documents and Settings\hatcliff\Desktop\Cadena2\workspace\chat
name
chat
path
/chat
chat
Start
58%
10:51 PM

Generate IDL Descriptions

Component developer executes Cadena plug-in to generate interface descriptions written in CCM IDL.

HandleMessage Event



Problems | Javadoc | Declaration | Properties | Console | Error Lc

Property	Value
Core	
name	HandleMessage
state members <collection>	<collection>
0<CCMStateMember>	<struct>
name <string>	sender
type <CCMBasicType>	string
visibility <enum>	public
1<CCMStateMember>	<struct>
name <string>	message
type <CCMBasicType>	string
visibility <enum>	public

auto-generate

chat.idl3

```
module chat {
typeprefix chat
"ccm.objectweb.org";
eventtype HandleMessage {
public string message;
public string sender;
};
};
```

Generate IDL Descriptions

Component developer executes Cadena plug-in to generate interface descriptions written in CCM IDL.

SendMessage Interface

Property	Value
Core	
attributes <collection>	<collection>
name	SendMessage
operations <collection>	<collection>
0<CCMOperation>	<struct>
name <string>	sendMessageOp
parameters <collection>	<collection>
0<CCMOperationParameter>	<struct>
attribute <enum>	in
name <string>	
type <CCMBasicType>	
1<CCMOperationParameter>	
attribute <enum>	
name <string>	
type <CCMBasicType>	
return-type <CCMBasicOperationReturnT	void

auto-generate

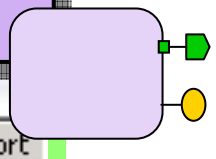
chat.idl3

```
module chat {
typeprefix chat
"ccm.objectweb.org";
interface SendMessage {
  void sendMessageOp(
    in string sender,
    in string message);
};};
```

Generate IDL Descriptions

Component developer executes Cadena plug-in to generate interface descriptions written in CCM IDL.

ChatServer Component Interface



Component Types

Name	Kind	Parent Type / Port
ChatClient	CCMComponent	
ChatServer	CCMComponent	
broadcastMessage	publishes	HandleMessage
toBroadcast	provides	SendMessage

auto-generate

chat.idl3

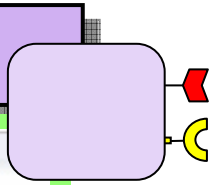
```
module chat {
typeprefix chat "ccm.objectweb.org";
component ChatServer {
  provides SendMessage toBroadcast;
  publishes HandleMessage broadcastMessage;
};

home ChatServerHome
      manages ChatServer {
};
};
```

Generate IDL Descriptions

Component developer executes Cadena plug-in to generate interface descriptions written in CCM IDL.

ChatClient Component Interface



Component Types

Name	Kind	Parent Type / Port
ChatClient	CCMComponent	
handleMessage	consumes	HandleMessage
postMessage	uses	SendMessage
ChatServer	CCMComponent	

auto-generate

chat.idl3

```
module chat {
typeprefix chat "ccm.objectweb.org";
component ChatClient {
  uses SendMessage postMessage;
  consumes HandleMessage handleMessage;
};

home ChatClientHome manages ChatClient {
};
```

Component Systems Workflow

Idealized!

Component developer

I. Interface Specification

- Design interfaces
- Design component interfaces

Component integrator

III. Component Integration

- Allocate component instances
- Connect components

II. Component Implementation

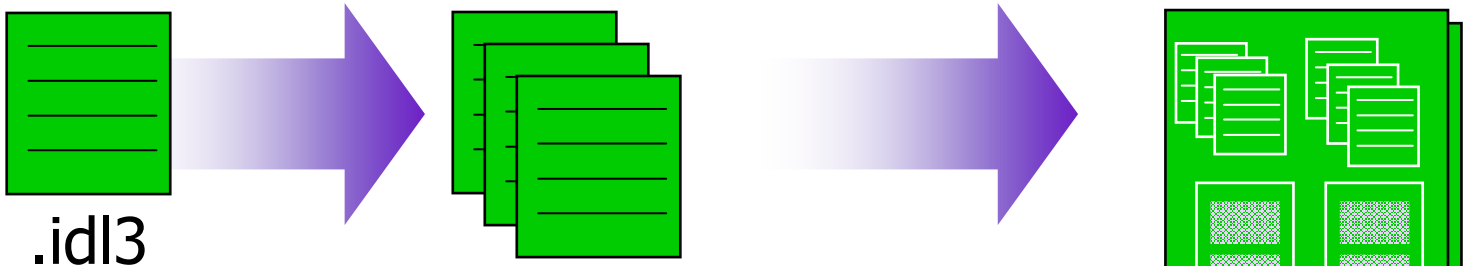
- Generate Interface Definition Language descriptions
- Generate infrastructure code and executer skeletons
- Implement business logic
- Unit level testing

IV. System Deployment

- Generate deployment code
- Execute system

Generate Code from IDL3

Underlying CCM IDL3/IDL compiler is called to generate stubs/skeletons (CORBA infrastructure) and executor (business logic) template.



.idl3
(component level IDL)

chat/chat.idl3

.idl2
(component level IDL)

chat/generated/idl/chat.idl
chat/generated/idl/chat_local.idl
chat/generated/ild/chat_cif.idl

.java

Infrastructure Code
chat/generated/skeletons
chat/generated/stubs

Business Logic Templates
chat/generated/src

Component Systems Workflow

Idealized!

