Model-Driven Development of Real-Time Embedded Systems in Boeing's BoldStroke Product-Line

SAnToS Laboratory, Kansas State University, USA

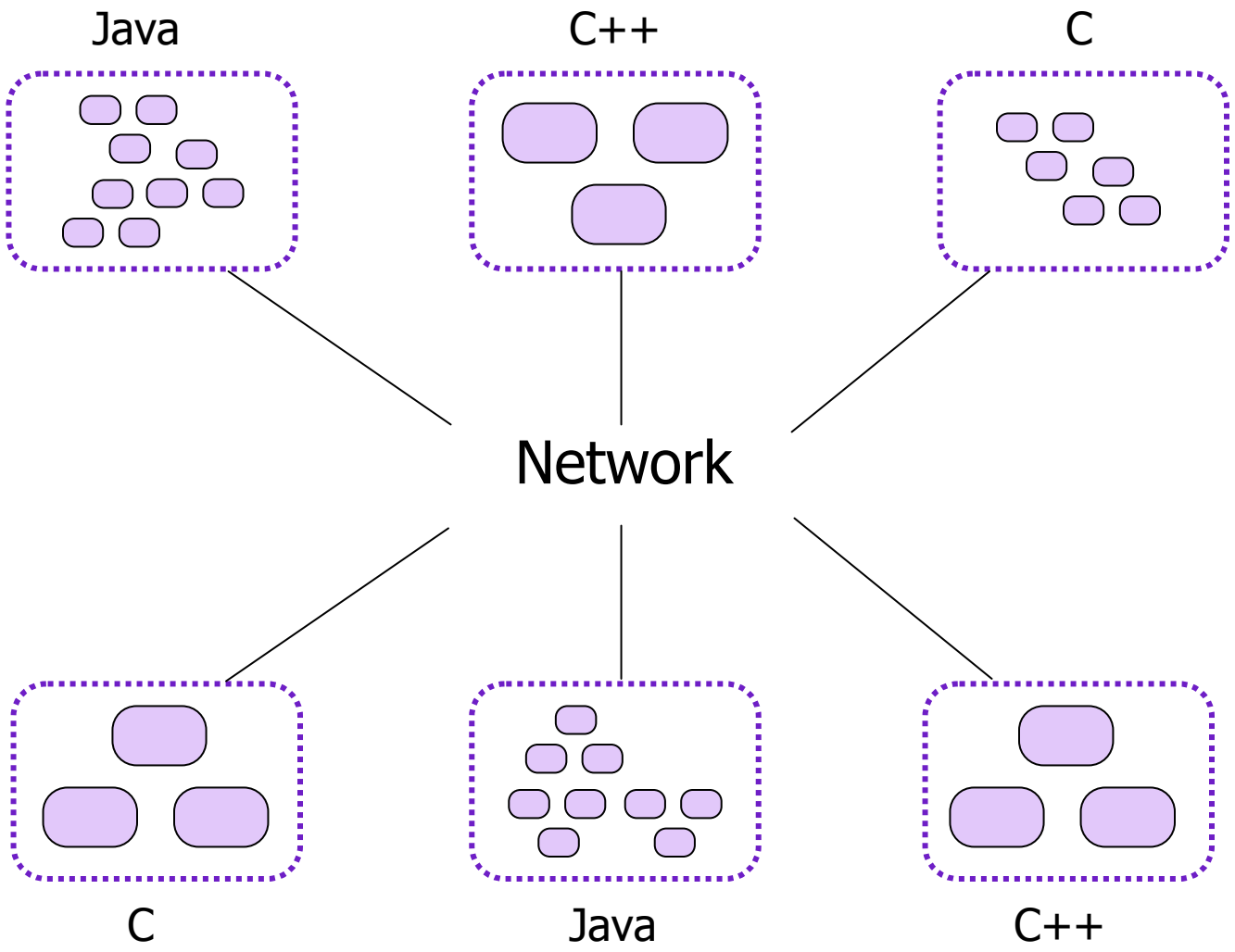
http://cadena.projects.cis.ksu.edu

John Hatcliff
William Deng
Venkatesh Ranganath
Matt Dwyer
Georg Jung
Robby
Jesse Greenwald
Gurdip Singh
Adam Childs
Prashant Shanti Kumar
Matt Hoosier

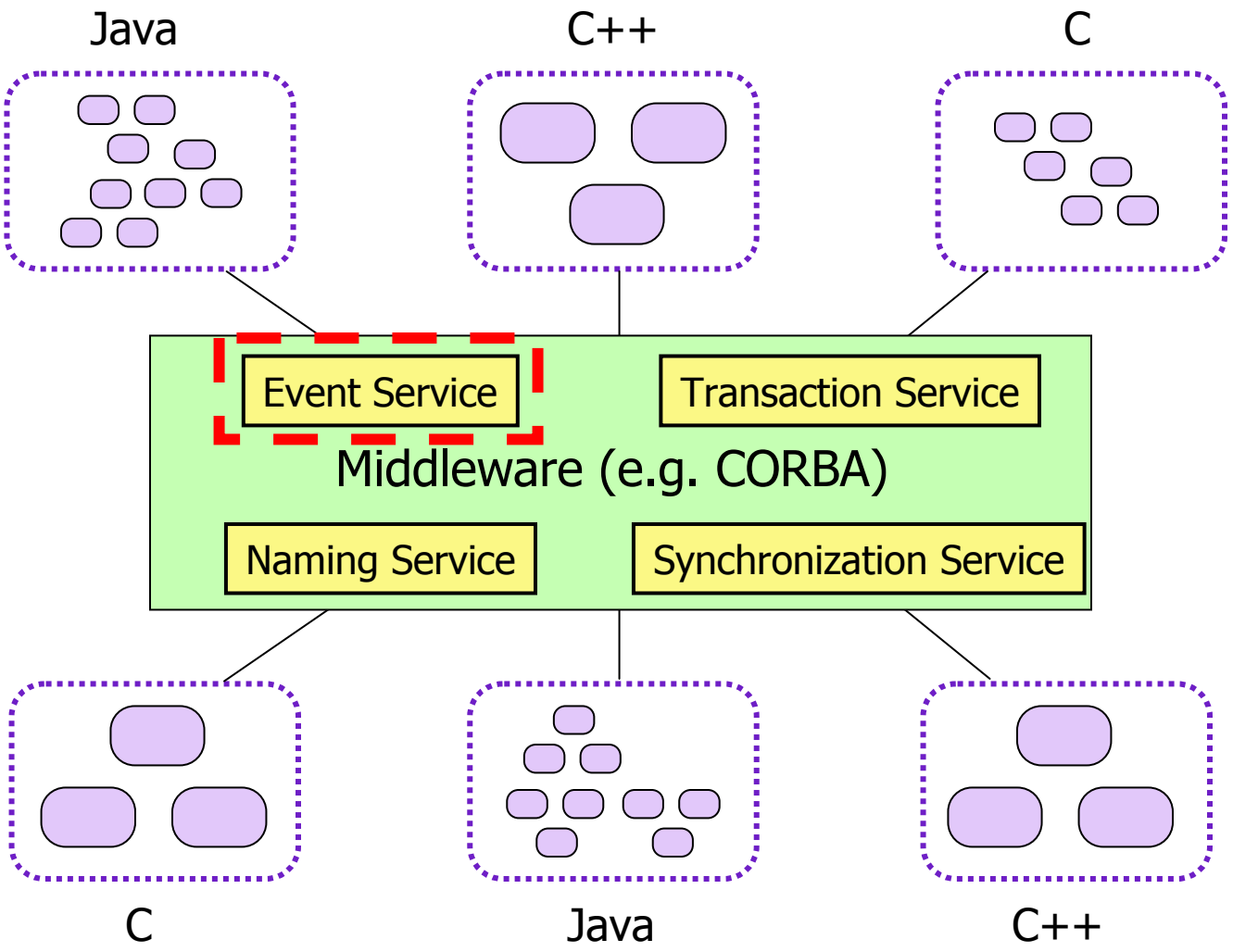
Support

US Army Research Office (ARO)
US National Science Foundation (NSF)
US Department of Defense
Advanced Research Projects Agency (DARPA-PCES)
Rockwell-Collins ATC
Lockheed-Martin [Eagan] – (PCES sub-contract)
Honeywell Technology Center and NASA Langley
IBM, Intel, Sun Microsystems

Distributed Components



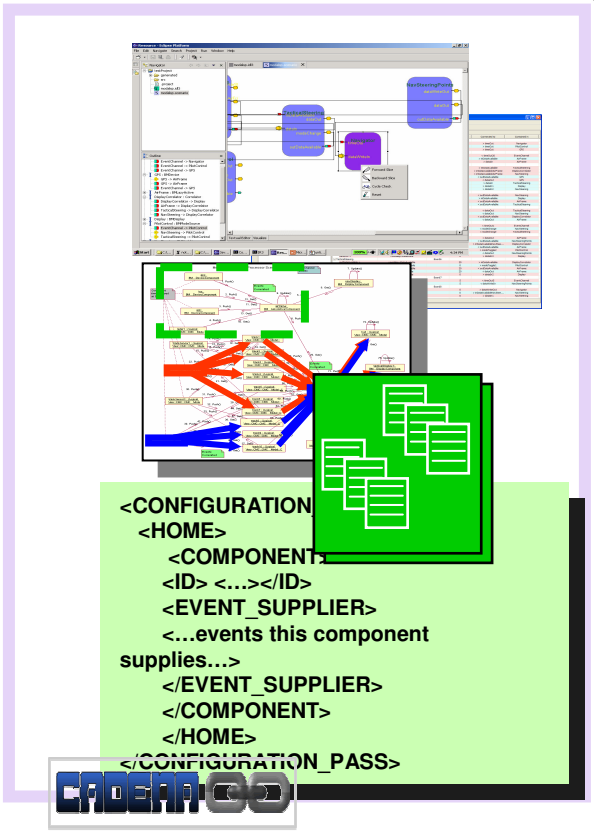
Distributed Components



Cadena Development

Cadena I

…tailored to CCM

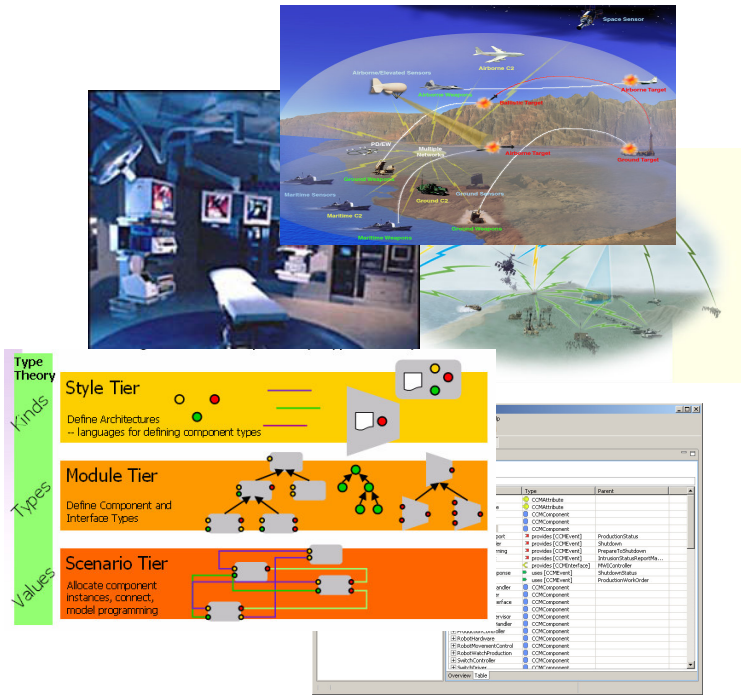


June 2002 – May 2005

The focus of this talk

Cadena II

…full meta-modeling (CCM. EJB, PRiSM, etc.) for product-line architectures



Jan 2005 – …

Avionics Mission Control Systems

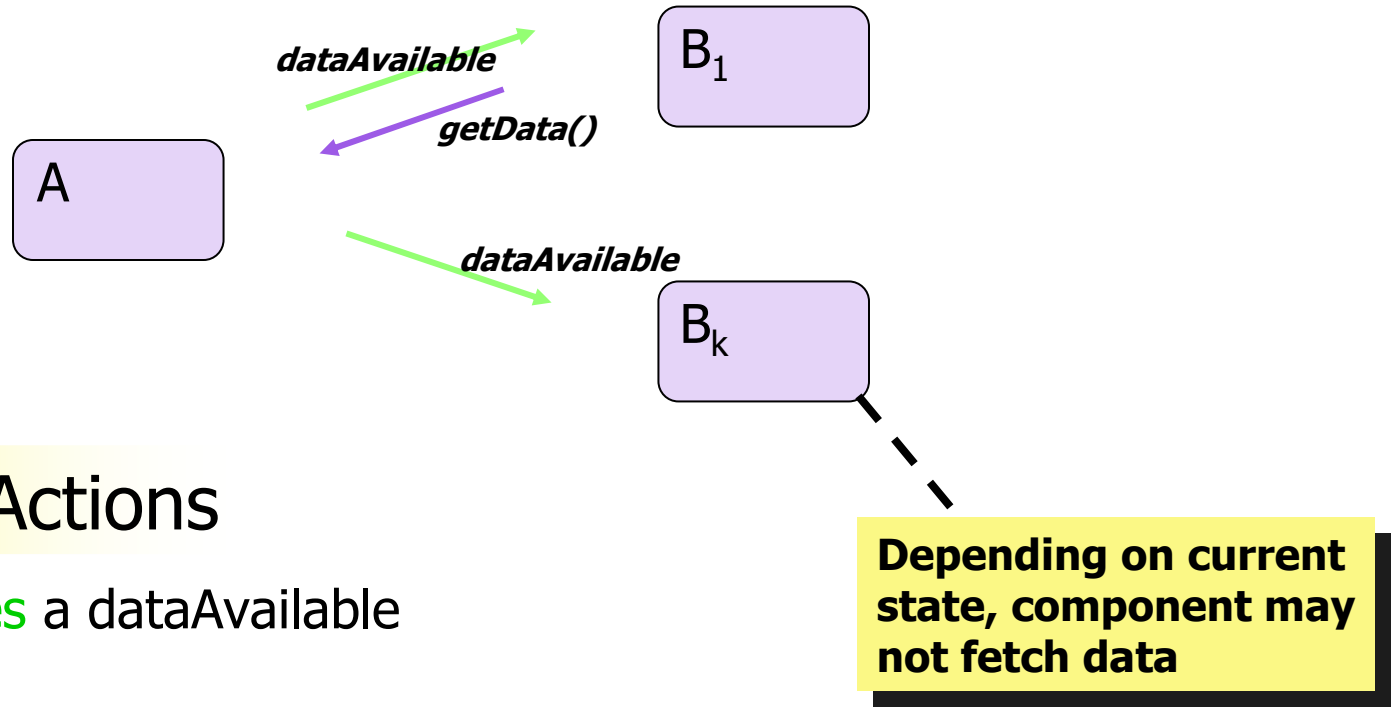
- PCES/MoBIES Experimental Platform
- Mission-control software for Boeing military aircraft
- Boeing's Bold Stroke Avionics Middleware
  - …built on top of ACE/TAO RT CORBA

- Provided with an Open Experimental Platform (OEP) from Boeing
  - a sanitized version of the real system
  - 100,000+ lines of C++ code (including RT CORBA middleware)
  - 50+ page document that outline development process and describe challenge problems
- Must provide…
  - tool-based solutions that can be applied by Boeing research team to realistic systems
  - solutions that fit within current development process, code base, etc.
  - metrics for that allow Boeing research team to evaluate tool performance and ease of use

Control-Push Data-Pull

Typical situation

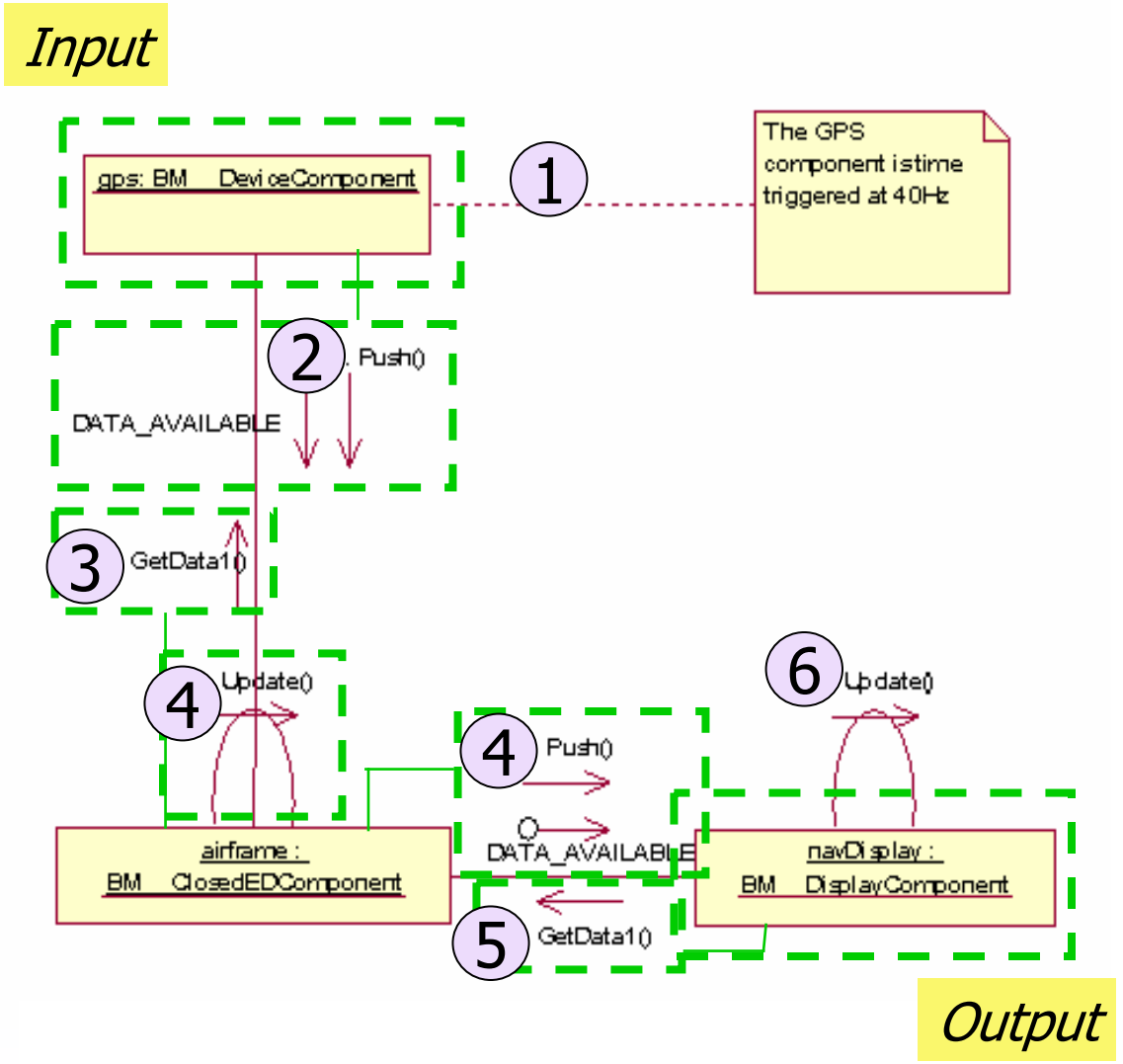
Component A computes some data that is to be read by one or more components B_i



Run-time Actions

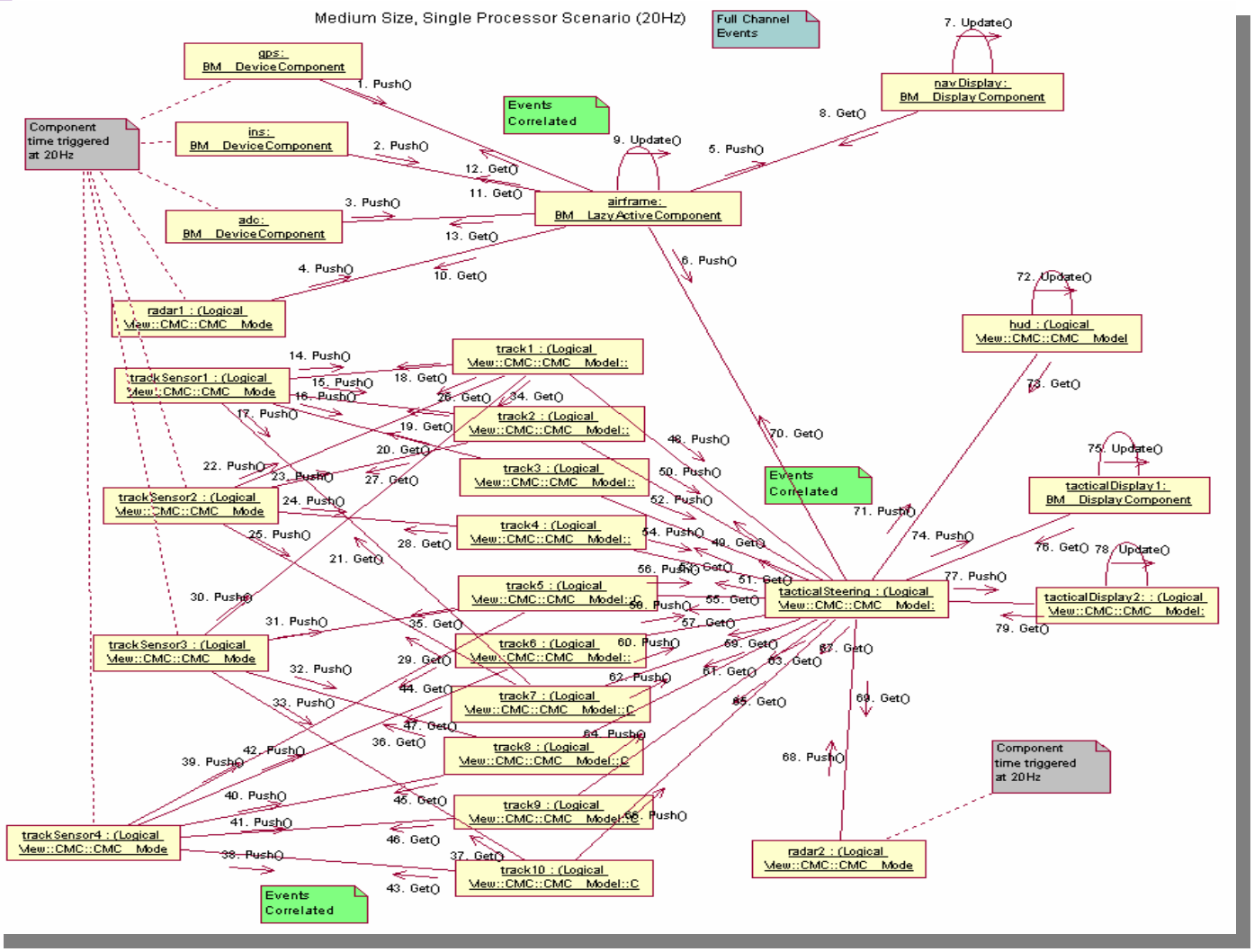
- A publishes a dataAvailable event
- B_i call the getData() method of A to fetch the data