Component developer

I. Interface Specification

- Design interfaces
- Design component interfaces

Component integrator

III. Component Integration

- Allocate component instances
- Connect components

II. Component Implementation

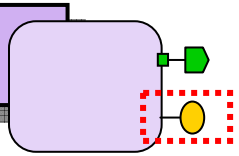
- Generate Interface Definition Language descriptions
- Generate infrastructure code and executer skeletons
- Implement business logic
- Unit level testing

IV. System Deployment

- Generate deployment code
- Execute system

Implement Component Business Logic

ChatServer Component



```
public void sendMessageOp(java.lang.String sender,
                          java.lang.String message) {

   System.err.println("<" + sender + "> : " + message);

   HandleMessage outgoingMessage = new HandleMessageImpl();
                outgoingMessage.sender = sender;
                outgoingMessage.message = message;

   get_context().push_broadcastMessage(outgoingMessage);
}
```

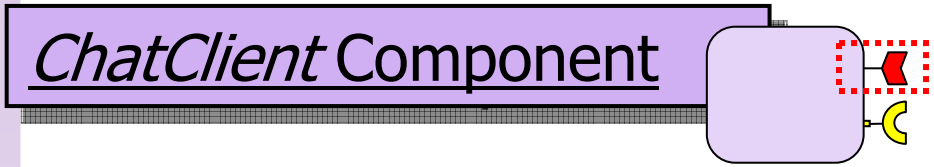
Allocate message structure

Assign values to message slots

Publish message on `BroadcastMessage` port

- `sendMessageOp` is method on the `ToBroadcast` port (called by clients to broadcast a message

Implement Component Business Logic



Receive the message event

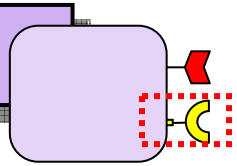
```
public void push(org.objectweb.ccm.chat.HandleMessage event) {
  // Display the new message in the main message area
  mainMessageArea.append("<" + event.sender + ">  "
                                    + event.message + "\n");
}
```

Extract fields and send to display

- `push` is the event handler method on the on the `EventHandler` Consumes port (called by server to notify clients of a message)

Implement Component Business Logic

ChatClient Component



```
public void actionPerformed(ActionEvent arg0) {
  // Send a new message to the server
  SendMessage sendMessage =
      get_context().get_connection_postMessage();
  sendMessage.sendMessageOp(
      nameText.getText(),
      newMessageText.getText());
  newMessageText.setText("");
}
```

Get the handle for the `postMessage` port

Call `sendMessageOp` method on `postMessage` port

Clear text in composition frame of GUI window

- `actionPerformed` is invoked by SWING GUI code when the "send" button is pressed in the chat GUI window.

…note ChatClient business logic also includes about 20 lines of SWING code to implement GUI window.

Component Systems Workflow

Idealized!

Component developer

I. Interface Specification

- Design interfaces
- Design component interfaces

Component integrator

III. Component Integration

- Allocate component instances
- Connect components

II. Component Implementation

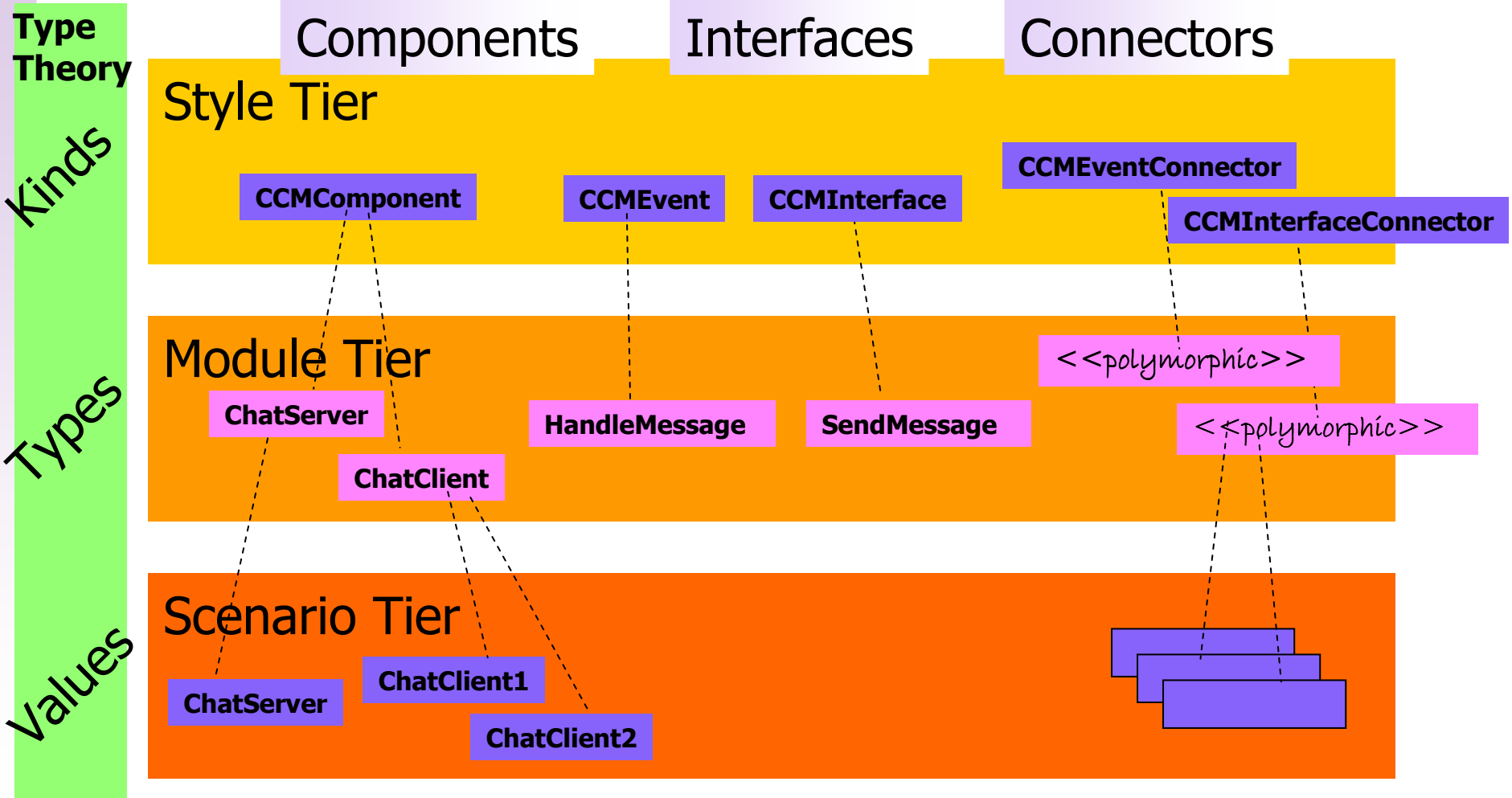
- Generate Interface Definition Language descriptions
- Generate infrastructure code and executer skeletons
- Implement business logic
- Unit level testing

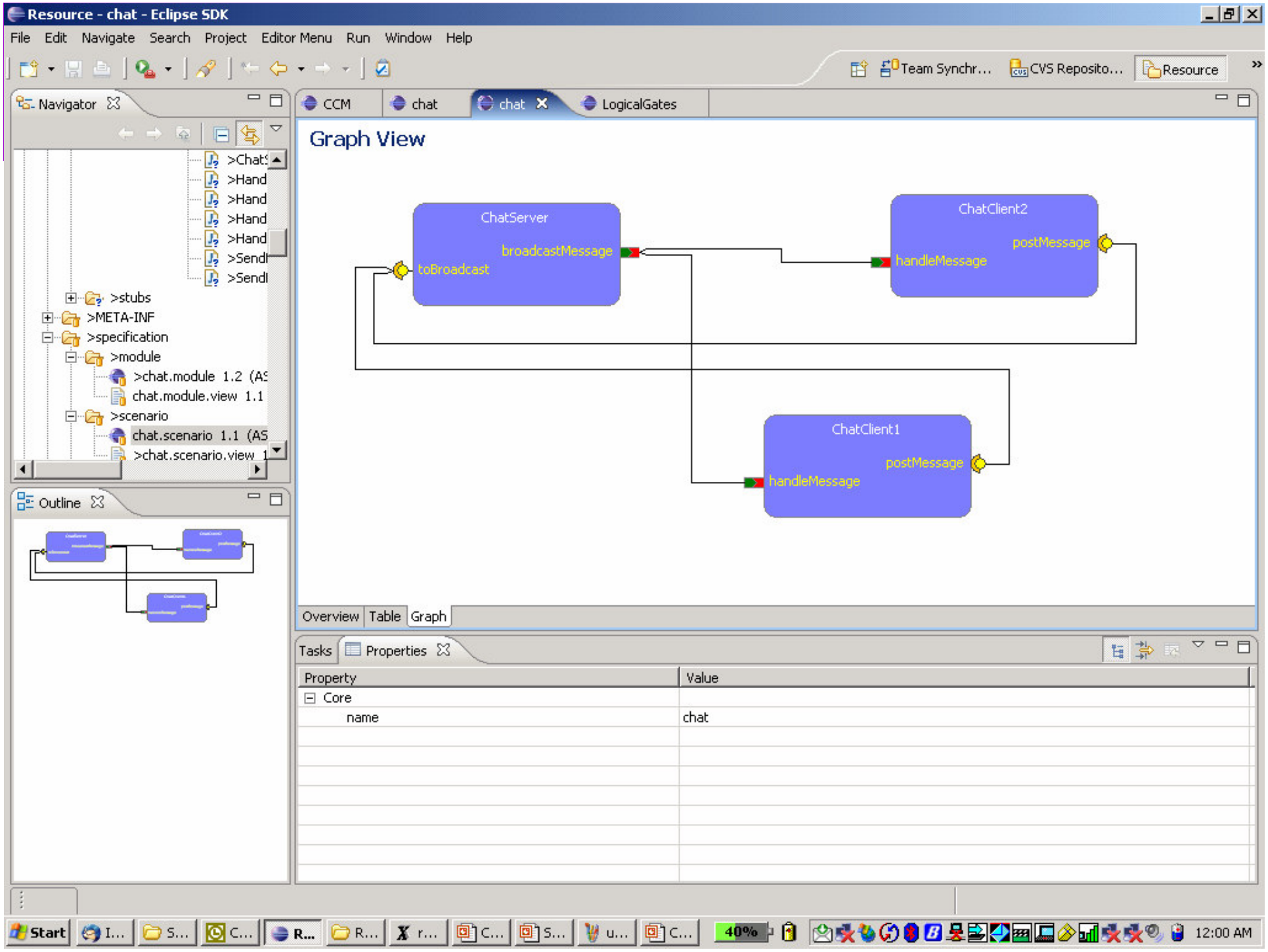
IV. System Deployment

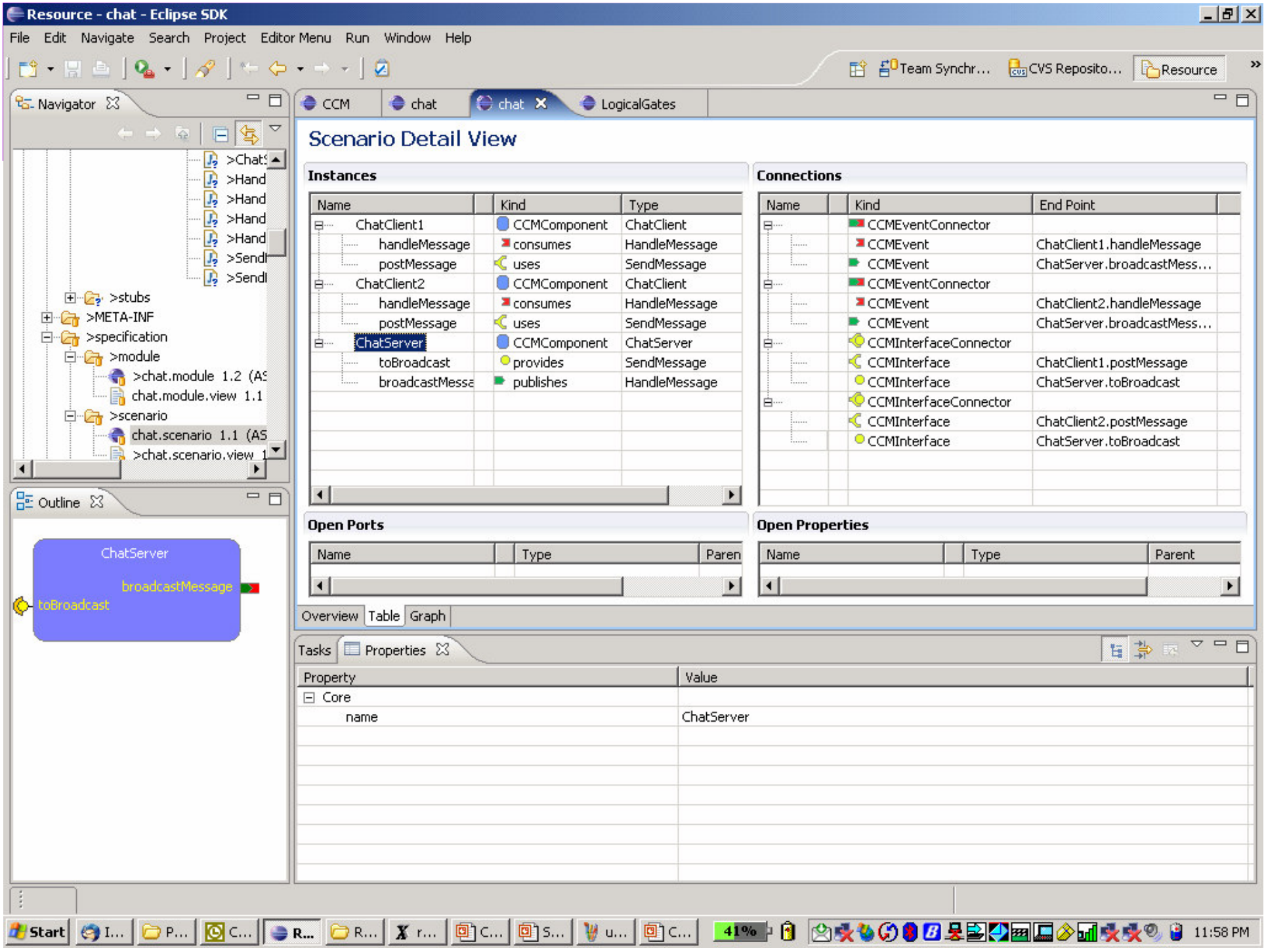
- Generate deployment code
- Execute system

Types from Kinds

Declaring Types use vocabulary defined by Kinds




Resource - chat - Eclipse SDK
File Edit Navigate Search Project Editor Menu Run Window Help
Team Synchr...
CVS Reposito...
Resource
Navigator
>stubs
>META-INF
>specification
>module
>chat.module 1.2 (AS
chat.module.view 1.1
>scenario
chat.scenario 1.1 (AS
>chat.scenario.view 1
Outline
CCM
chat
chat
LogicalGates
Graph View
ChatServer
broadcastMessage
toBroadcast
ChatClient2
postMessage
handleMessage
ChatClient1
postMessage
handleMessage
Overview
Table
Graph
Tasks
Properties
Property
Value
Core
name
chat
Start
40%
12:00 AM


Resource - chat - Eclipse SDK
File Edit Navigate Search Project Editor Menu Run Window Help
Team Synchr...
CVS Reposito...
Resource
Navigator
>stubs
>META-INF
>specification
>module
chat.module.view 1.1
>scenario
chat.scenario 1.1 (AS
Outline
ChatServer