Control-Push Data-Pull Structure



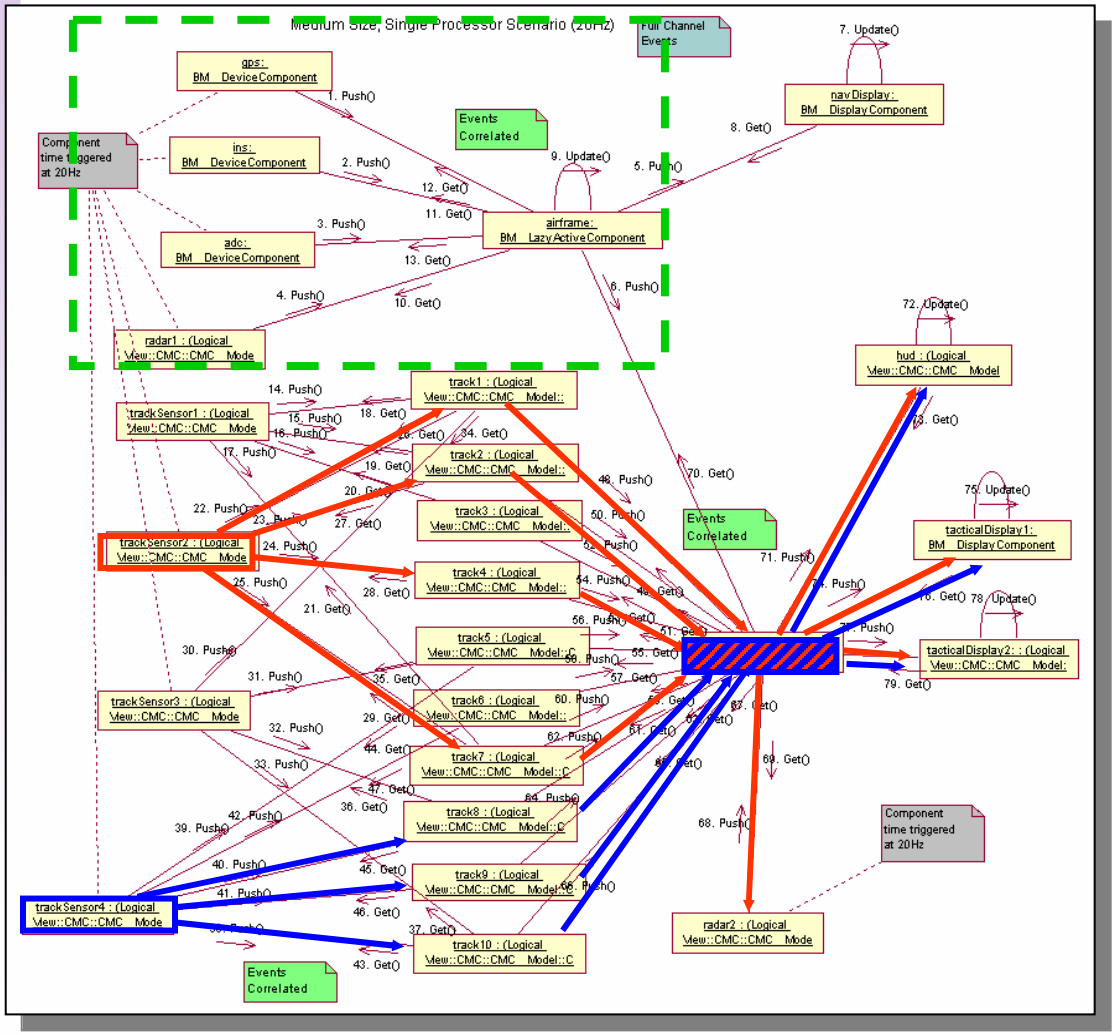
1. Logical GPS component *receives a periodic event* indicating that it should read the physical GPS device.
2. Logical GPS *publishes DATA_AVAILABLE event*
3. Airframe component *fetches GPS data* by calling GPS GetData method
4. Airframe *updates its position data* and publishes DATA_AVAILABLE event
5. NavDisplay component *fetches AirFrame data* by calling AirFrame GetData method
6. NavDisplay *updates the physical display*

Larger Configuration



…moving up to 1000+ components

Lack of Model Analysis



Boeing OEP Challenge Problems

1. Forward & backward data and event dependencies

2. Dependency intersections

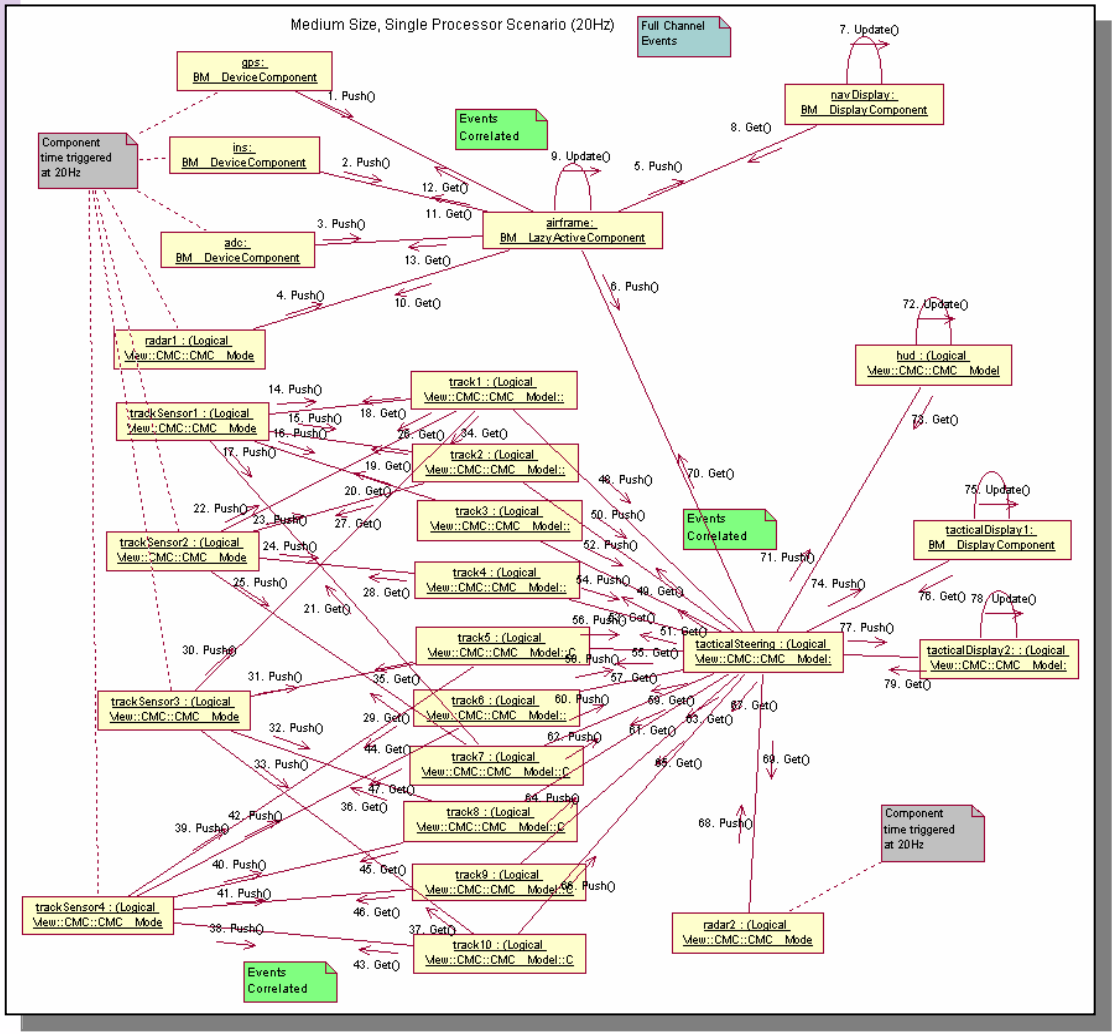
3. Components with high data coupling

4. All components from a particular rate group

5. Cycle checks

…15-20 others related to dependencies

Lack of Model Analysis



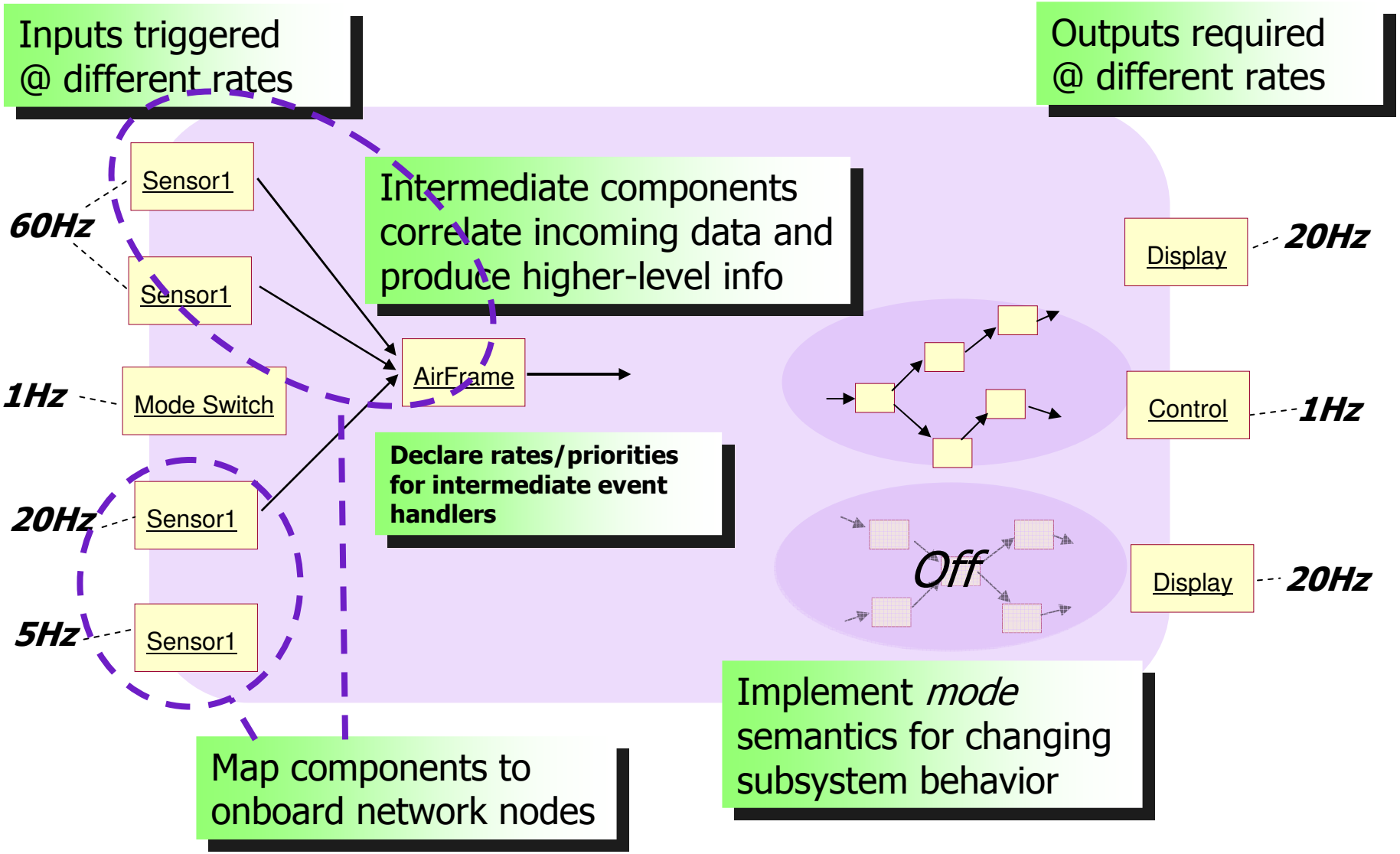
Boeing OEP Challenge Problems

If component 1 is in mode A when component 2 produces event E, then component 3 will consume event F

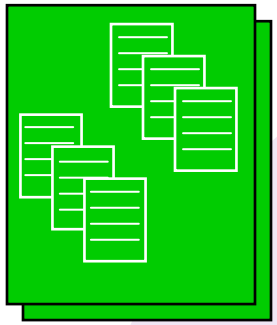
(Section 4.1.5.3.6)

A temporal property well-suited for model-checking!

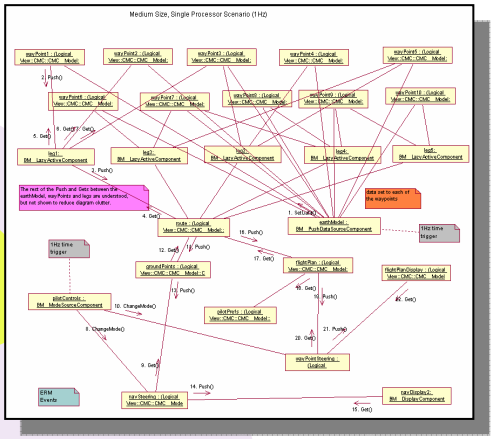
System Design Aspects



No Unifying Mechanism



C++
Component Code



UML Design Artifacts

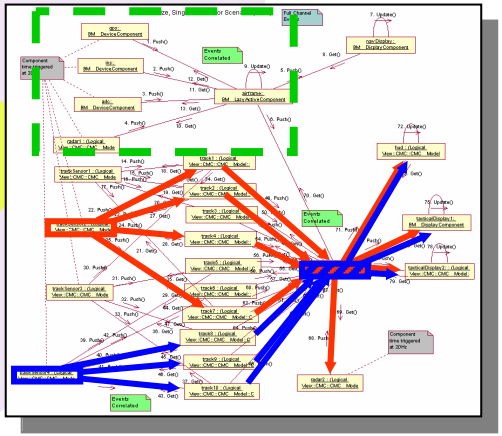
```
<CONFIGURATION_PASS>
  <HOME>          <...>
    <COMPONENT>
    <ID> <...></ID>
    <EVENT_SUPPLIER>
    <...events this component supplies...
    </EVENT_SUPPLIER>
    </COMPONENT>
    </HOME>
</CONFIGURATION_PASS>
```

Deployment XML
Configurator Info

?

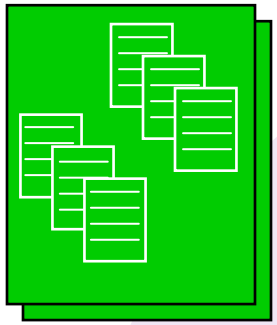
Model-Driven
Design, Analysis,
Configuration