broadcastMessage
toBroadcast
CCM
chat
chat
LogicalGates
Scenario Detail View
Instances
Name
Kind
Type
ChatClient1
CCMComponent
ChatClient
handleMessage
consumes
HandleMessage
postMessage
uses
SendMessage
ChatClient2
CCMComponent
ChatClient
handleMessage
consumes
HandleMessage
postMessage
uses
SendMessage
ChatServer
CCMComponent
ChatServer
toBroadcast
provides
SendMessage
broadcastMessa
publishes
HandleMessage
Connections
Name
Kind
End Point
CCMEventConnector
CCMEvent
ChatClient1.handleMessage
CCMEvent
ChatServer.broadcastMess...
CCMEventConnector
CCMEvent
ChatClient2.handleMessage
CCMEvent
ChatServer.broadcastMess...
CCMInterfaceConnector
CCMInterface
ChatClient1.postMessage
CCMInterface
ChatServer.toBroadcast
CCMInterfaceConnector
CCMInterface
ChatClient2.postMessage
CCMInterface
ChatServer.toBroadcast
Open Ports
Name
Type
Paren
Open Properties
Name
Type
Parent
Overview
Table
Graph
Tasks
Properties
Property
Value
Core
name
ChatServer
Start
41%
11:58 PM

Component Systems Workflow

Idealized!