Integrated Development
Environment



Analysis and QoS
Aspect Synthesis

Cadena



Java/C++
Component Code

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

Bold Stroke XML
Configurator Info

Cadena

CCM Interface Definition Language

System Assembly Support

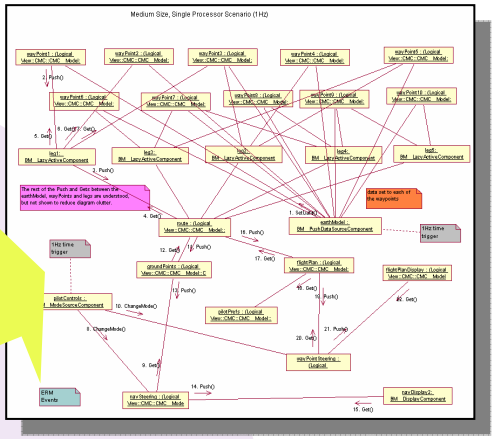
Light-weight Semantic Specs

Analyses

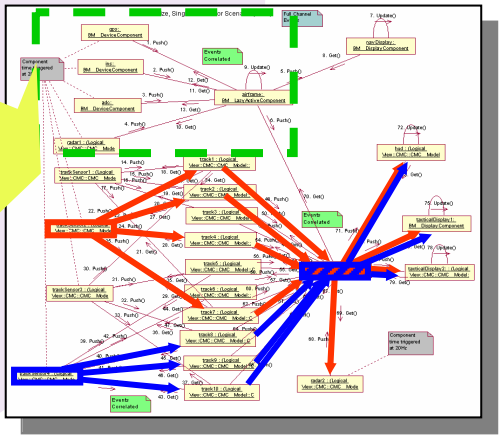
CIAO/TAO

Eclipse Plug-In

Integrated Development Environment

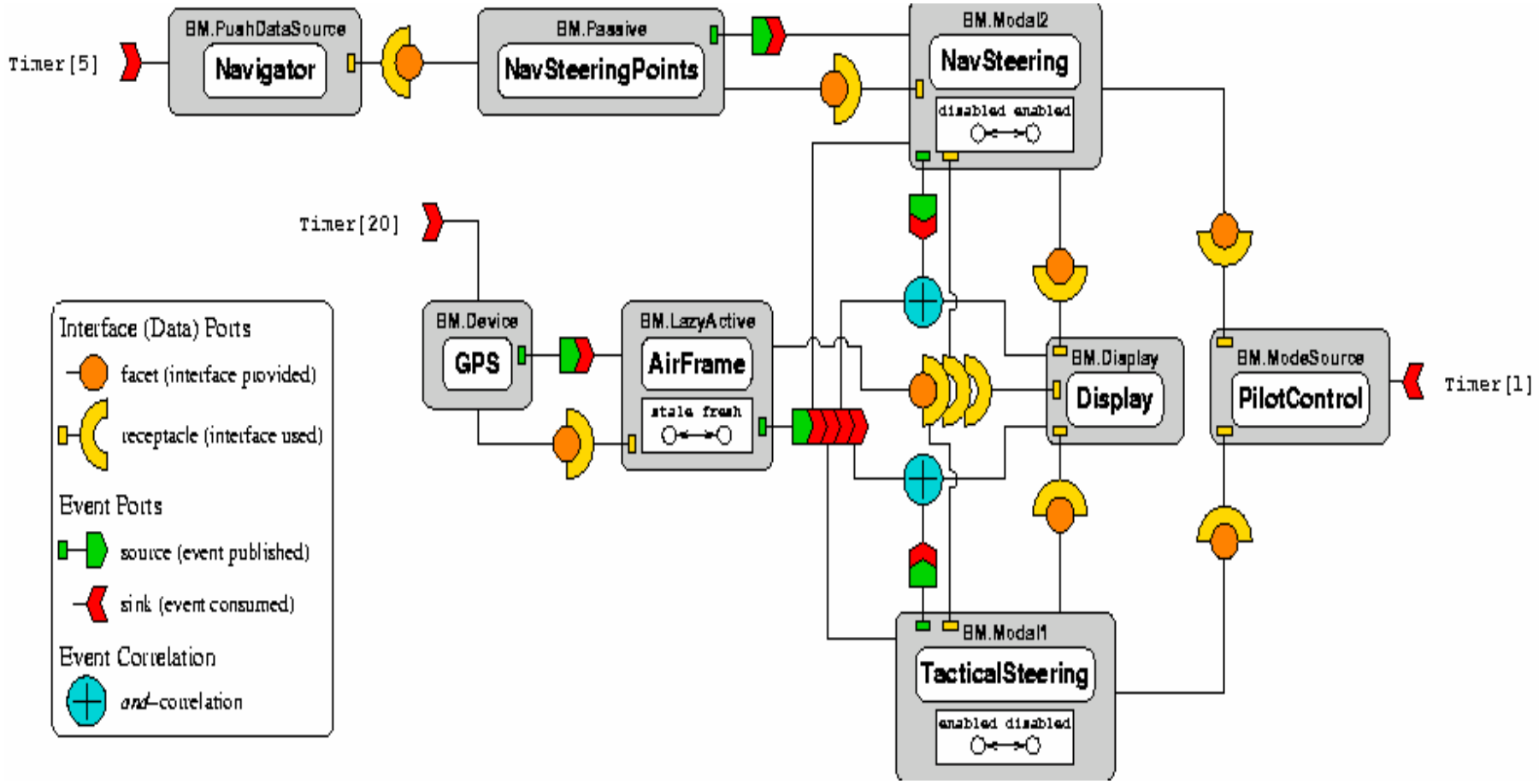


UML Design Artifacts

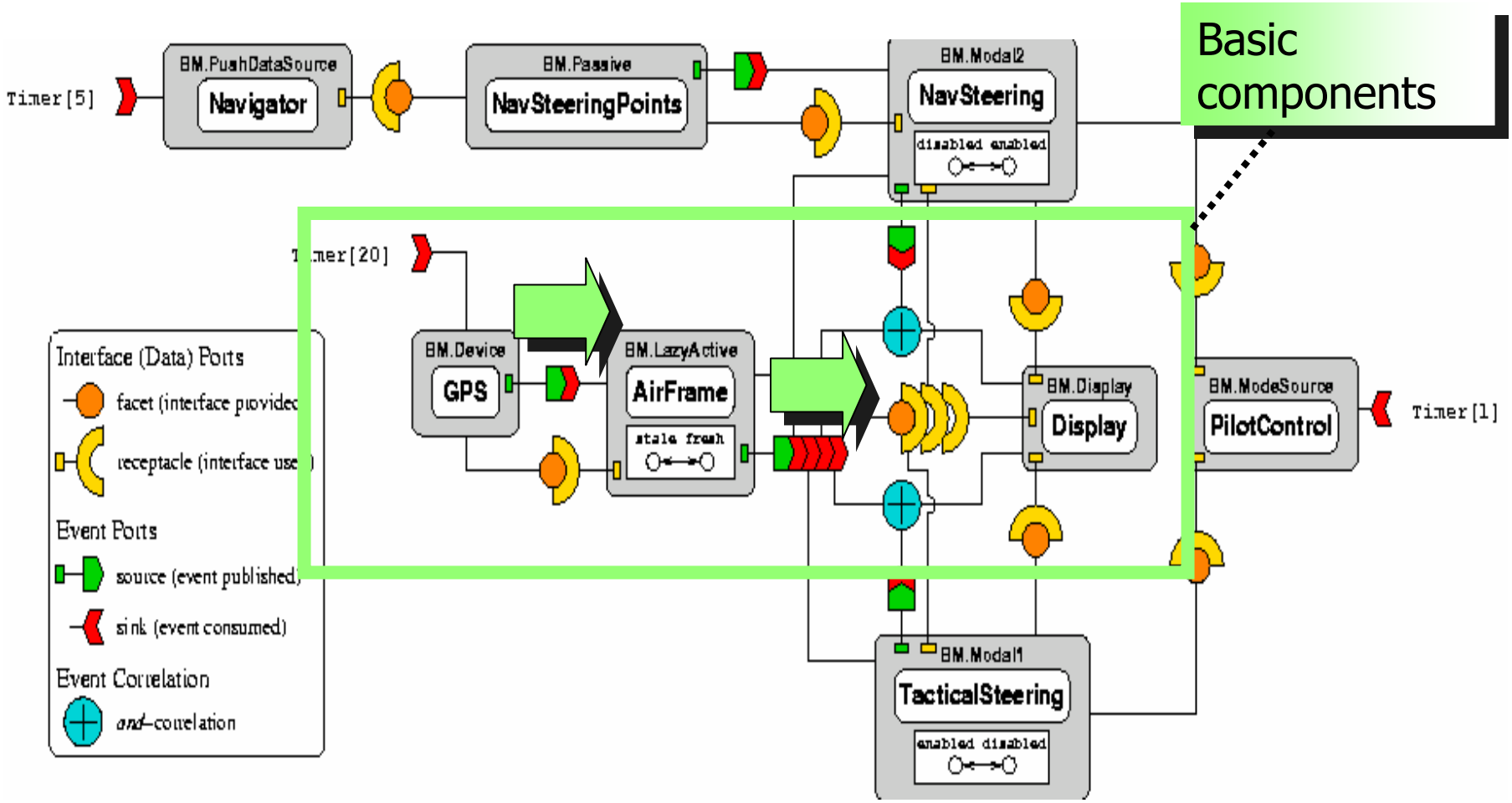


Analysis and QoS
Aspect Synthesis

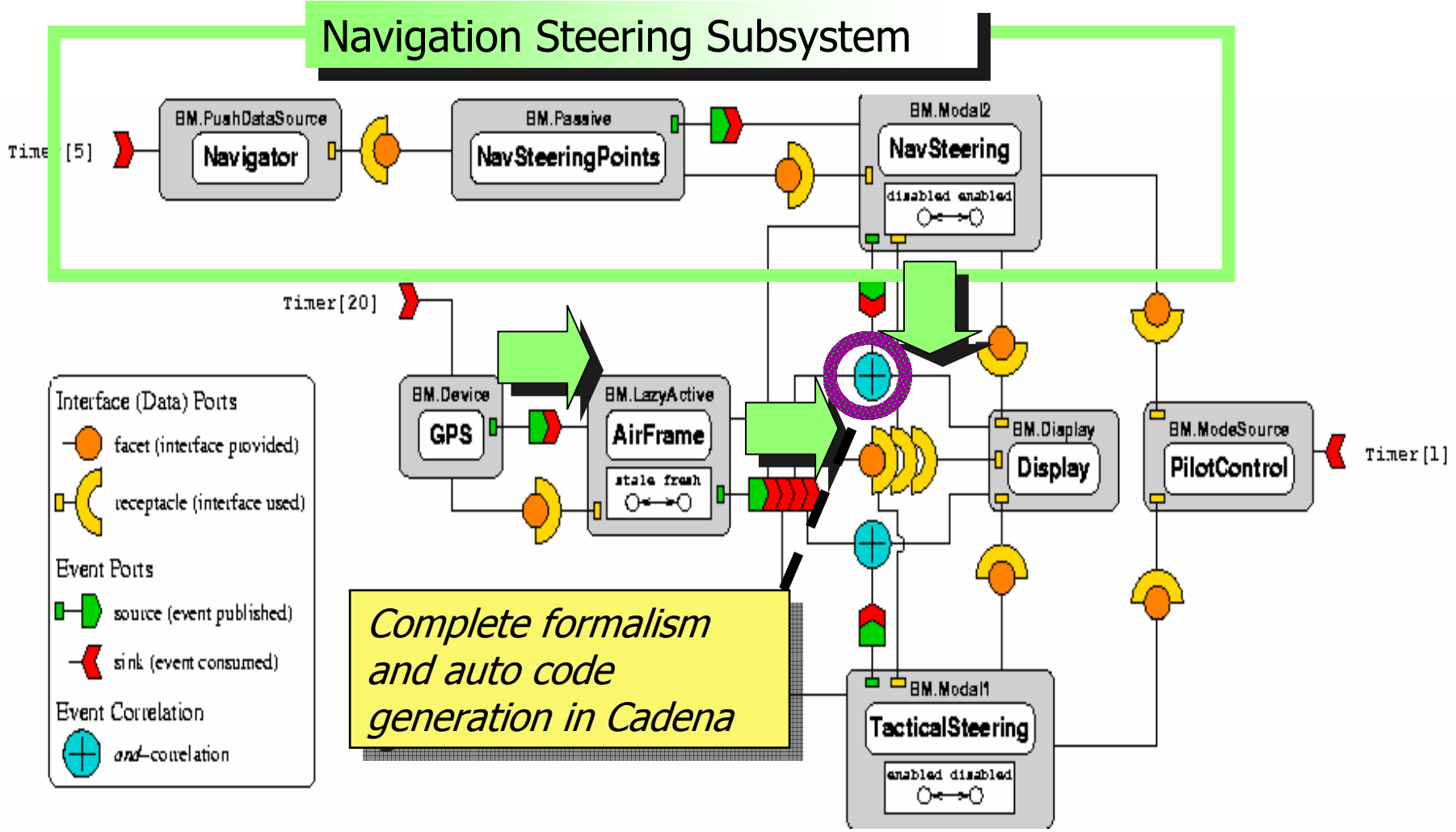
Example System



Example System

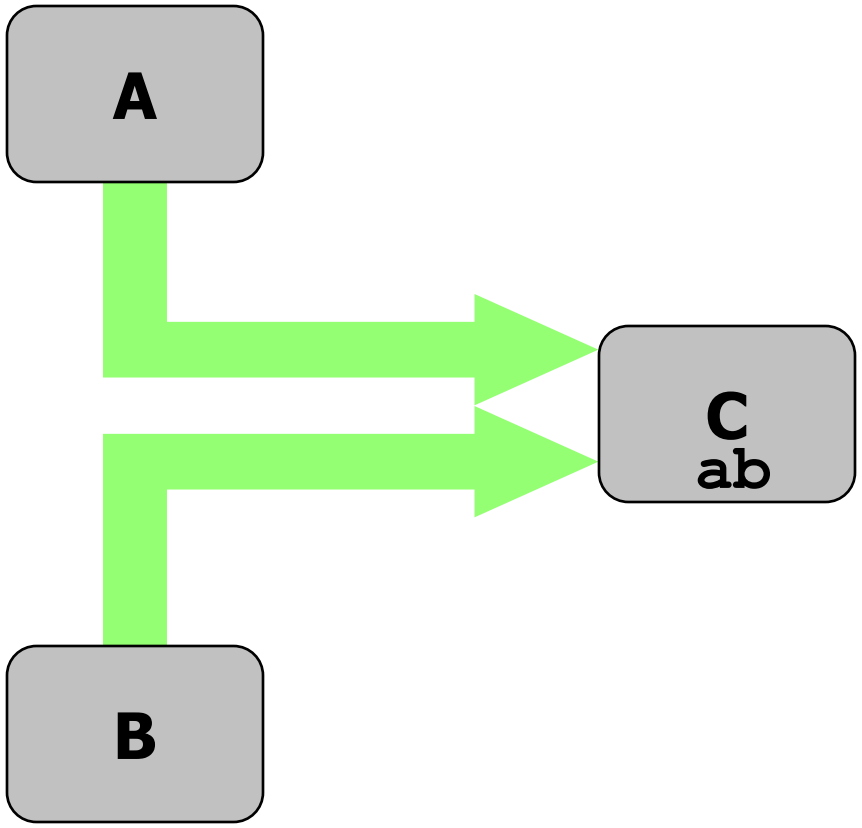


Example System



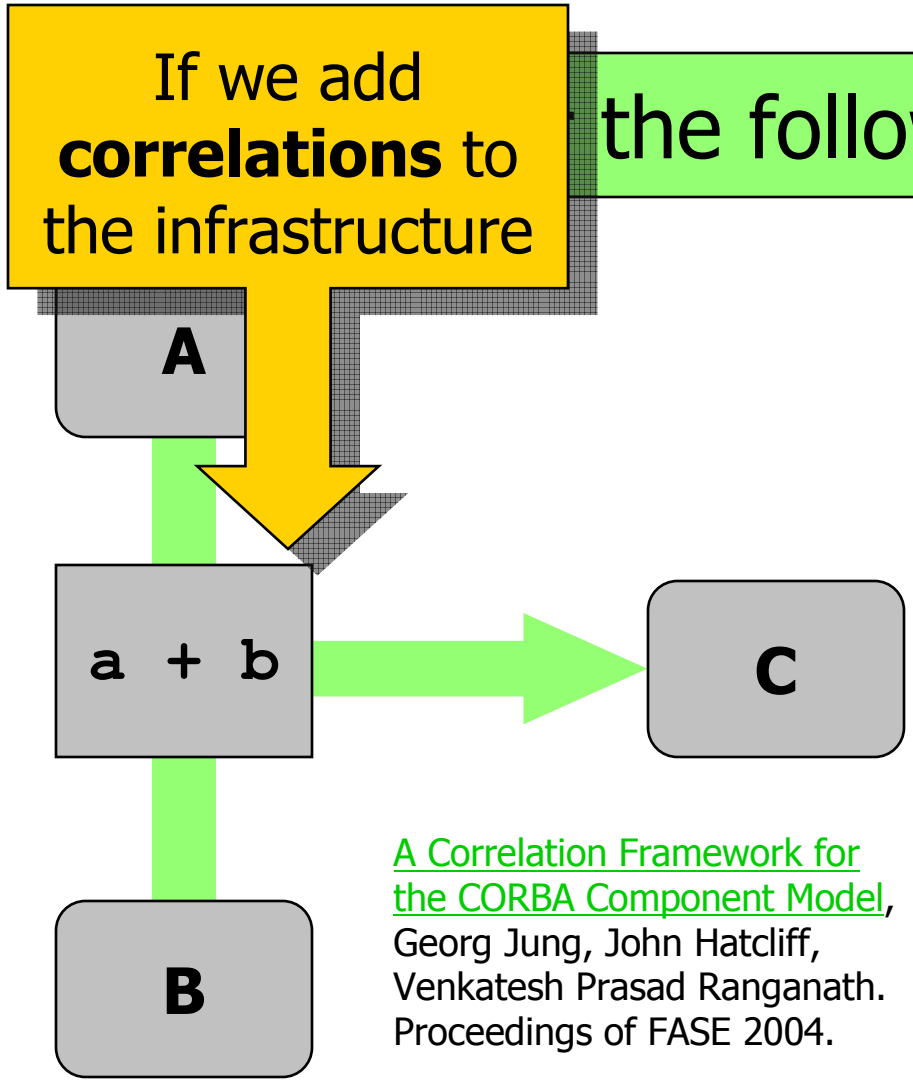
Challenges of Event Communication

Consider the following situation:



- Component **C** is receiving from two components **A** and **B**
- **A** and **B** send at different rates
- **C** needs both inputs to become active

Challenges of Event Communication

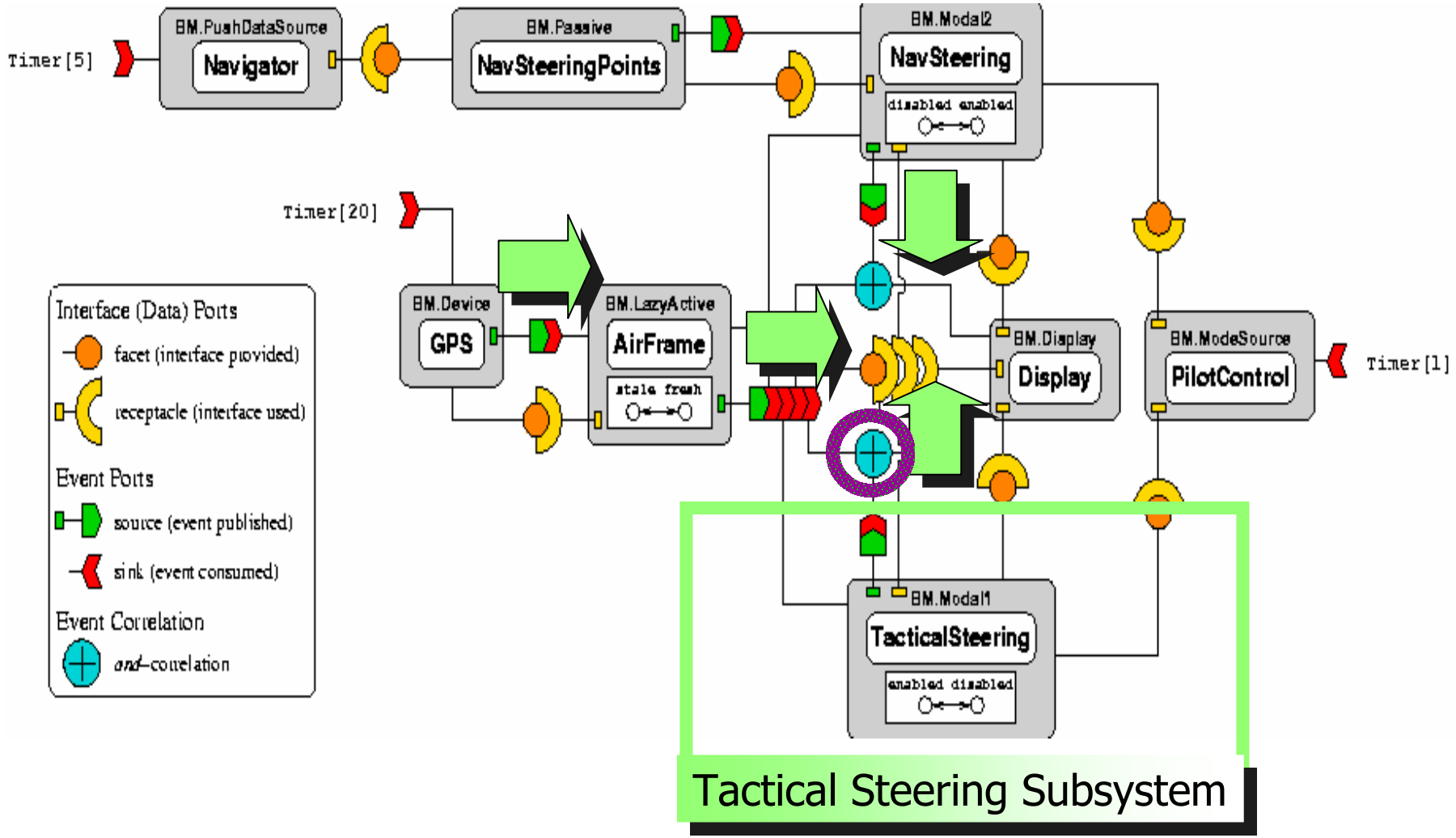


A Correlation Framework for the CORBA Component Model, Georg Jung, John Hatcliff, Venkatesh Prasad Ranganath. Proceedings of FASE 2004.

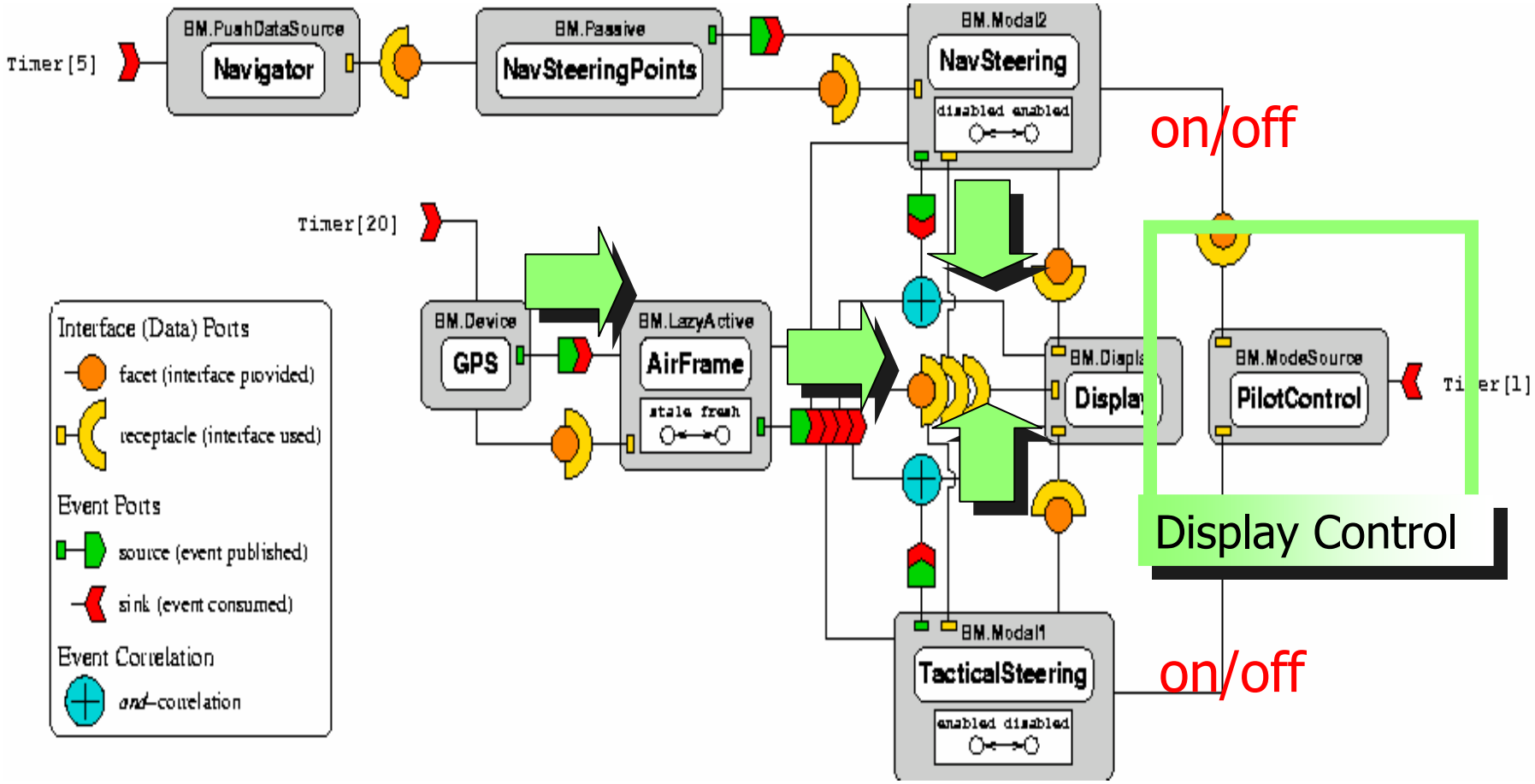
We can:

- Reduce network traffic
- Simplify computation inside the component
- Clarify the design
- Define the components in a more general way

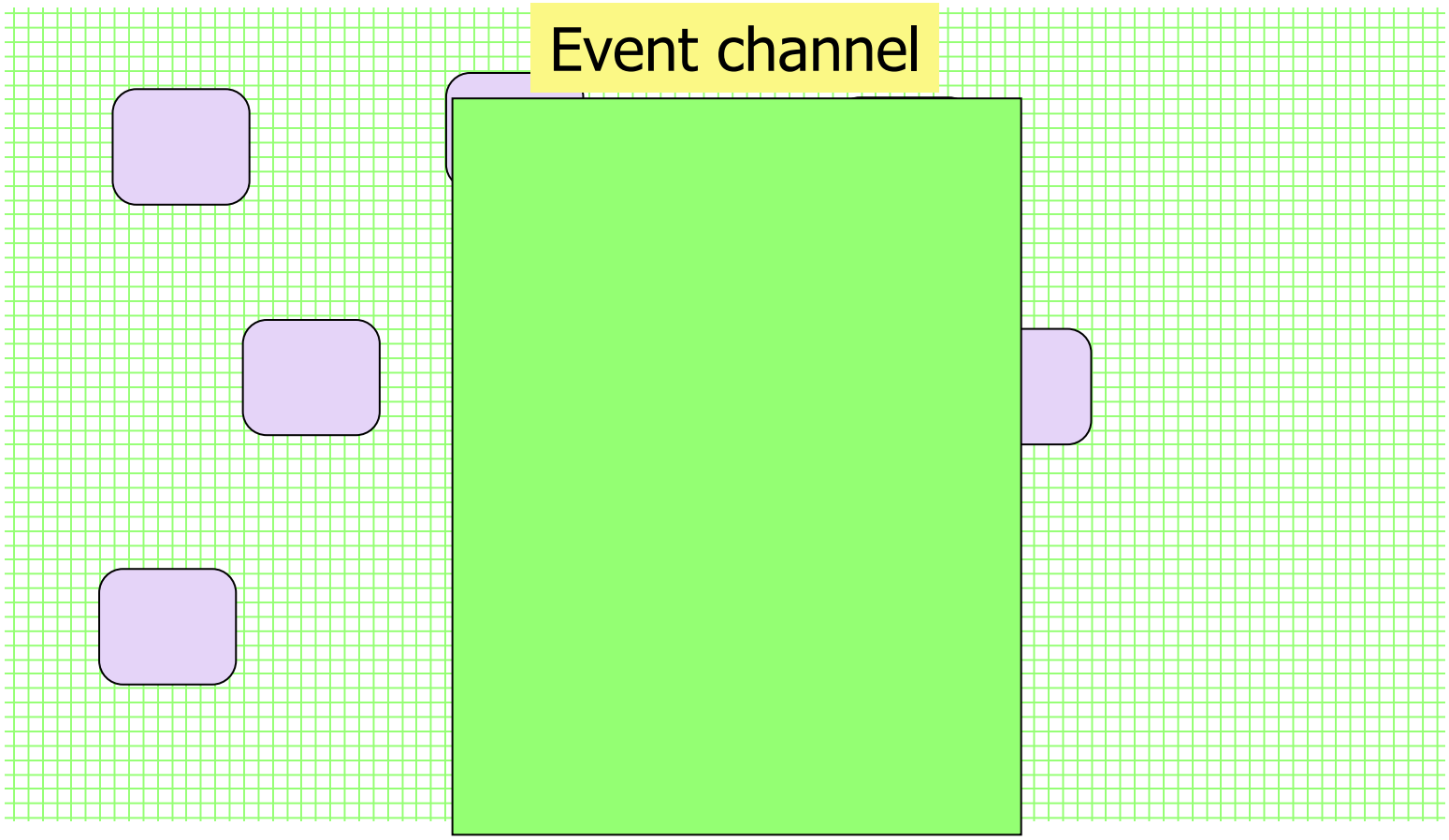
Example System



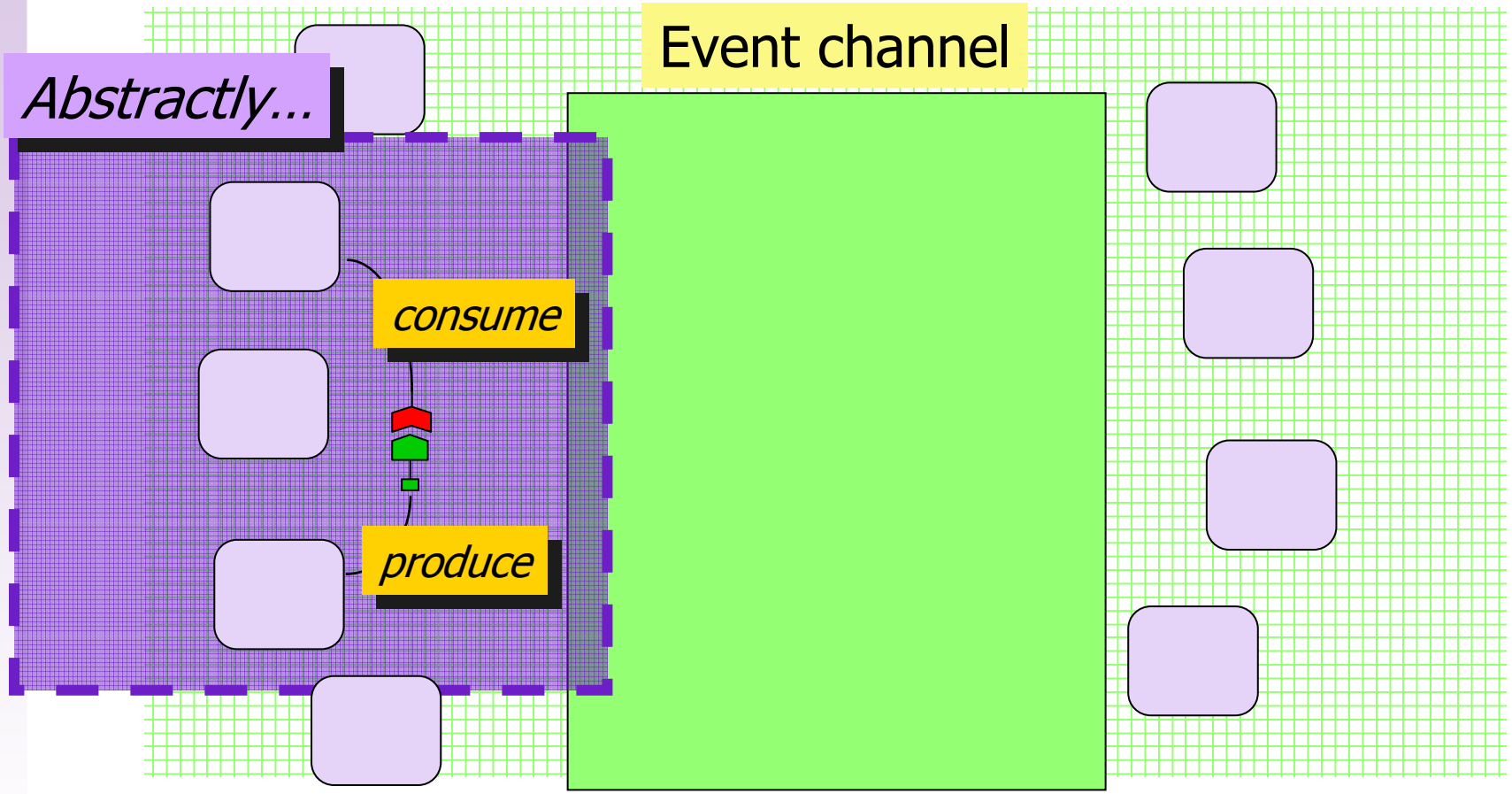
Example System



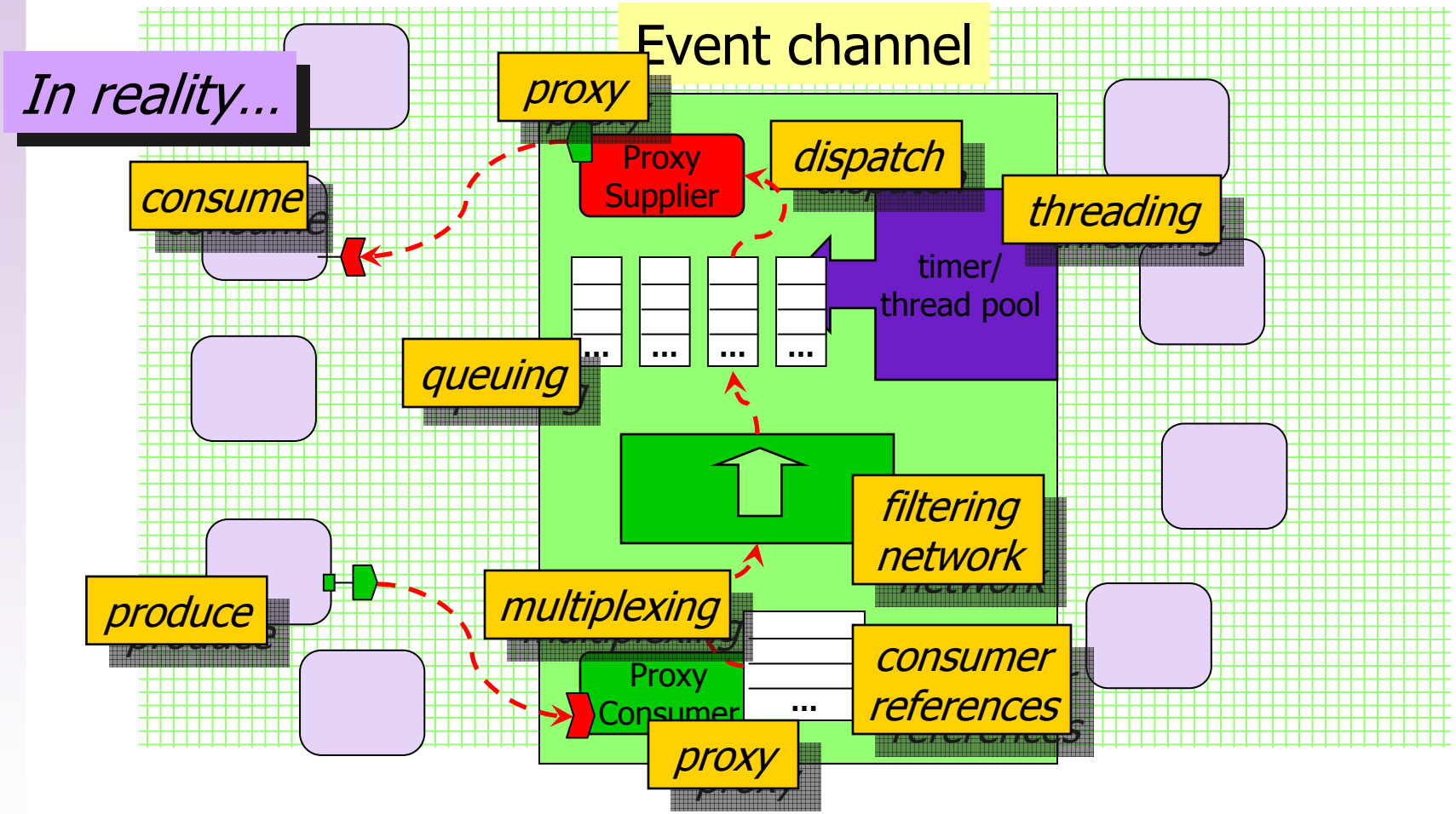
RT Event Channel Infrastructure



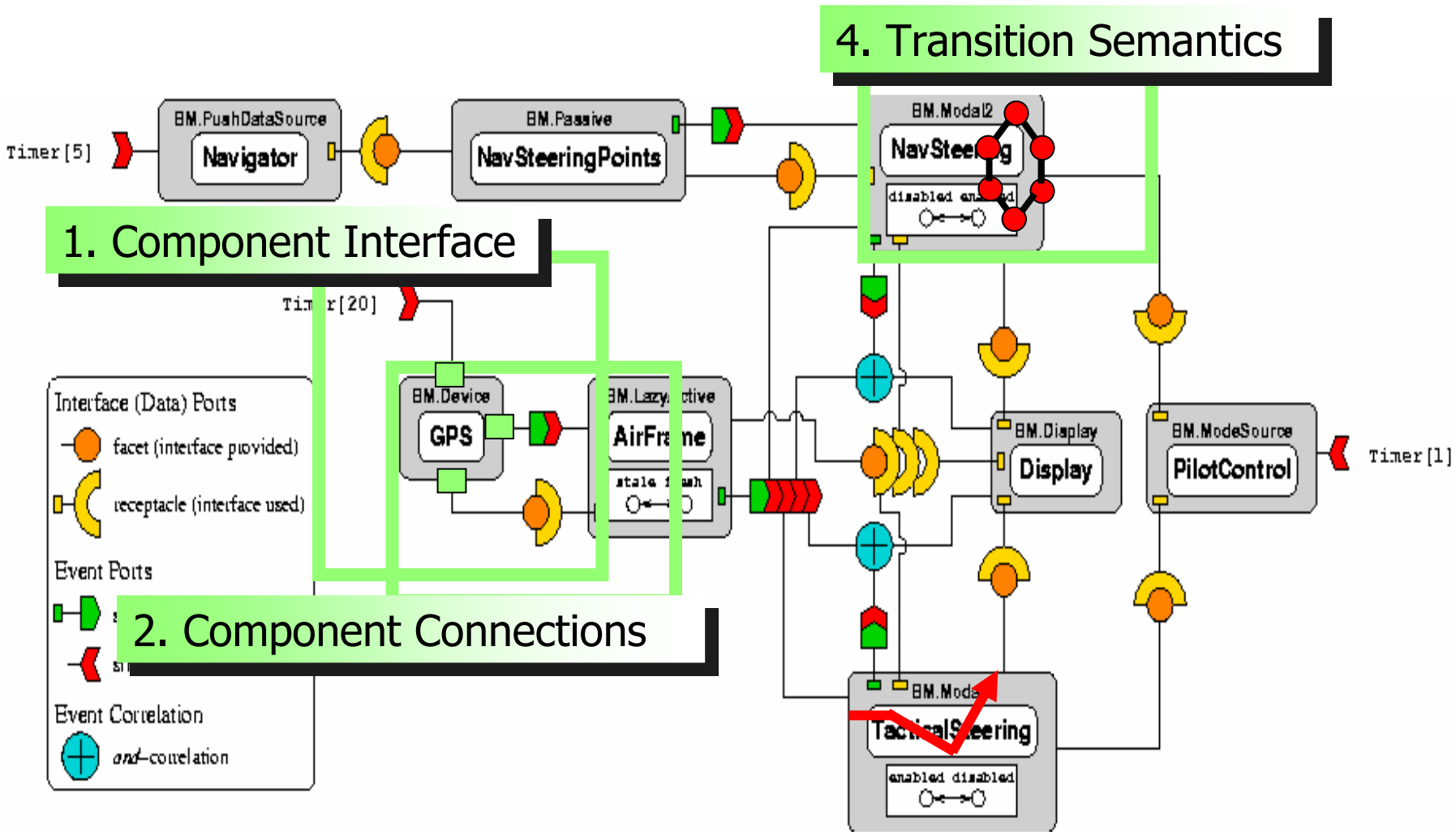
RT Event Channel Infrastructure



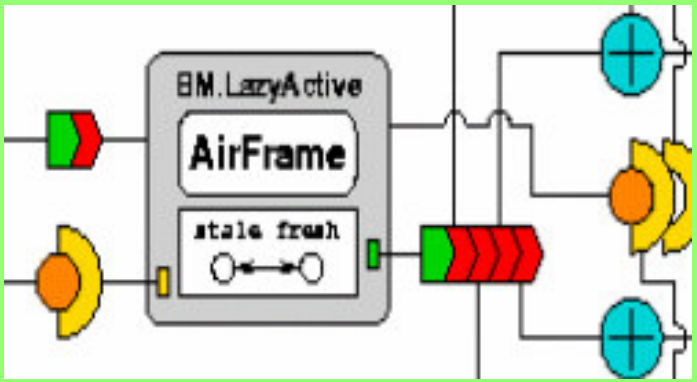
Boeing Threading Model



Outline


4. Transition Semantics
BM.PushDataSource
Navigator
Timer[5]
BM.Passive
NavSteeringPoints
BM.Modal2
1. Component Interface
Timer[20]
BM.Device
GPS
BM.Display
Display
BM.ModeSource
PilotControl
Timer[1]
Interface (Data) Ports
facet (interface provided)
receptacle (interface used)
Event Ports
Event Correlation
and-correlation
2. Component Connections
enabled disabled
3. Dependence Information

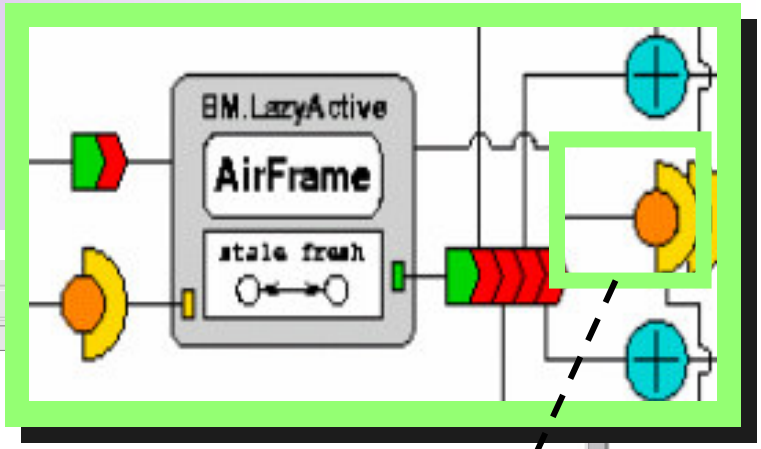
Component IDL




```
component BMLazyActive {
  provides  ReadData outData;
  uses      ReadData inData;
  publishes DataAvailable outDataAvailable;
  consumes  DataAvailable inDataAvailable;
  attribute LazyActiveMode dataStatus;
};
```

CORBA 3
CCM IDL
ModalSP Components

Component IDL

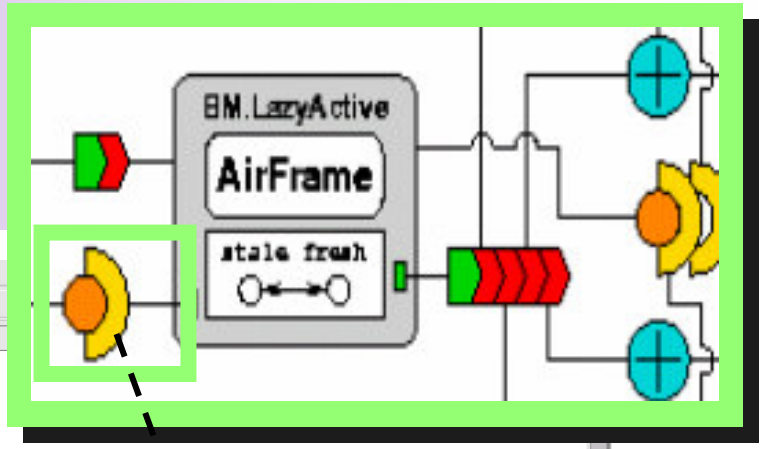



```
component BMLazyActive {
  provides  ReadData outData;
  uses      ReadData inData;
  publishes DataAvailable outDataAvailable;
  consumes  DataAvailable inDataAvailable;
  attribute LazyActiveMode dataStatus;
};
```

output data port (facet)

CORBA 3
CCM IDL
ModalSP Components

Component IDL

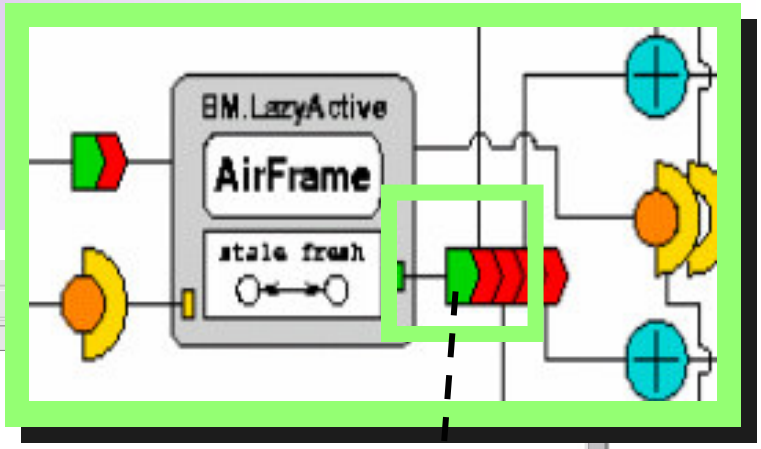



```
component BMLazyActive {
  provides  ReadData outData;
  uses      ReadData inData;
  publishes DataAvailable outDataAvailable;
  consumes  DataAvailable inDataAvailable;
  attribute LazyActiveMode dataStatus;
};
```

input data port (receptacle)