Component developer

I. Interface Specification

- Design interfaces
- Design component interfaces

Component integrator

III. Component Integration

- Allocate component instances
- Connect components

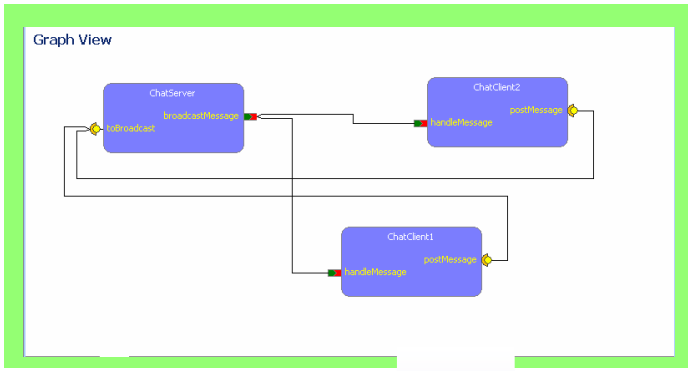
II. Component Implementation

- Generate Interface Definition Language descriptions
- Generate infrastructure code and executer skeletons
- Implement business logic
- Unit level testing

IV. System Deployment

- Generate deployment code
- Execute system

Generate CCM Deployment XML



Cadena CCM Plug-in
(model interpreter for scenarios)

Component Instance Allocations

....

....

Component Connection Info

....

....

CCM Component Assembly Description (.cad) XML file

CCM Component Assembly

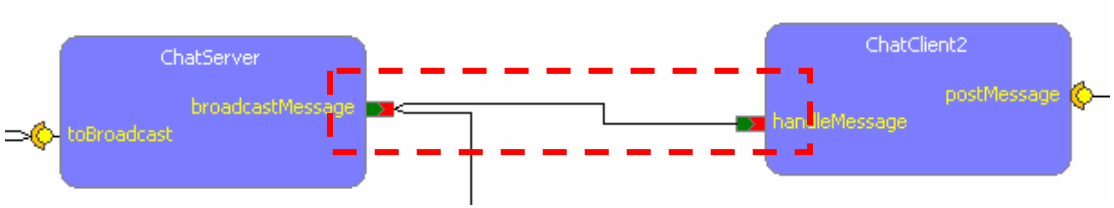
Component Instantiation (example)


```
<componentinstantiation id="ChatClient1">
    <registercomponent>
      <registerwithnaming name="demo/chat/ChatClient1"/>
      <registerwithtrader>
        <traderexport>
          <traderservicetypename>ComponentService</traderservicetypename>
          <traderproperties>
            <traderproperty>
              <traderpropertyname>description</traderpropertyname>
              <traderpropertyvalue>chat: The ChatClient1 component</traderpropertyvalue>
            </traderproperty>
          </traderproperties>
        </traderexport>
      </registerwithtrader>
    </registercomponent>
  </componentinstantiation>
```

CCM .cad file for Chat System

CCM Component Assembly

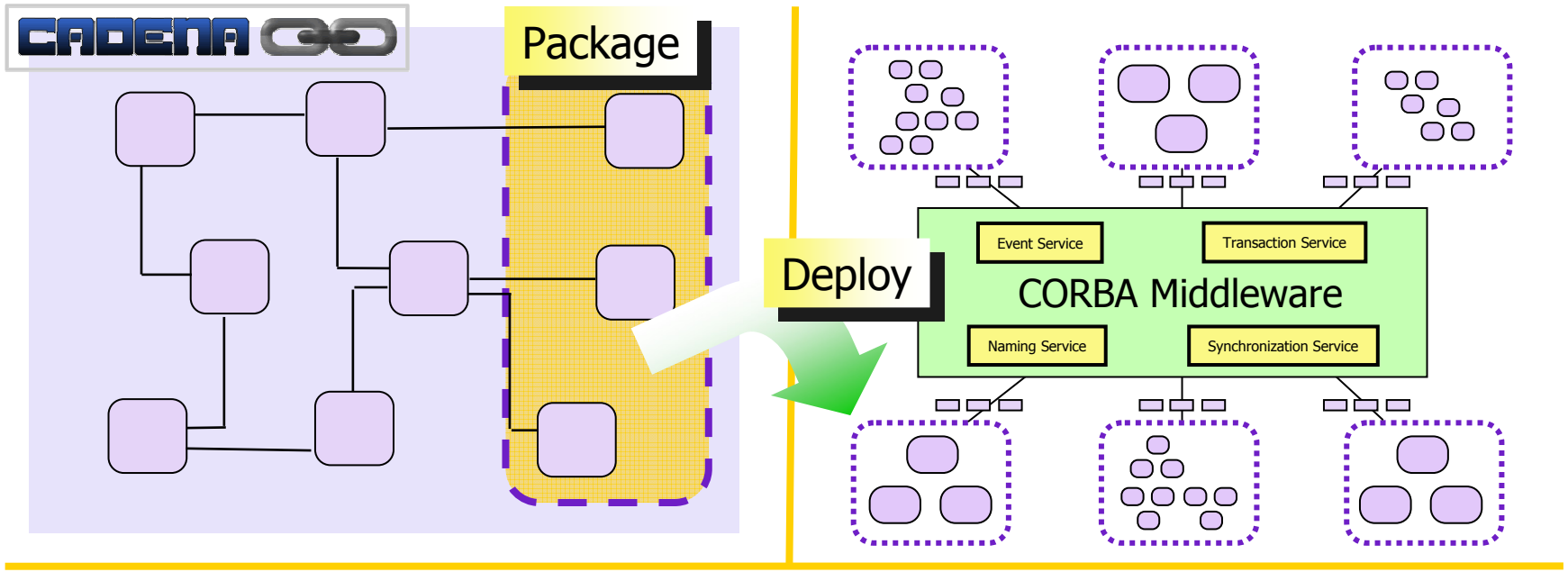
Component Connection (example)




```
<connectevent>
    <publishesport>
      <publishesidentifier>broadcastMessage</publishesidentifier>
      <componentinstantiationref idref="ChatServer"/>
    </publishesport>
    <consumesport>
      <consumesidentifier>handleMessage</consumesidentifier>
      <componentinstantiationref idref="ChatClient2"/>
    </consumesport>
  </connectevent>
```