CORBA 3
CCM IDL
ModalSP Components

Component IDL

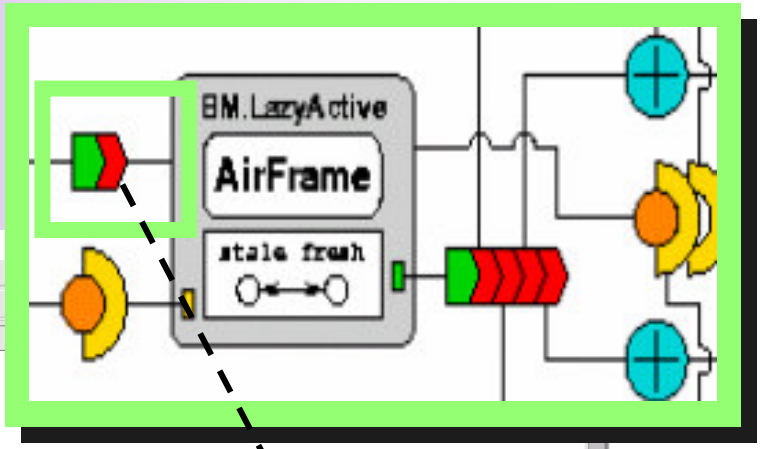



```
component BMLazyActive {
  provides  ReadData outData;
  uses      ReadData inData;
  publishes DataAvailable outDataAvailable;
  consumes  DataAvailable inDataAvailable;
  attribute LazyActiveMode dataStatus;
};
```

output event port (event source)

CORBA 3
CCM IDL
ModalSP Components

Component IDL

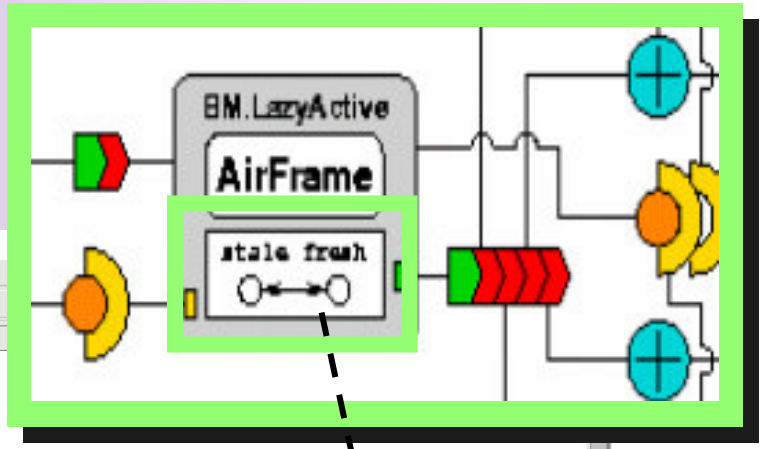



```
component BMLazyActive {
  provides  ReadData outData;
  uses      ReadData inData;
  publishes DataAvailable outDataAvailable;
  consumes  DataAvailable inDataAvailable;
  attribute LazyActiveMode dataStatus;
};
```

input event port (event sink)

CORBA 3
CCM IDL
ModalSP Components

Component IDL

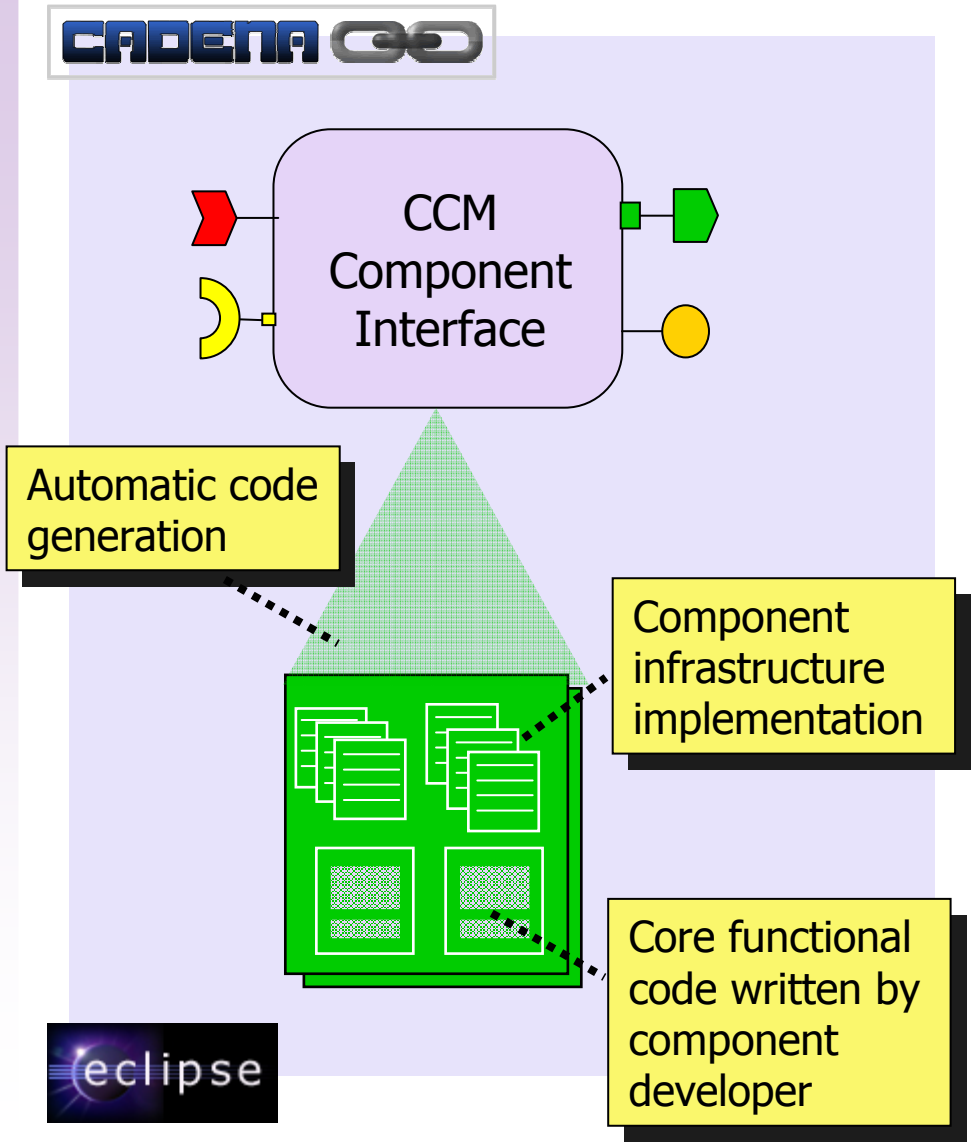



```
component BMLazyActive {
  provides  ReadData outData;
  uses      ReadData inData;
  publishes DataAvailable outDataAvailable;
  consumes  DataAvailable inDataAvailable;
  attribute LazyActiveMode dataStatus;
};
```

mode attribute

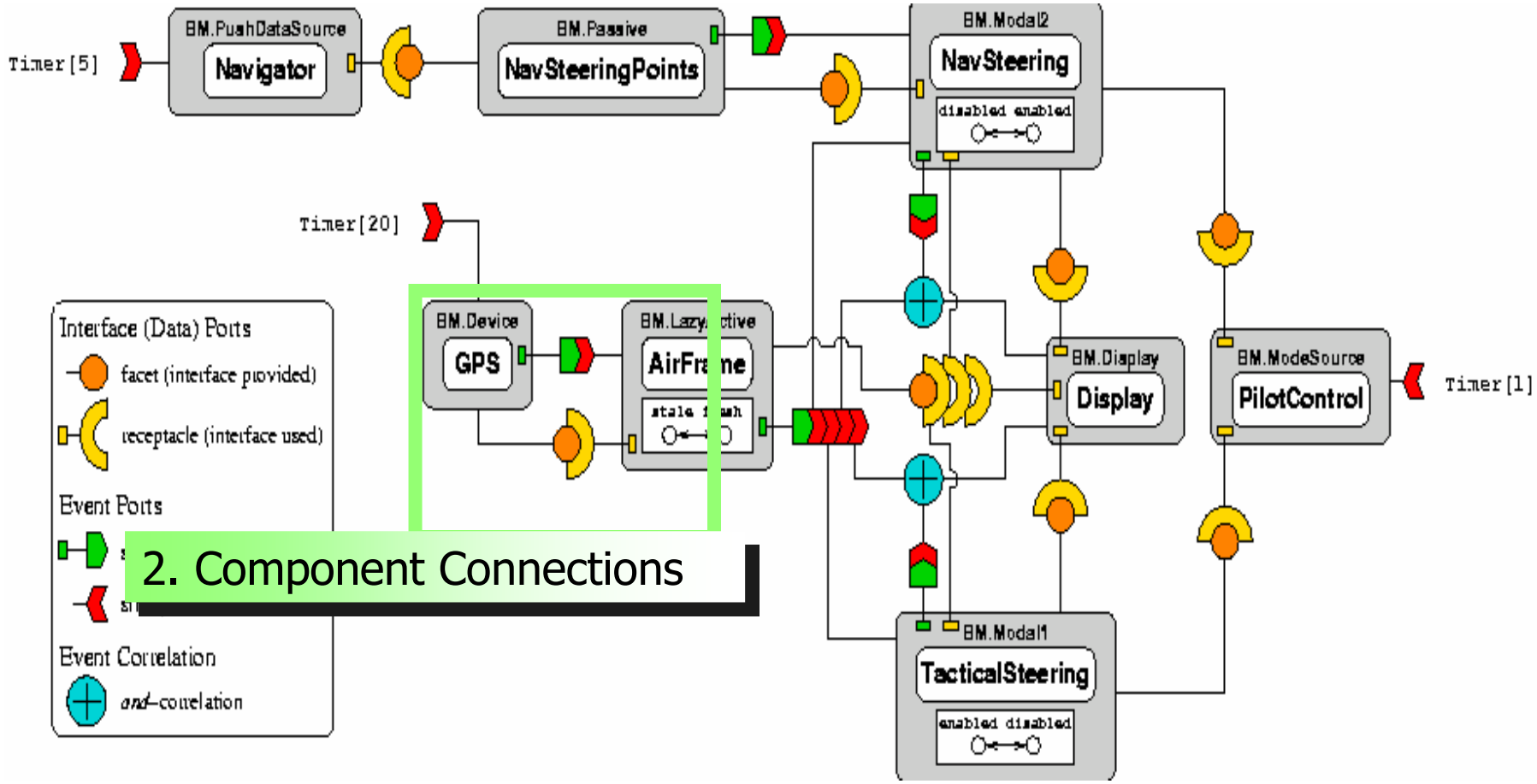
CORBA 3
CCM IDL
ModalSP Components

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

Outline


Timer[5]
BM.PushDataSource
Navigator
BM.Passive
NavSteeringPoints
BM.Modal2
NavSteering
disabled enabled
Timer[20]
BM.Device
GPS
BM.Lazy
AirFrame
stale flush
BM.Display
Display
BM.ModeSource
PilotControl
Timer[1]
BM.Modal1
TacticalSteering
enabled disabled
Interface (Data) Ports
facet (interface provided)
receptacle (interface used)
Event Ports
Event Correlation
and-correlation
2. Component Connections

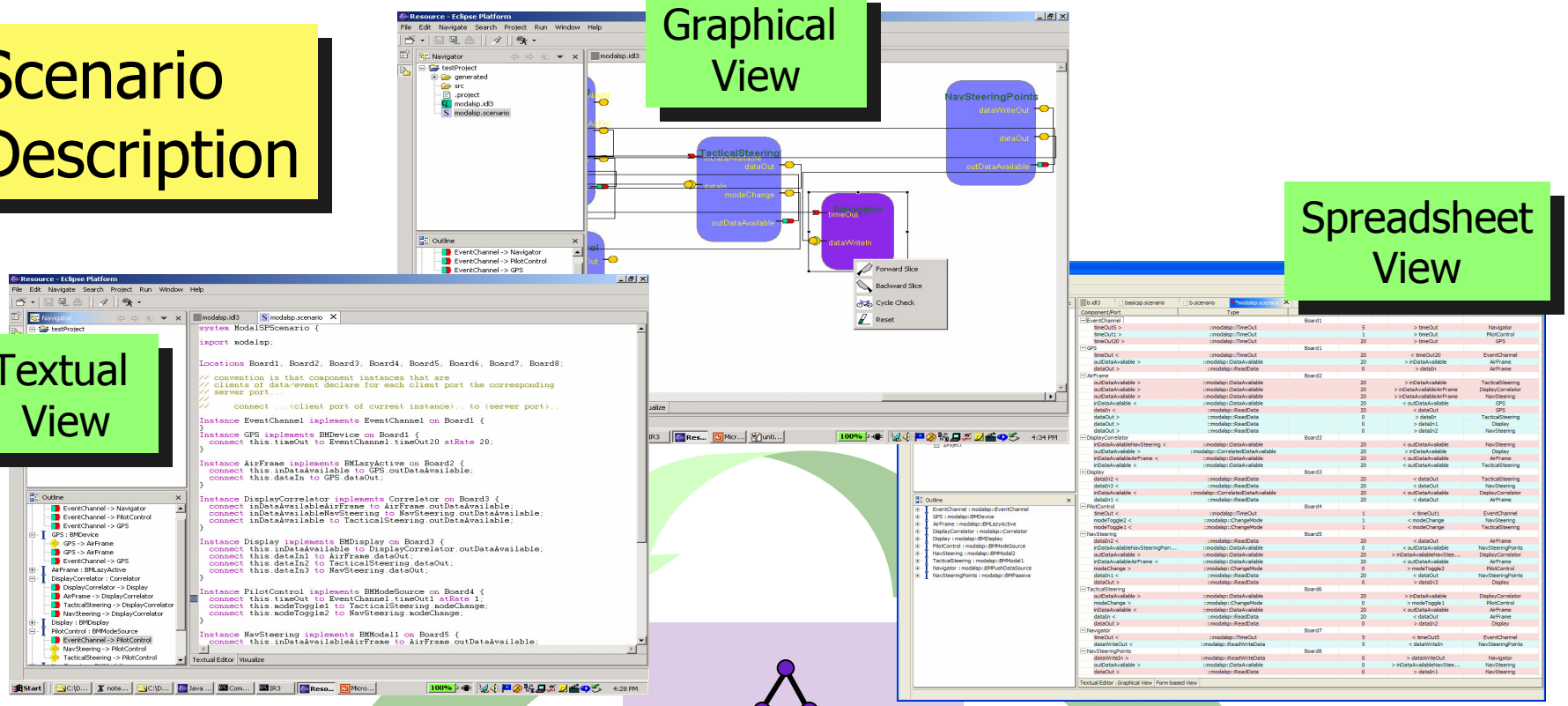
Three Synchronized Views

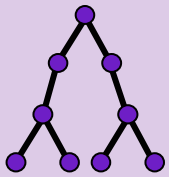
Scenario Description

Graphical View

Textual View

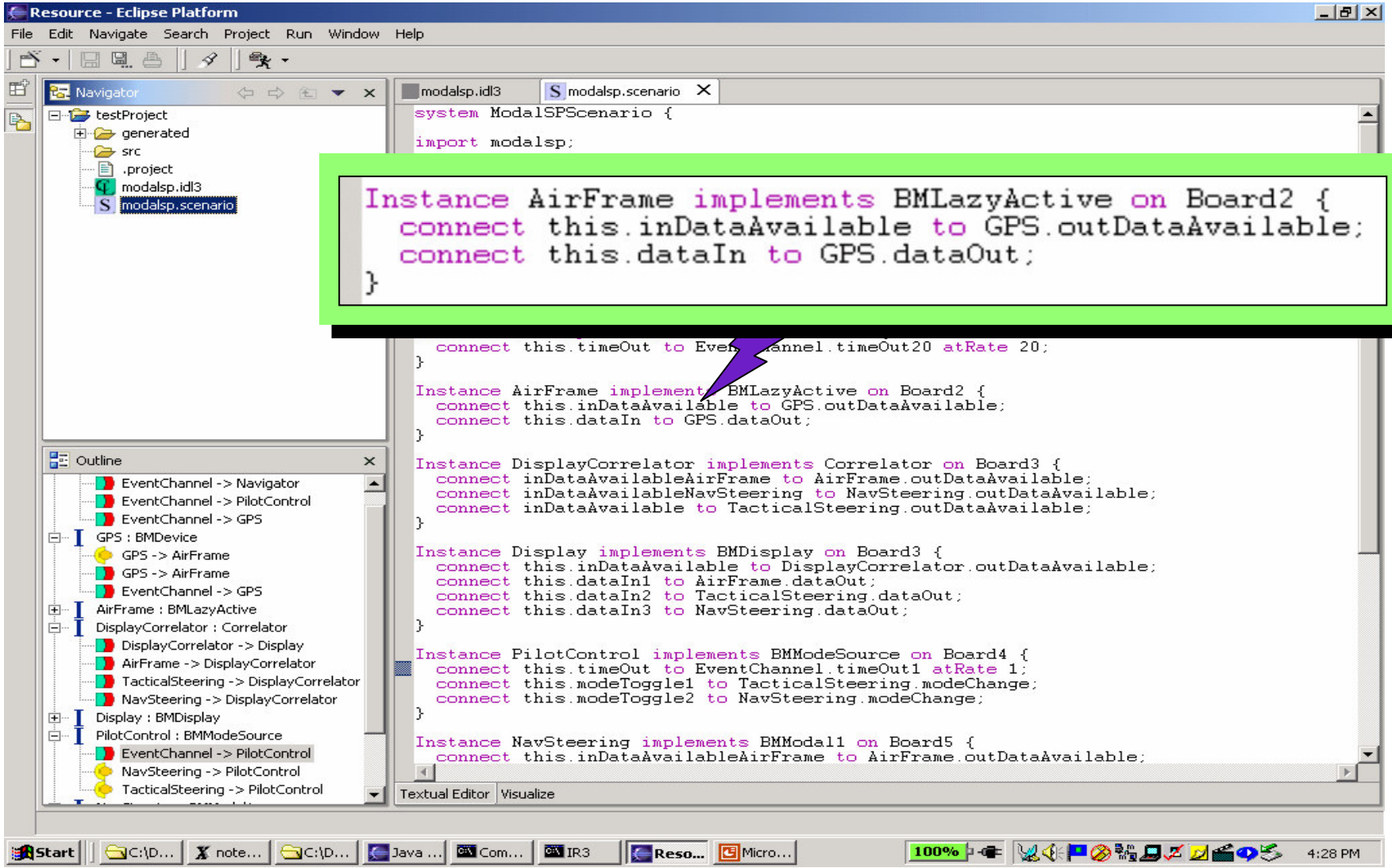
Spreadsheet View



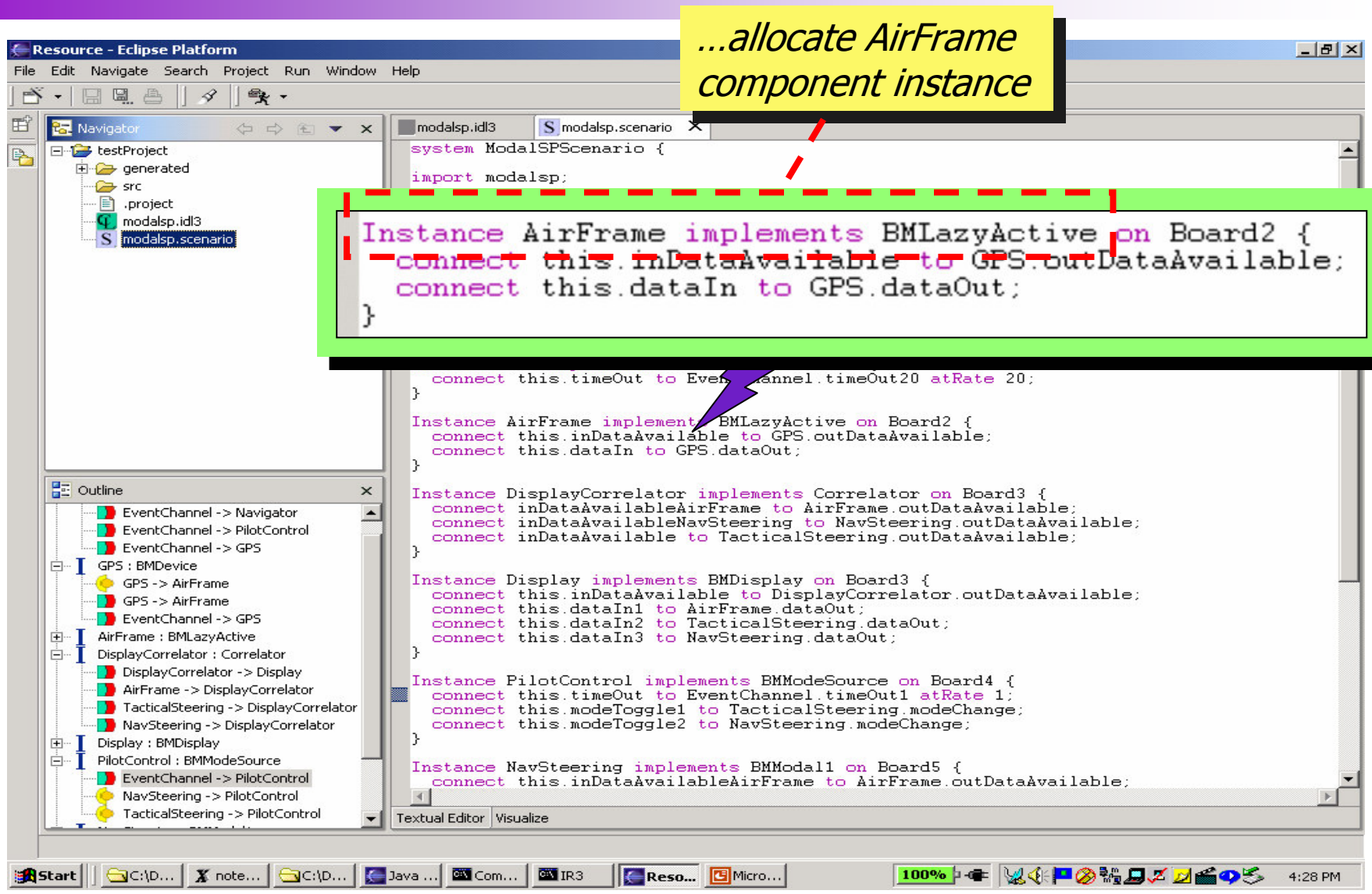


Single Internal Representation

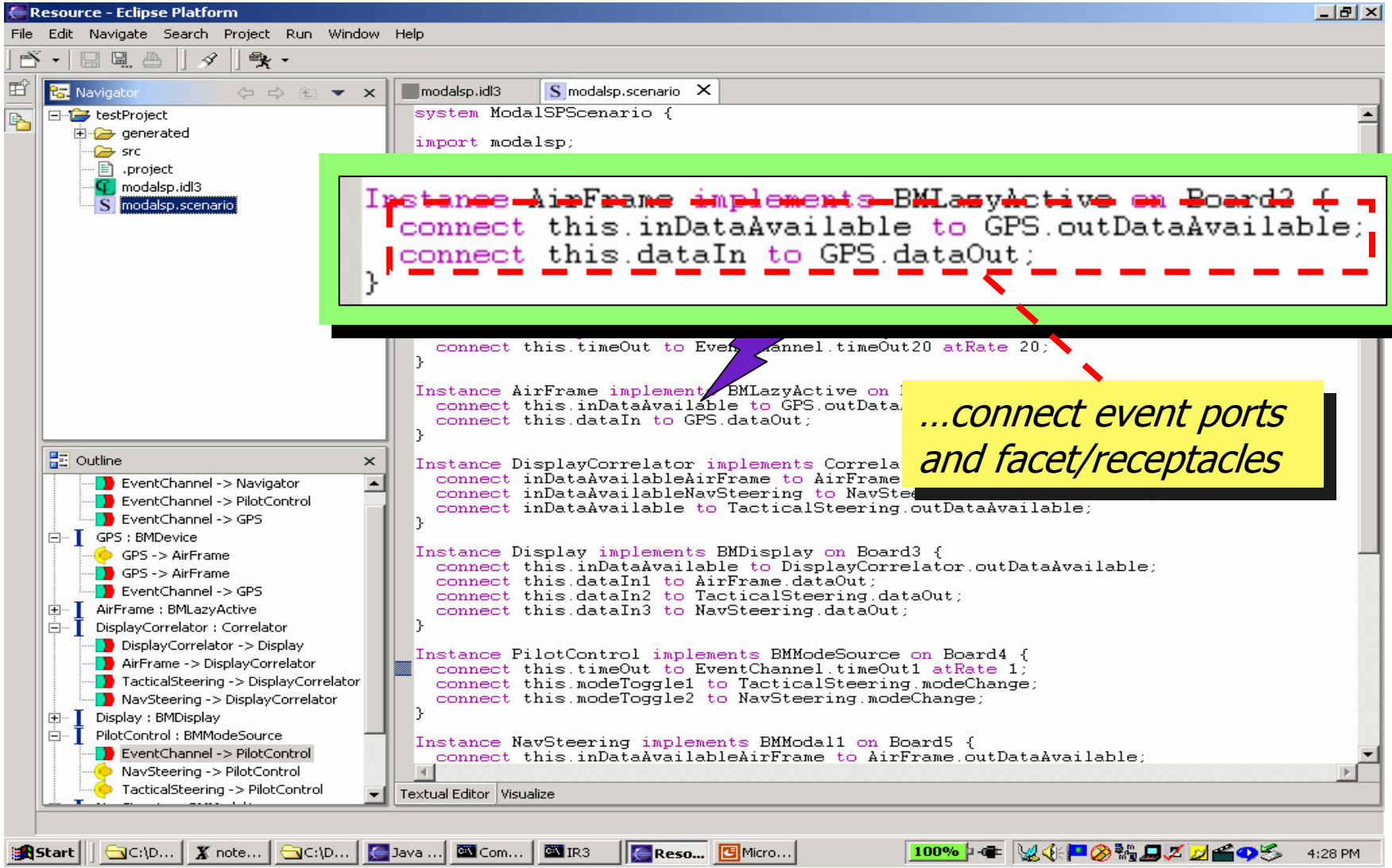
Textual View



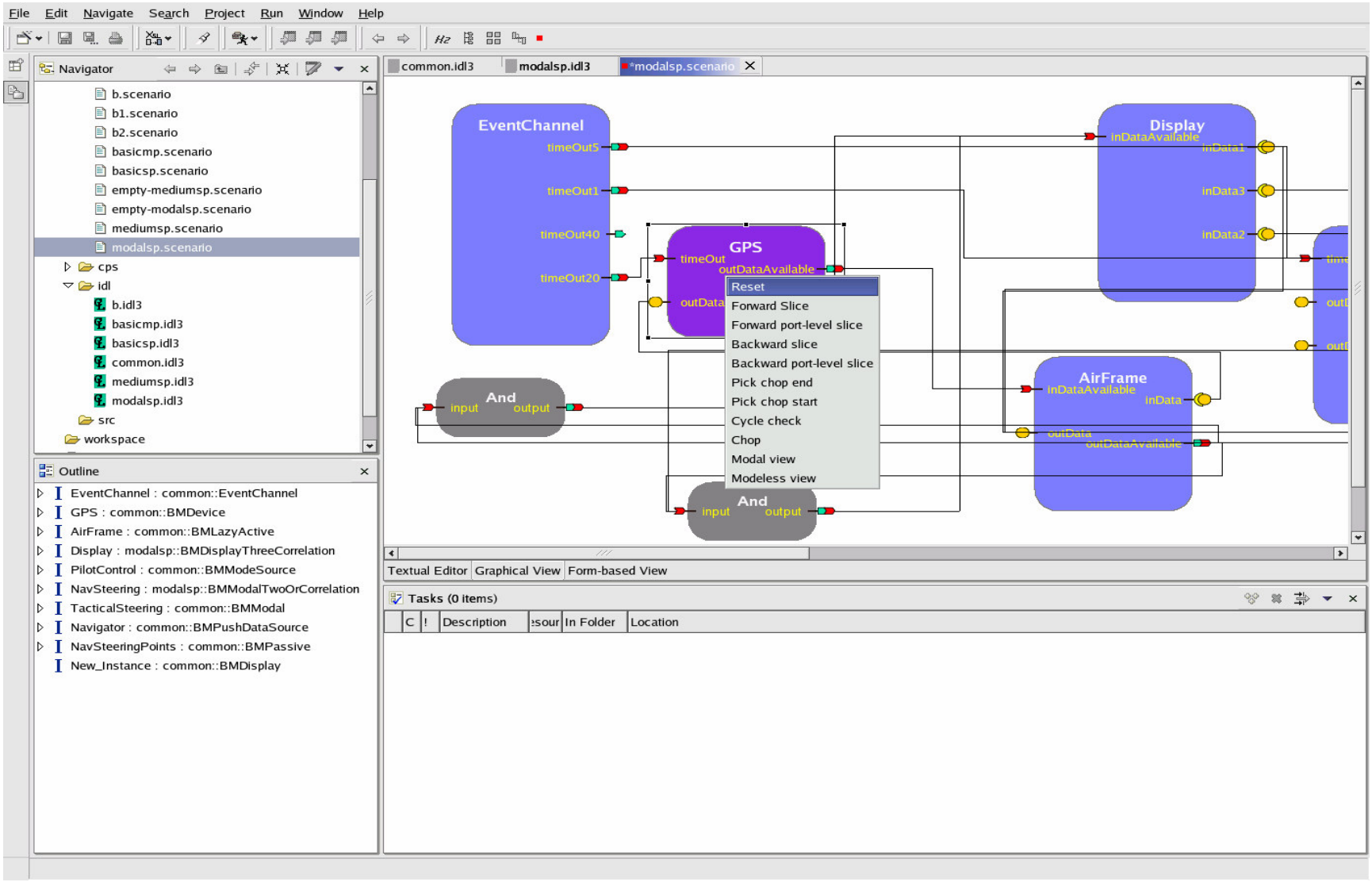
Textual View



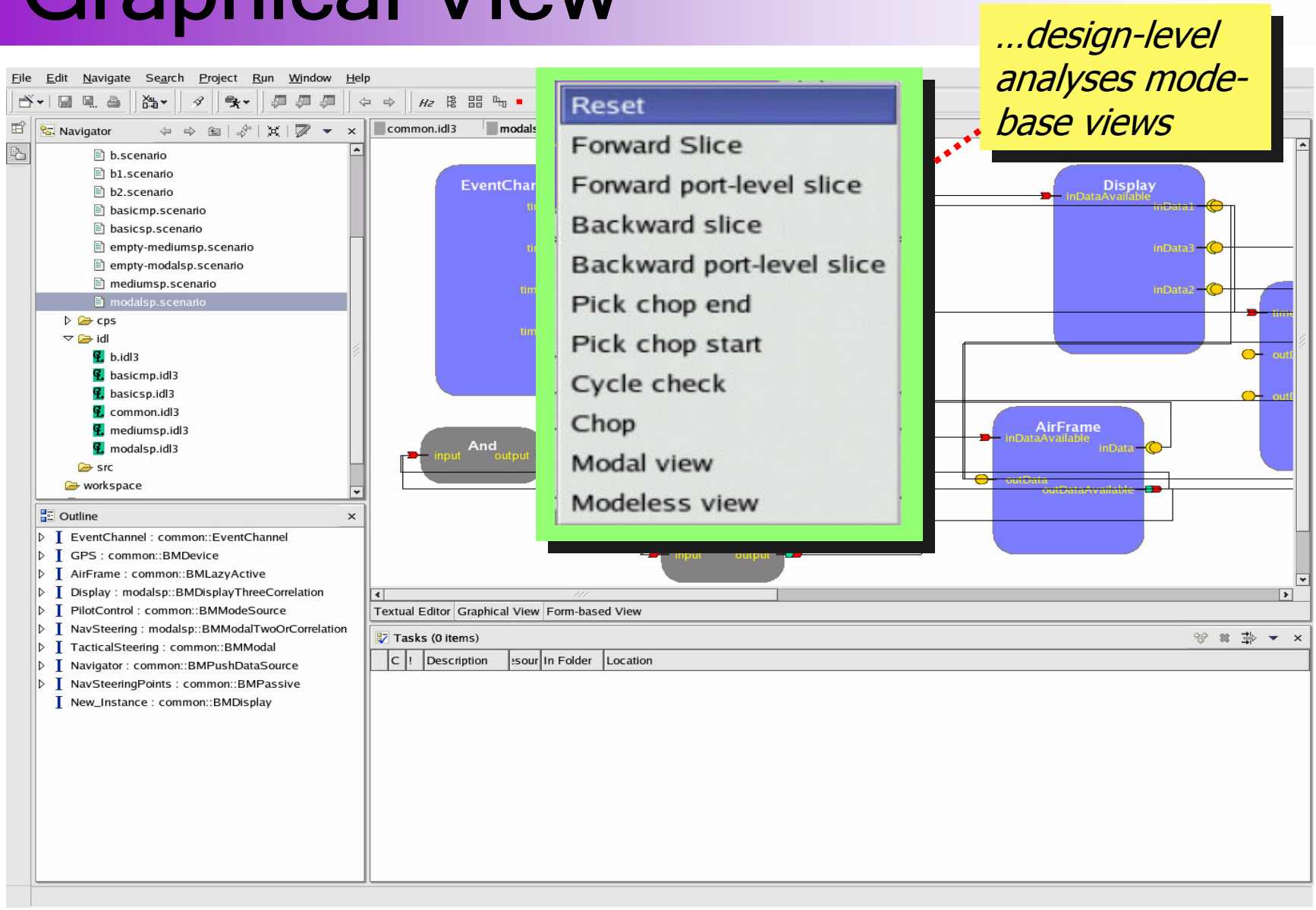
Textual View



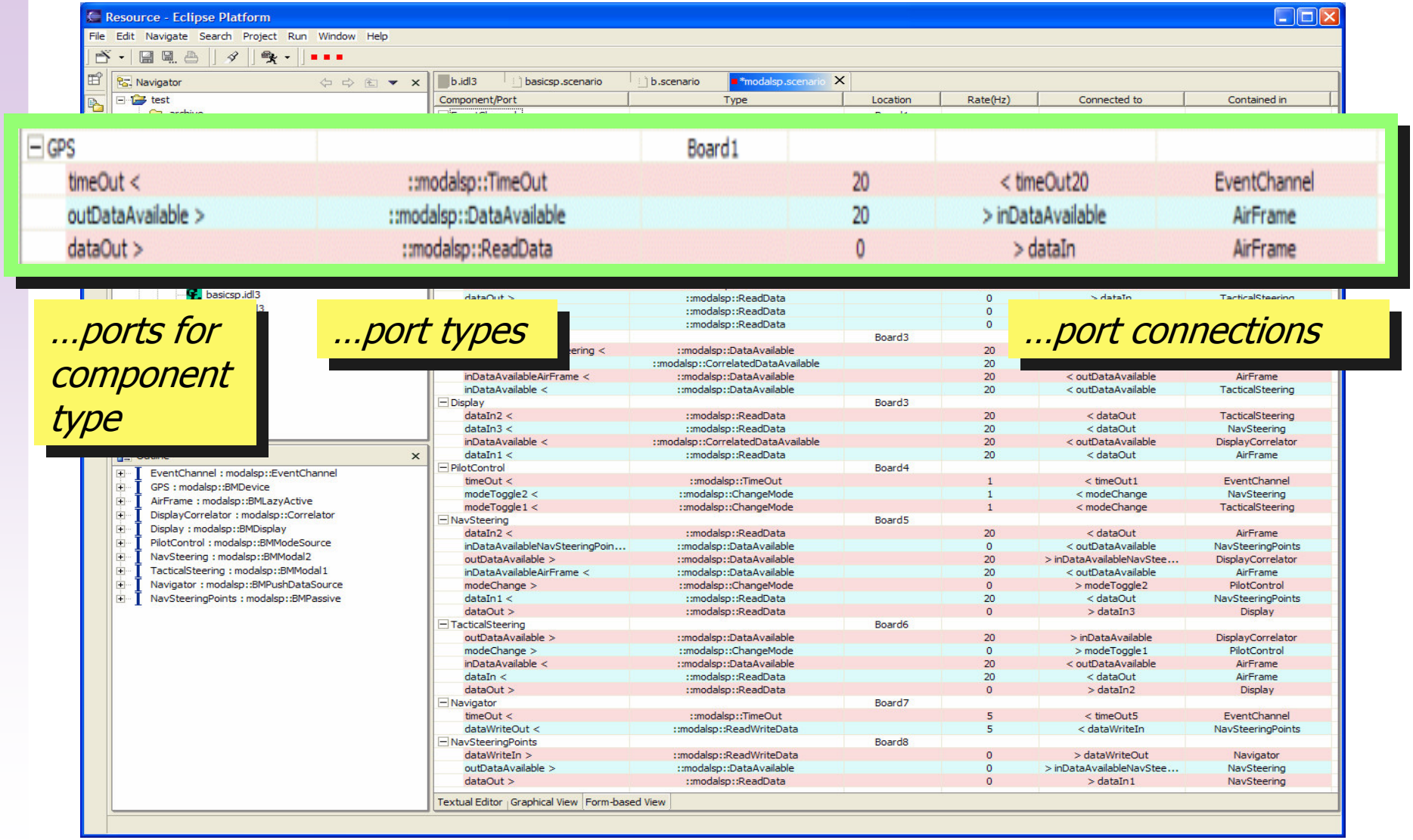
Graphical View



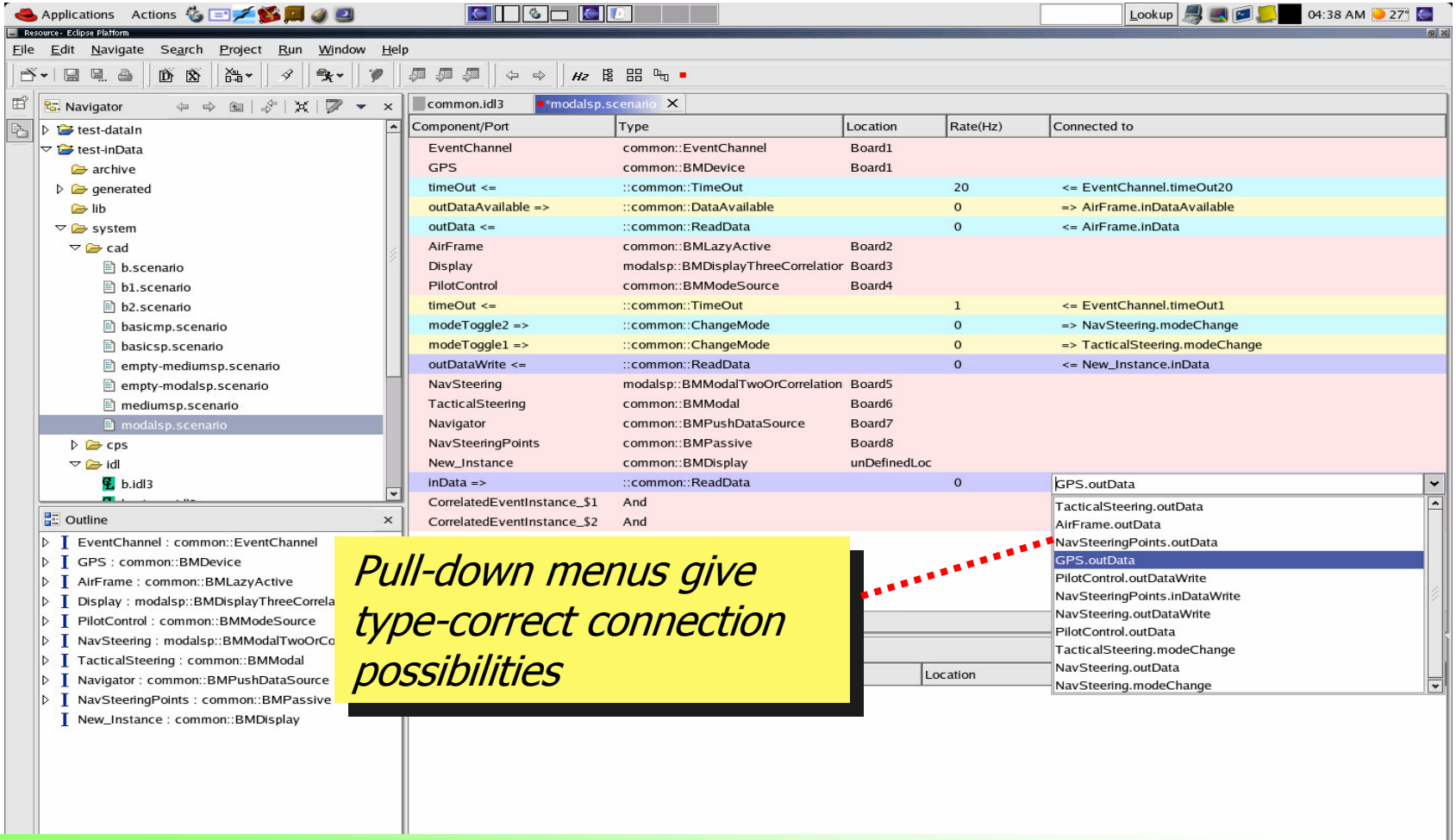
Graphical View



Spreadsheet View

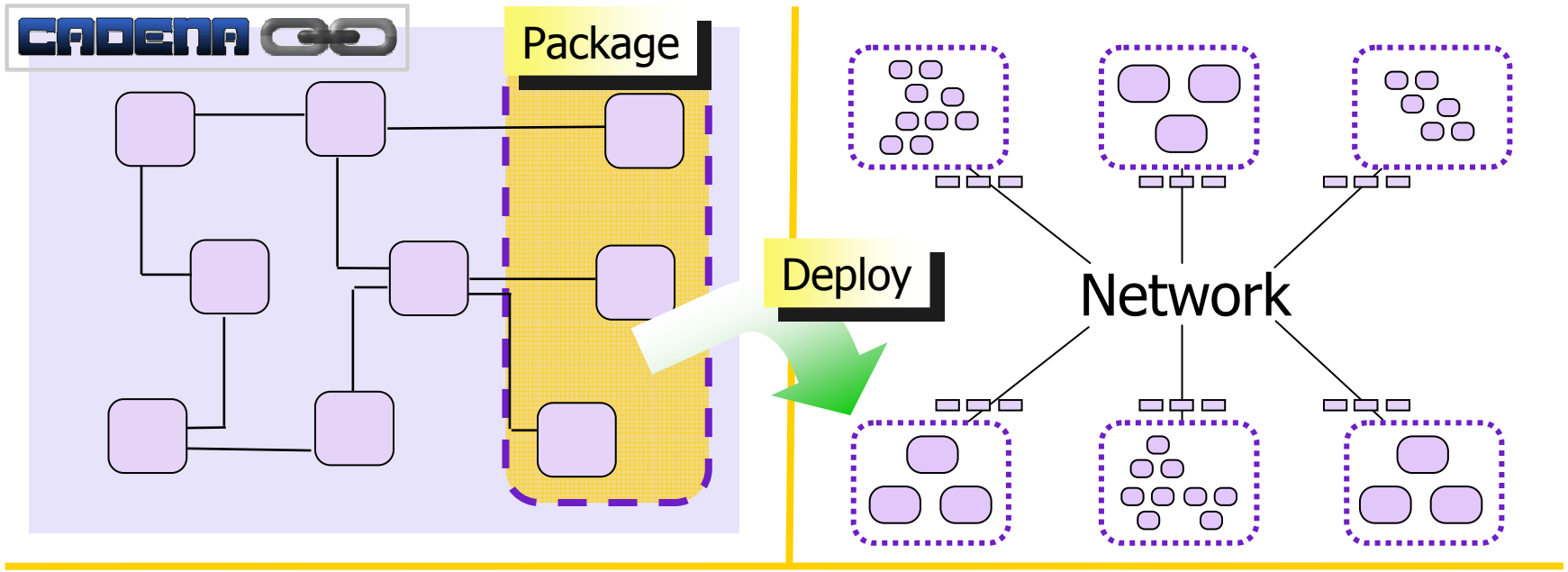


Spreadsheet View



Value Added: *Incremental, iterative* scenario construction with multiple forms of visualization, analyses, and automated "design advice".