CCM .cad file for Chat System

Packaging & Deployment

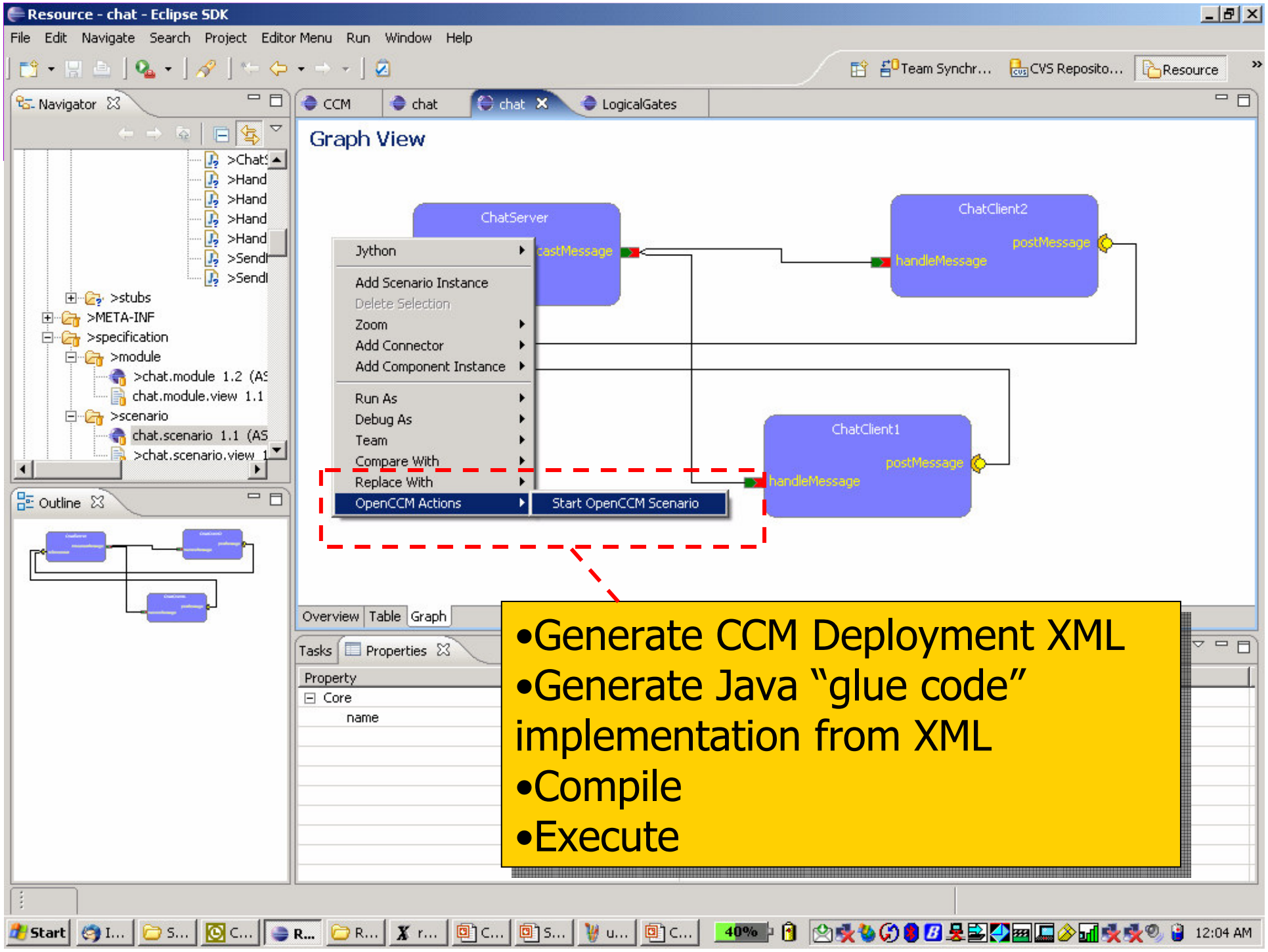


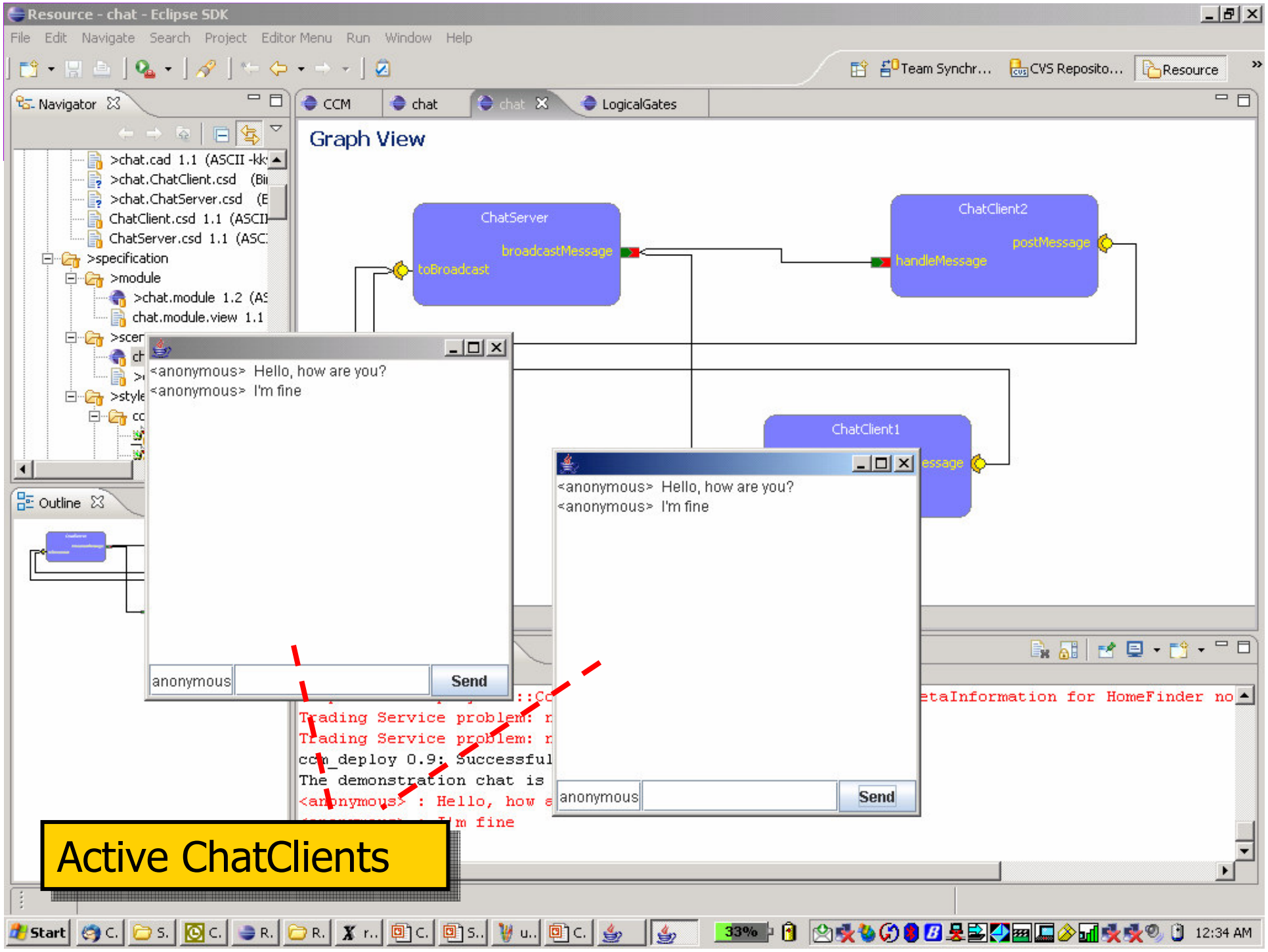
automatic generation

```
<CONFIGURATION_PASS>
  <HOME>          <...>
    <COMPONENT>
    <ID> <...></ID>
    <EVENT_SUPPLIER>
    <...events this component supplies...>
    </EVENT_SUPPLIER>
    </COMPONENT>
    </HOME>
</CONFIGURATION_PASS>
```