Packaging & Deployment



automatic generation

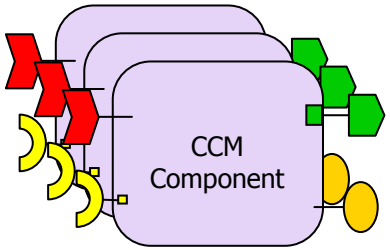
```
<CONFIGURATION_PASS>
  <HOME>          <...>
    <COMPONENT>
    <ID> <...></ID>
    <EVENT_SUPPLIER>
    <...events this component supplies...>
    </EVENT_SUPPLIER>
    </COMPONENT>
    </HOME>
</CONFIGURATION_PASS>
```

CCM XML-based Configuration and Deployment information

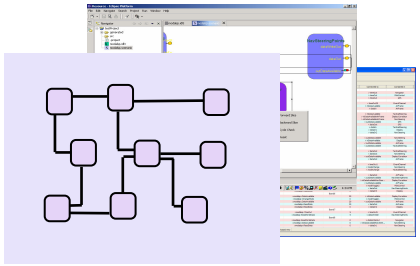
CCM Deployment Infrastructure (e.g., CIAO, OpenCCM)

Assessment

Cadena supports…



Component Interfaces and implementations



Component Assemblies

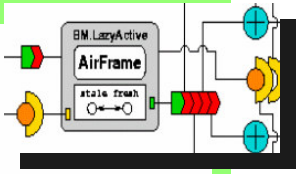
Integration with underlying component middleware frameworks (e.g., CIAO, OpenCCM) through well-defined interfaces (plug in your favorite CCM framework)

…but the *real* power of model-driven development comes from…

- decorating component interfaces and assemblies with a variety of forms of behavioral contracts and meta-data that can be leveraged by model-level analysis and optimizations
- these analyses/optimization may be general purpose, or tailored to specific product-lines or products

Leverage CORBA IDL

```
component BMLazyActive {
 provides  ReadData outData;
 uses      ReadData inData;
 publishes DataAvailable outDataAvailable;
 consumes  DataAvailable inDataAvailable;
 attribute LazyActiveMode dataStatus;
};
```



+

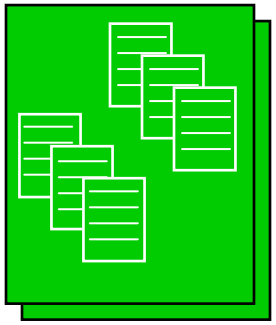
```
dependencydefault
   == none;

dependencies {
 inDataAvailable
   ->
  outDataAvailable;
}
```

Dependency Annotations

```
behavior {
 if (mode==enabled) {
  push outDataAvailable;
 else
  …
 }
```

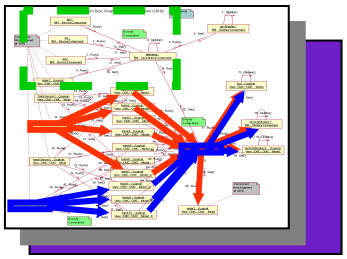
Transition System Semantics

IDL Compiler



Component Implementation Stubs & Skeletons

Model Builder



Dependency Analysis and Model-checking Engine

Incremental Specification

Increasing Effort & Strength of Verification

Specifications

port action dependencies

refinement

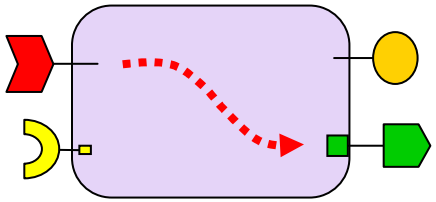
state-based dependencies

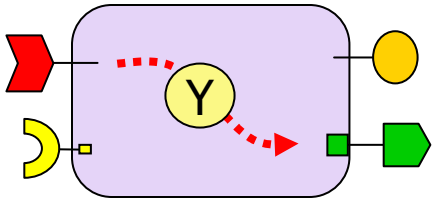
refinement

component transition semantics

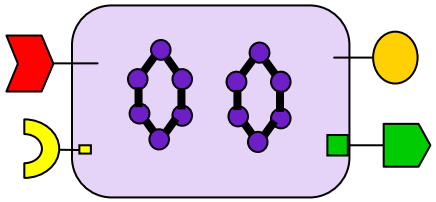
Lockheed-Martin collaboration: extract/connect this info from MSCs

Component Structure



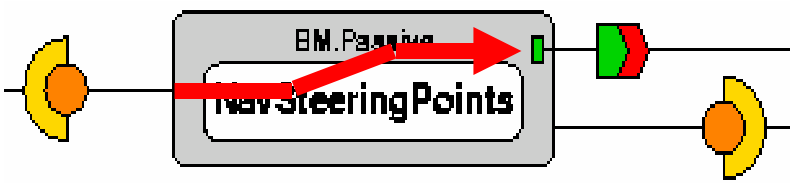


...only in mode Y



...state machines give abstract behavior

Light-weight Dependency Specs




```
dependencydefault == none;

dependencies {
 dataWriteOut.set_data() -> outDataAvailable;
}

behavior { ...  }
```

call on set_data()

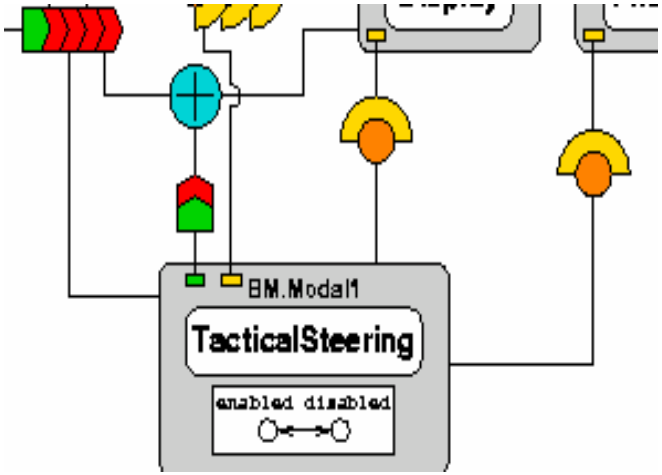
triggers

outDataAvailable port action

Light-weight Dependency Specs

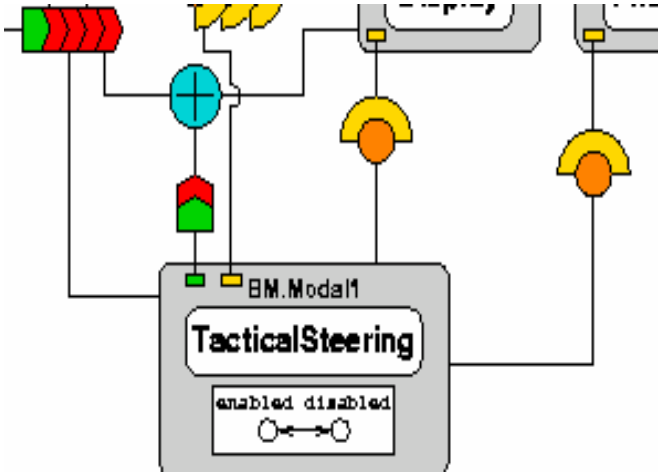
triggers no other actions

```
dependencydefault == all;

dependencies {
 modeChange() ->;
 case modeChange.modeVar of {
  enabled:  inDataAvailable
            -> dataIn.get_data(),
               outDataAvailable;
  disabled: inDataAvailable ->;
 }
}

behavior { ...  }
```



Light-weight Dependency Specs

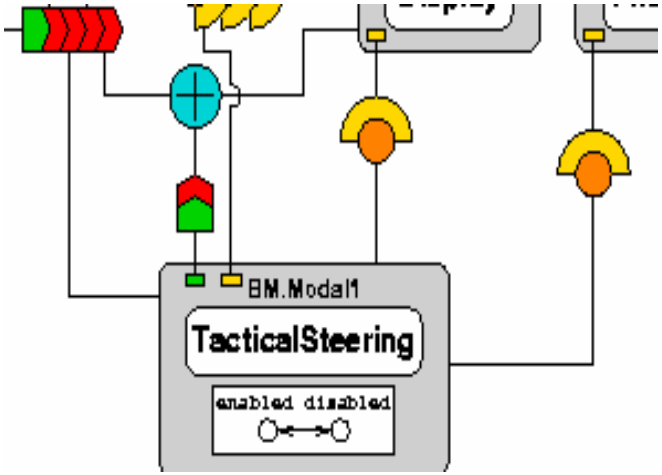
```
dependencydefault == all;

dependencies {
 modeChange() ->;
 case modeChange.modeVar of {
  enabled:  inDataAvailable
            -> dataIn.get_data(),
               outDataAvailable;
  disabled: inDataAvailable ->;
 }
}

behavior { ...  }
```



in enabled mode, shows actions triggered by receipt of event on inDataAvailable port

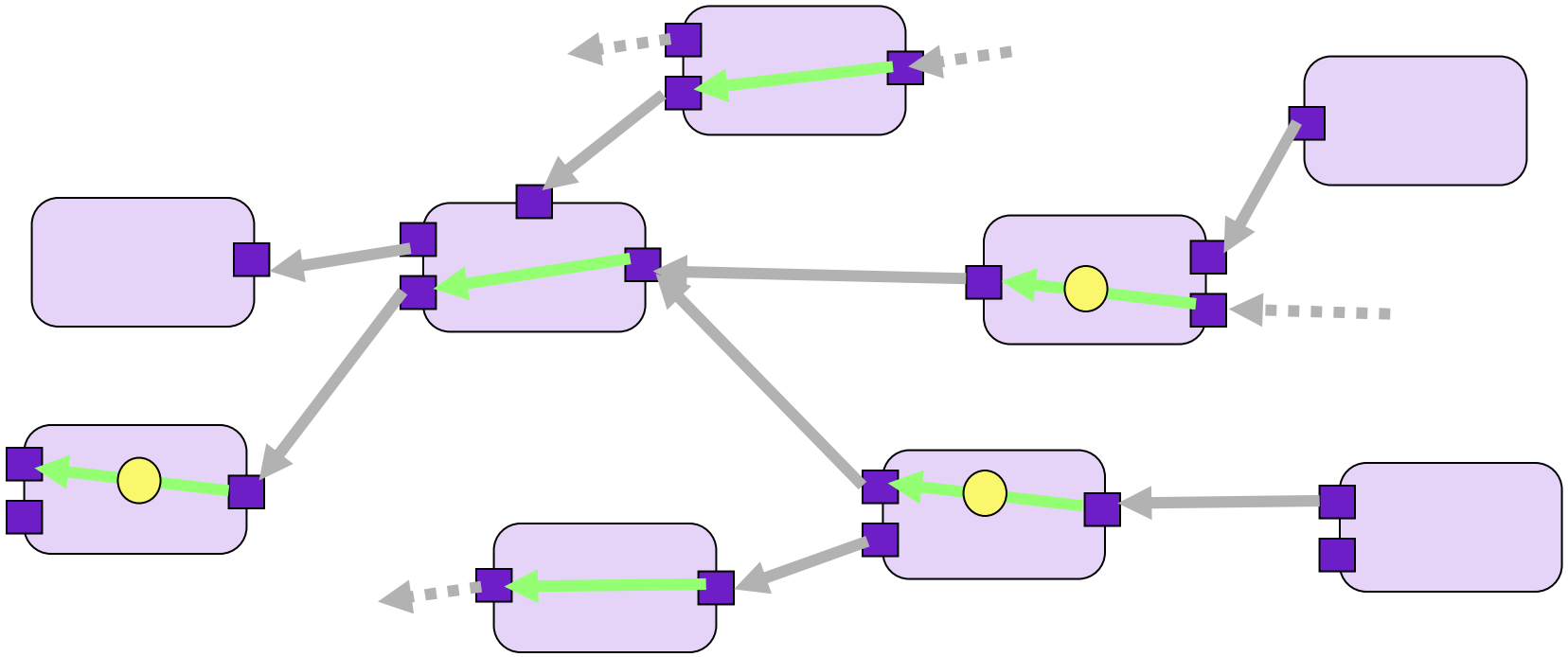
Light-weight Dependency Specs

```
dependencydefault == all;

dependencies {
 modeChange() ->;
 case modeChange.modeVar of {
  enabled:  inDataAvailable
            -> dataIn.get_data(),
               outDataAvailable;
  disabled: inDataAvailable ->;
 }
}

behavior { ...  }
```



in disabled mode, inDataAvailable triggers no other port actions

Design-Level Flow/Dependence Graphs



From system configuration information

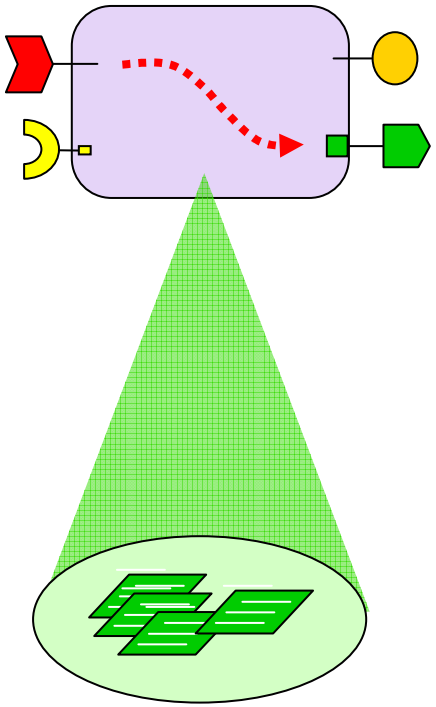
From user-specified intra-component dependences

State predicates giving conditional dependences

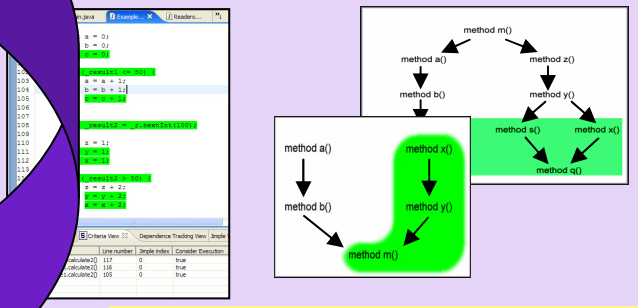
Theme

Connecting Models & Code

Model-level Specification/Abstraction



…use Indus to automatically *check* that code dependences satisfy specification



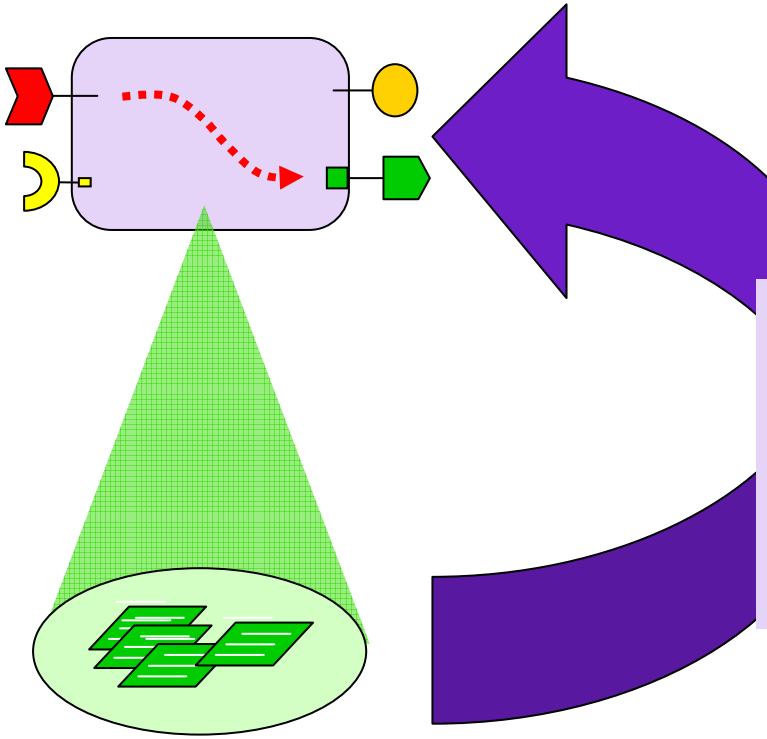
SAnToS Indus Dependence Calculation (Java Program Slicing)

Java Implementation

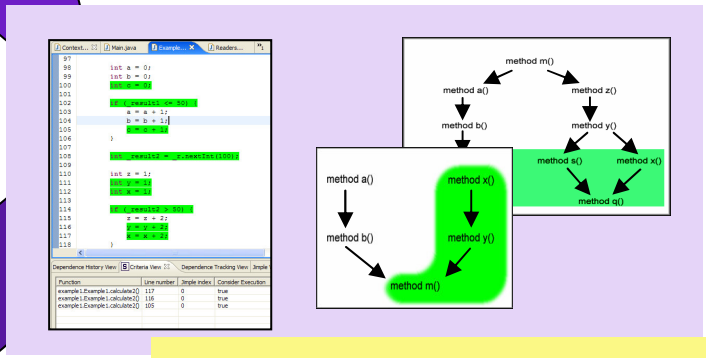
Theme

Connecting Models & Code

Model-level Specification/Abstraction



…use Indus to automatically *synthesize* (reverse engineer) model-level dependences from code



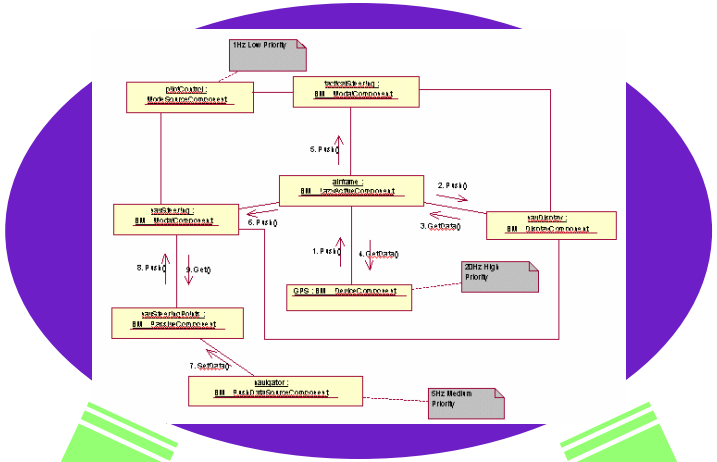
SAnToS Indus Dependence Calculation (Java Program Slicing)

Java Implementation

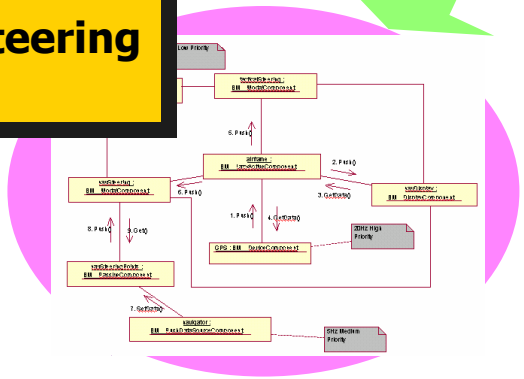
Simplifying Assembly Visualization

Goal: Simplify visualization of system by hiding component/connections in various modes.

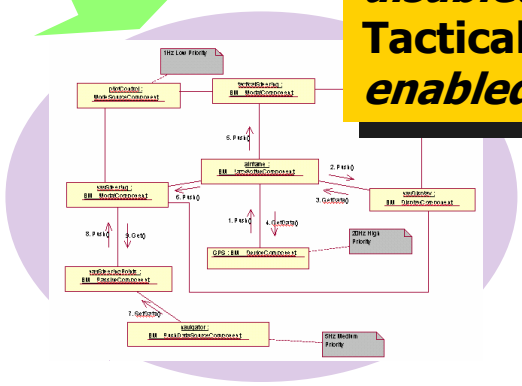
Scenario Diagram w/ Complete Connectivity



NavSteering *enabled*
TacticalSteering *disabled*



. . .