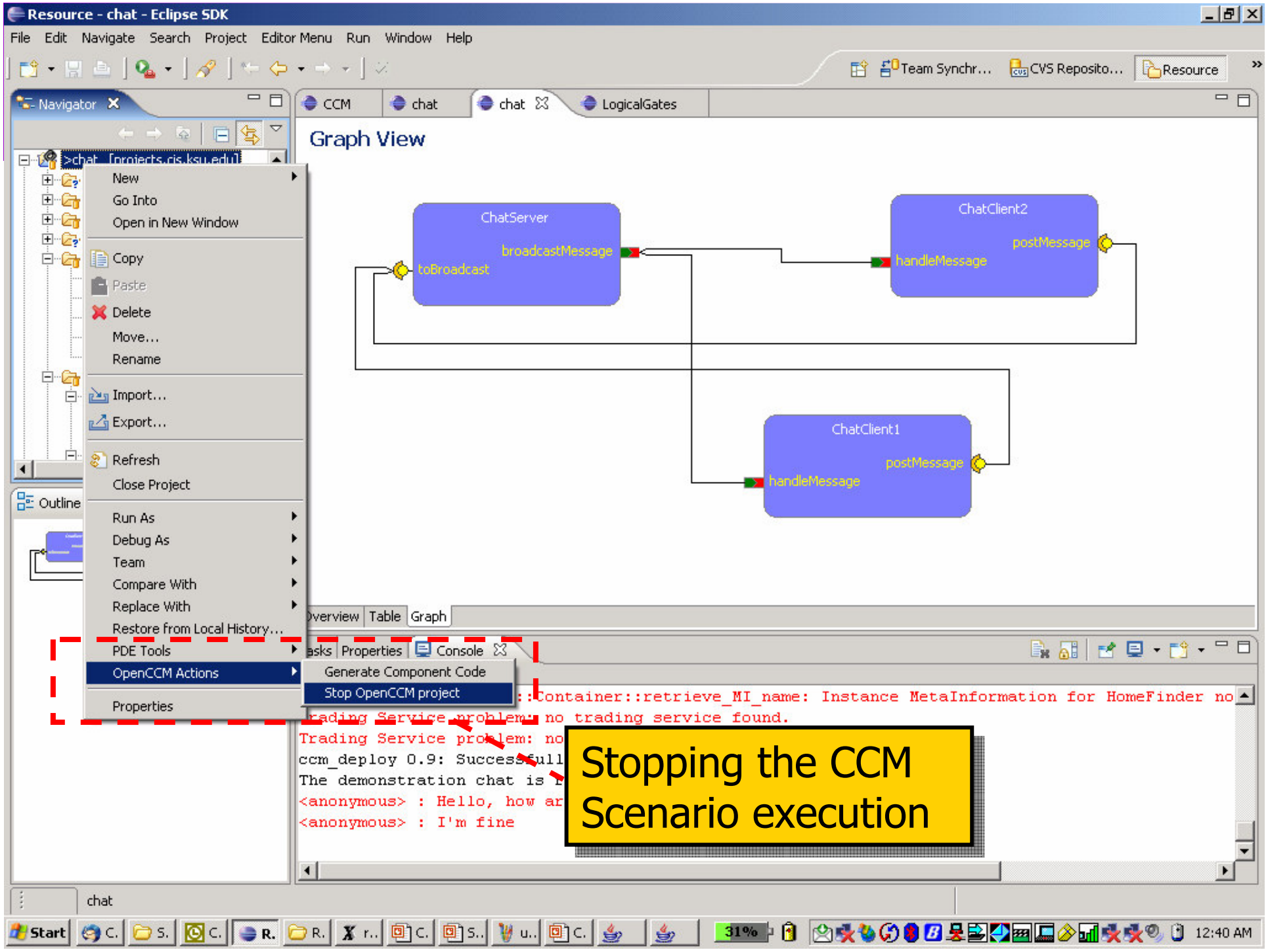
XML-based Configuration and Deployment information

CCM Deployment Infrastructure


Resource - chat - Eclipse SDK
File Edit Navigate Search Project Editor Menu Run Window Help
Navigator
Outline
CCM
chat
chat
LogicalGates
Graph View
ChatServer
castMessage
ChatClient2
postMessage
handleMessage
ChatClient1
postMessage
handleMessage
Jython
Add Scenario Instance
Delete Selection
Zoom
Add Connector
Add Component Instance
Run As
Debug As
Team
Compare With
Replace With
OpenCCM Actions
Start OpenCCM Scenario
Overview
Table
Graph
Tasks
Properties
Property
Core
name
•Generate CCM Deployment XML
•Generate Java "glue code" implementation from XML
•Compile
•Execute



Active ChatClients


Resource - chat - Eclipse SDK
File Edit Navigate Search Project Editor Menu Run Window Help
Navigator
CCM
chat
chat
LogicalGates
Graph View
ChatServer
broadcastMessage
toBroadcast
ChatClient2
postMessage
handleMessage
ChatClient1
postMessage
handleMessage
New
Go Into
Open in New Window
Copy
Paste
Delete
Move...
Rename
Import...
Export...
Refresh
Close Project
Run As
Debug As
Team
Compare With
Replace With
Restore from Local History...
PDE Tools
OpenCCM Actions
Properties
Generate Component Code
Stop OpenCCM project
Overview
Table
Graph
Tasks
Properties
Console
::Container::retrieve_MI_name: Instance MetaInformation for HomeFinder no
trading service problem: no trading service found.
Trading Service problem: no
ccm_deploy 0.9: Successfull
The demonstration chat is r
<anonymous> : Hello, how ar
<anonymous> : I'm fine
Stopping the CCM Scenario execution
chat
Start
31%
12:40 AM

Summary

- In this lecture, we have seen…
  - how to use Cadena to define a development environment for an industry standard component framework
  - examples of simple types of code generation plug-ins that can be incorporated into Cadena
- It is straightforward to build support for other component models and a wide variety of plug-ins in a very similar way.

Assessment

- In its present form, Cadena seems will suited for working with the types of static configurations found in embedded systems (see nesC illustration).
- At present, Cadena does not provide direct support for dynamic reconfiguration of component connections.

nesC – Sensor Network Components

```
module SurgeM {
  provides {
    interface StdControl;
  }
  uses {
    interface ADC;
    interface Timer;
    interface Leds;
    interface StdControl as Sounder;
    interface Send;
    interface Receive as Bcast;
    interface RouteControl;
  }
}
```

Assessment CCM

- CCM is just one example of a component framework
- Some complain that CCM is very heavy-weight, so lightweight versions of CCM have been developed for embedded systems
- Let's look at how CCM compares to a few other widely used component frameworks…