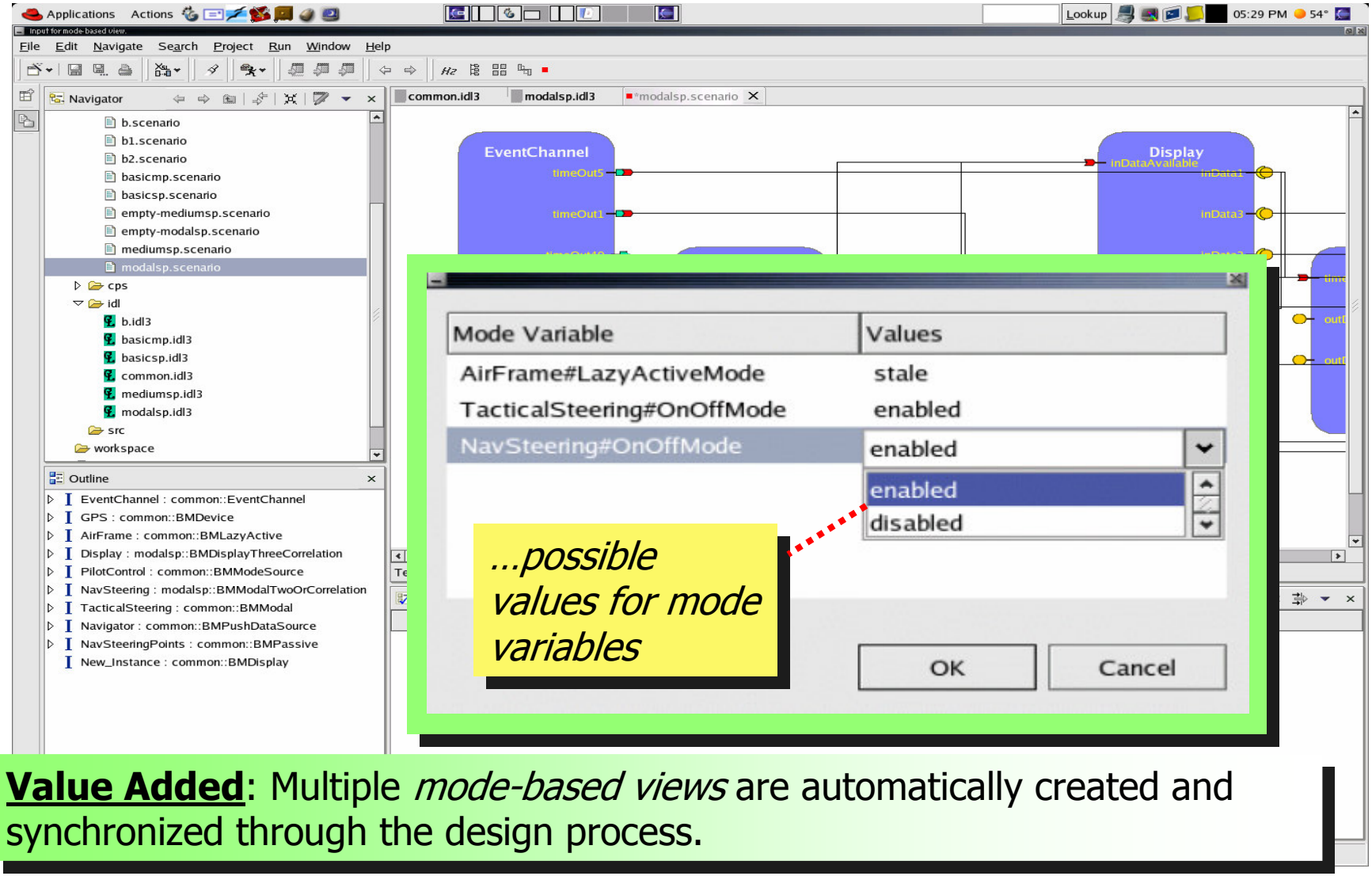


NavSteering *disabled*
TacticalSteering *enabled*

Projections of Systems with Enabled Connectivity for Different Modes

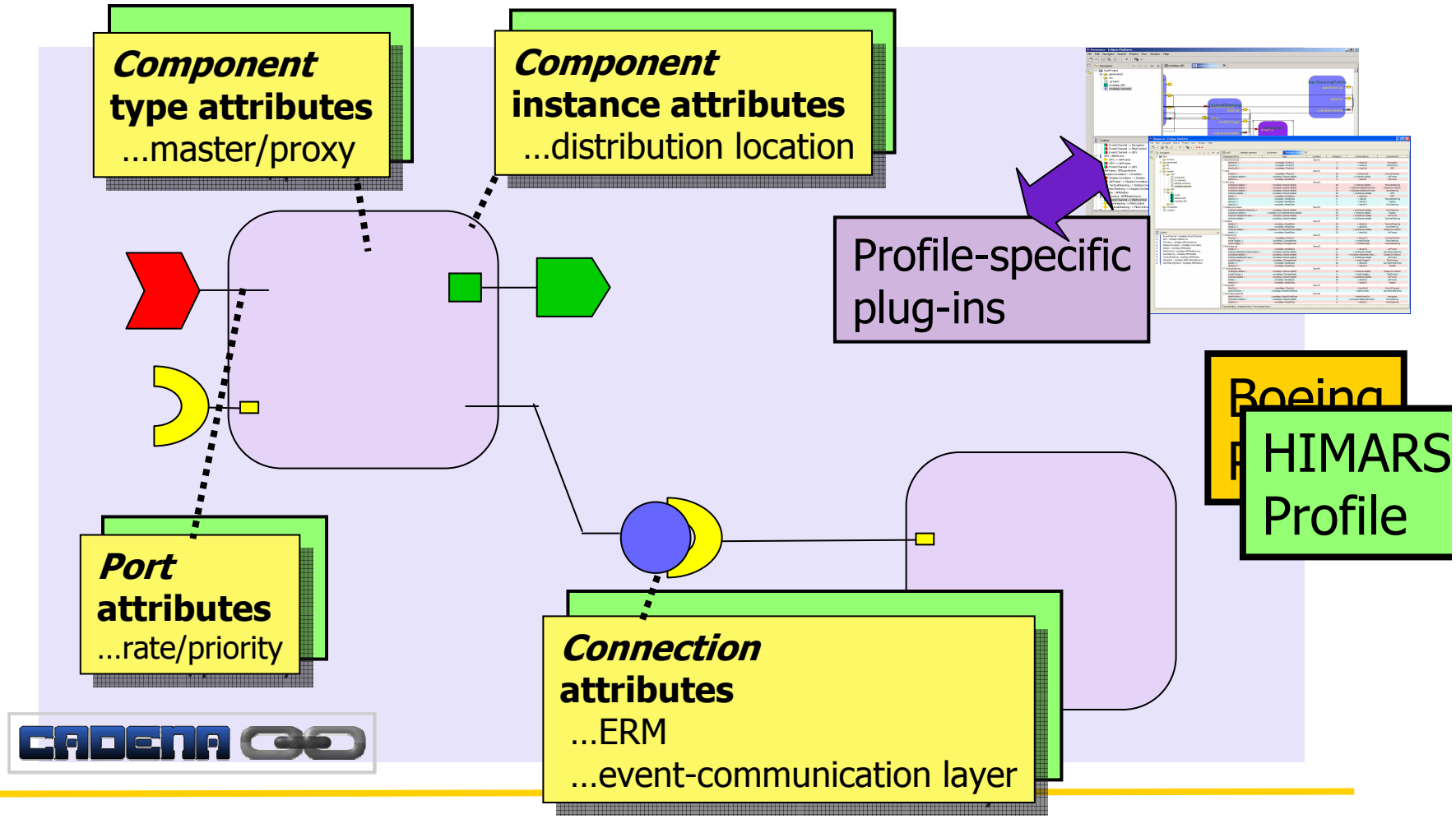
Automatic!

Mode-based Projections



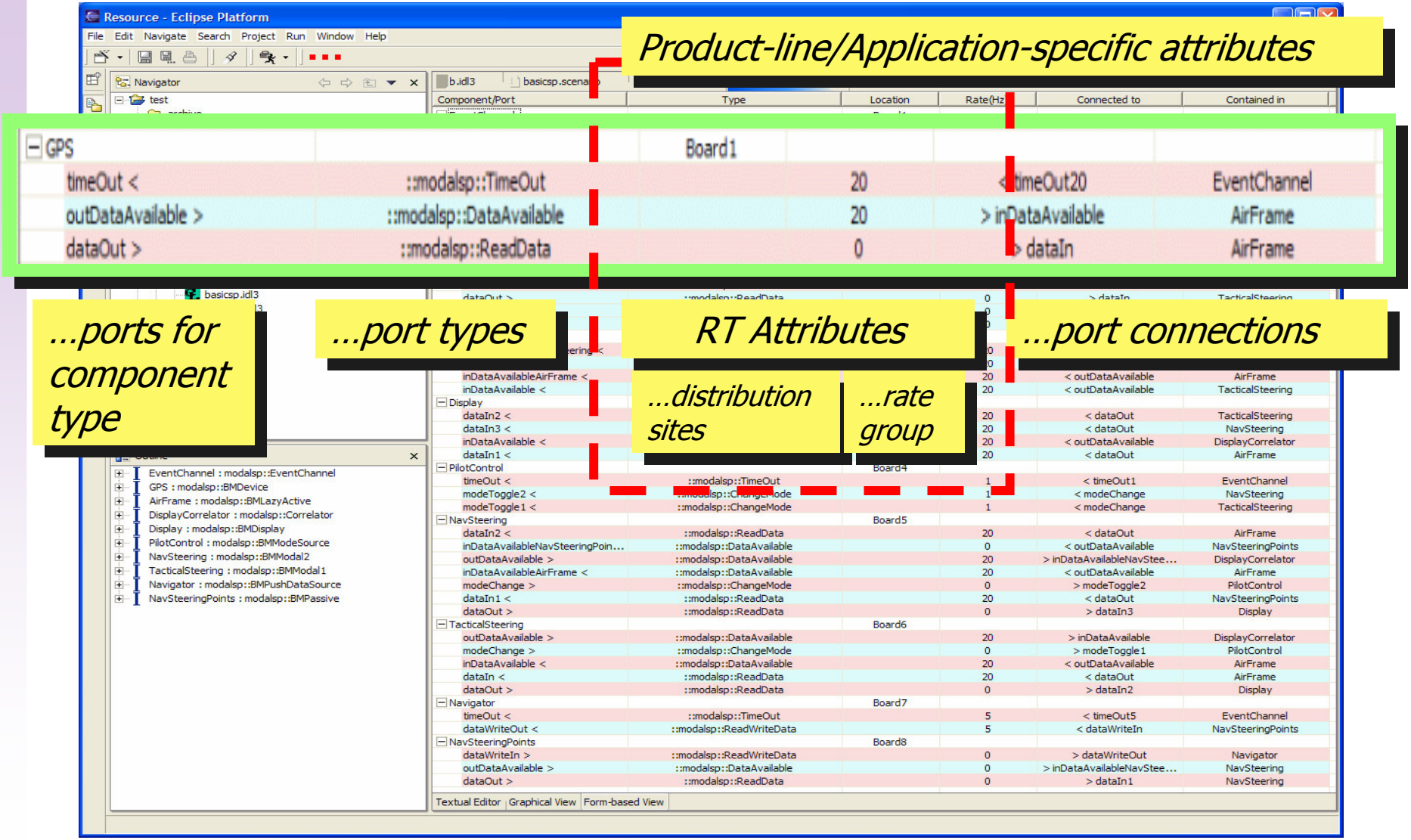
Value Added: Multiple *mode-based views* are automatically created and synchronized through the design process.

Configurable Product Line Profiles



Cadena **profiles** enable flexible definition of attributes for CCM model entities and APIs for plug-in tools to access and manipulate attribute values

Spreadsheet View

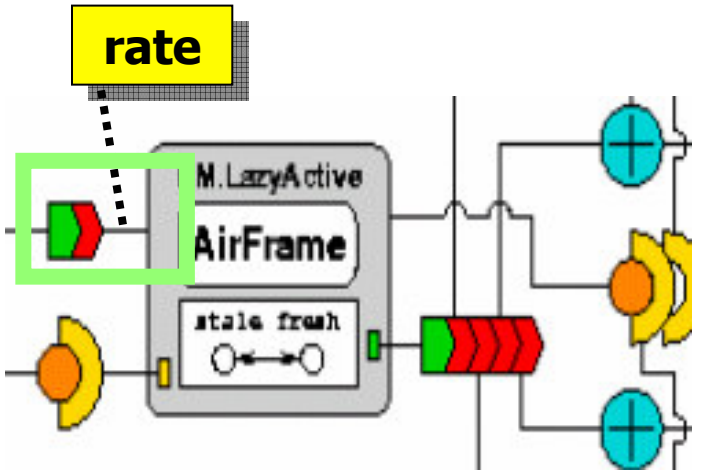


Automated Design Advice

Dependency-driven priority/rate assignment to event handlers

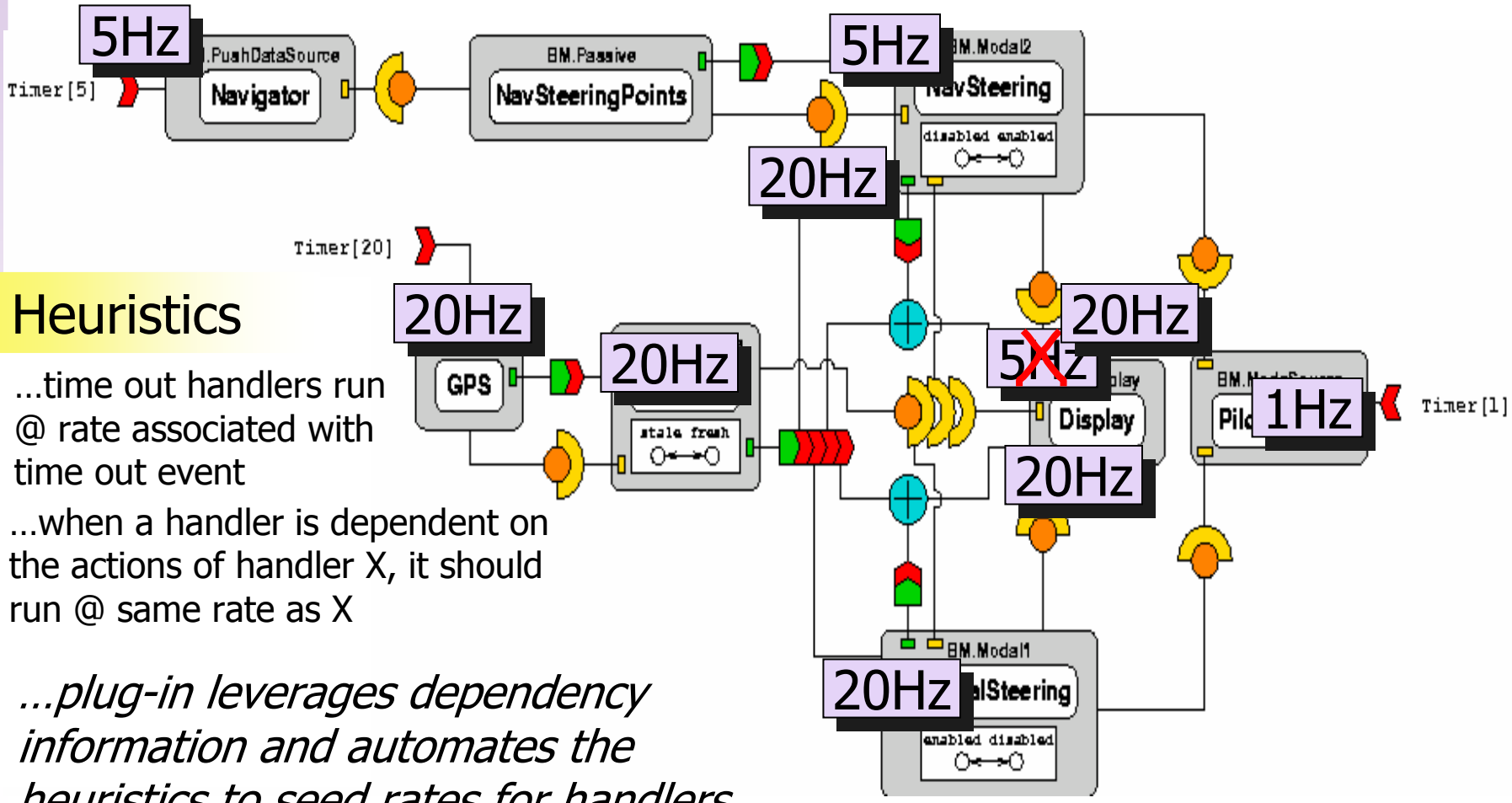
- Bold Stroke product-line architecture contains all threads in RT CORBA event channel
  - rate monotonic scheduling, so a group of thread where each thread has a certain priority/rate
- Each event handler has a model-level attribute to indicate the priority/rate of the thread that should run the handler

developer must select attribute value…



Automated Design Advice

Dependency-driven priority/rate assignment to event handlers



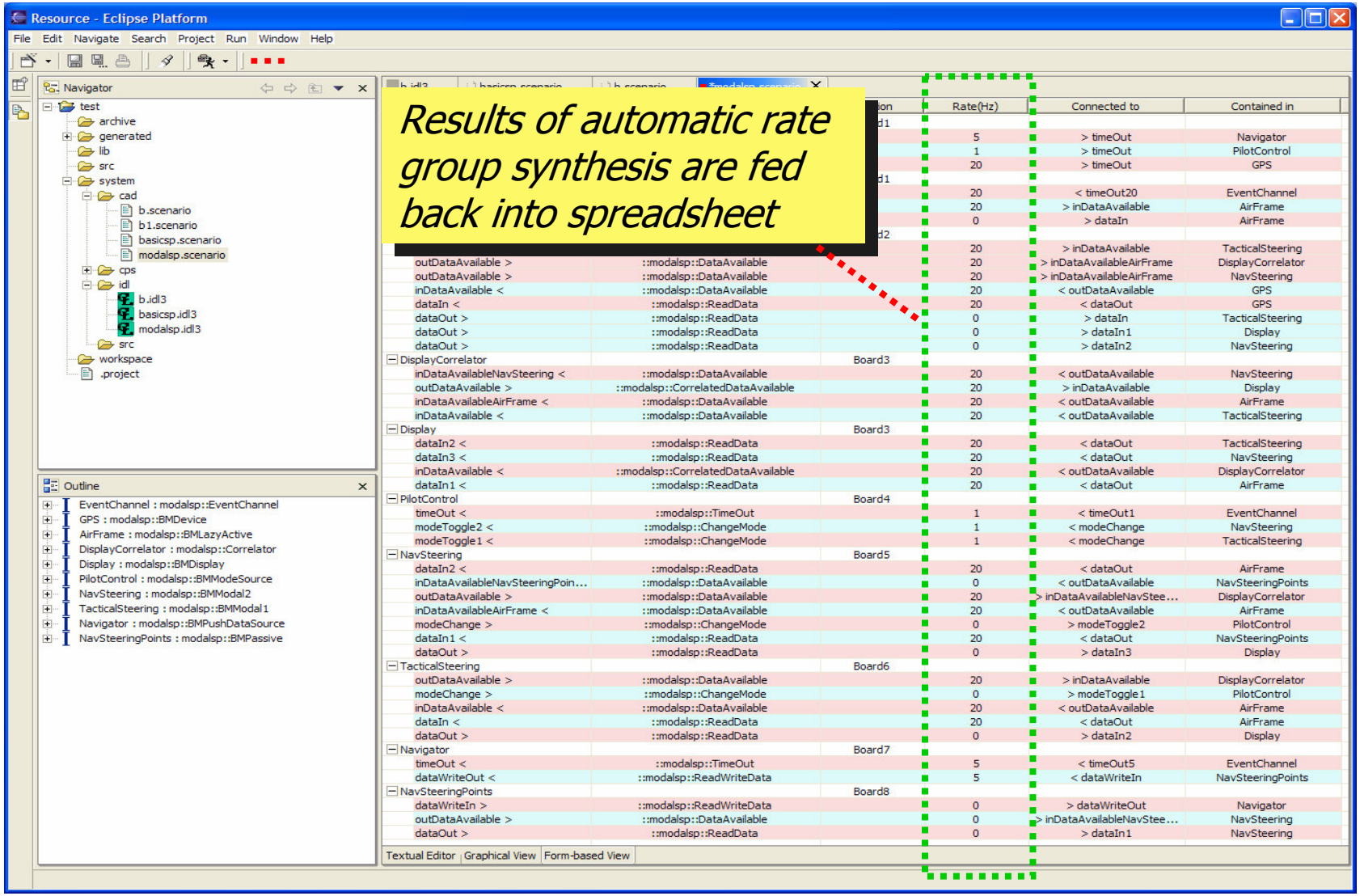
Heuristics

…time out handlers run @ rate associated with time out event

…when a handler is dependent on the actions of handler X, it should run @ same rate as X

…plug-in leverages dependency information and automates the heuristics to seed rates for handlers

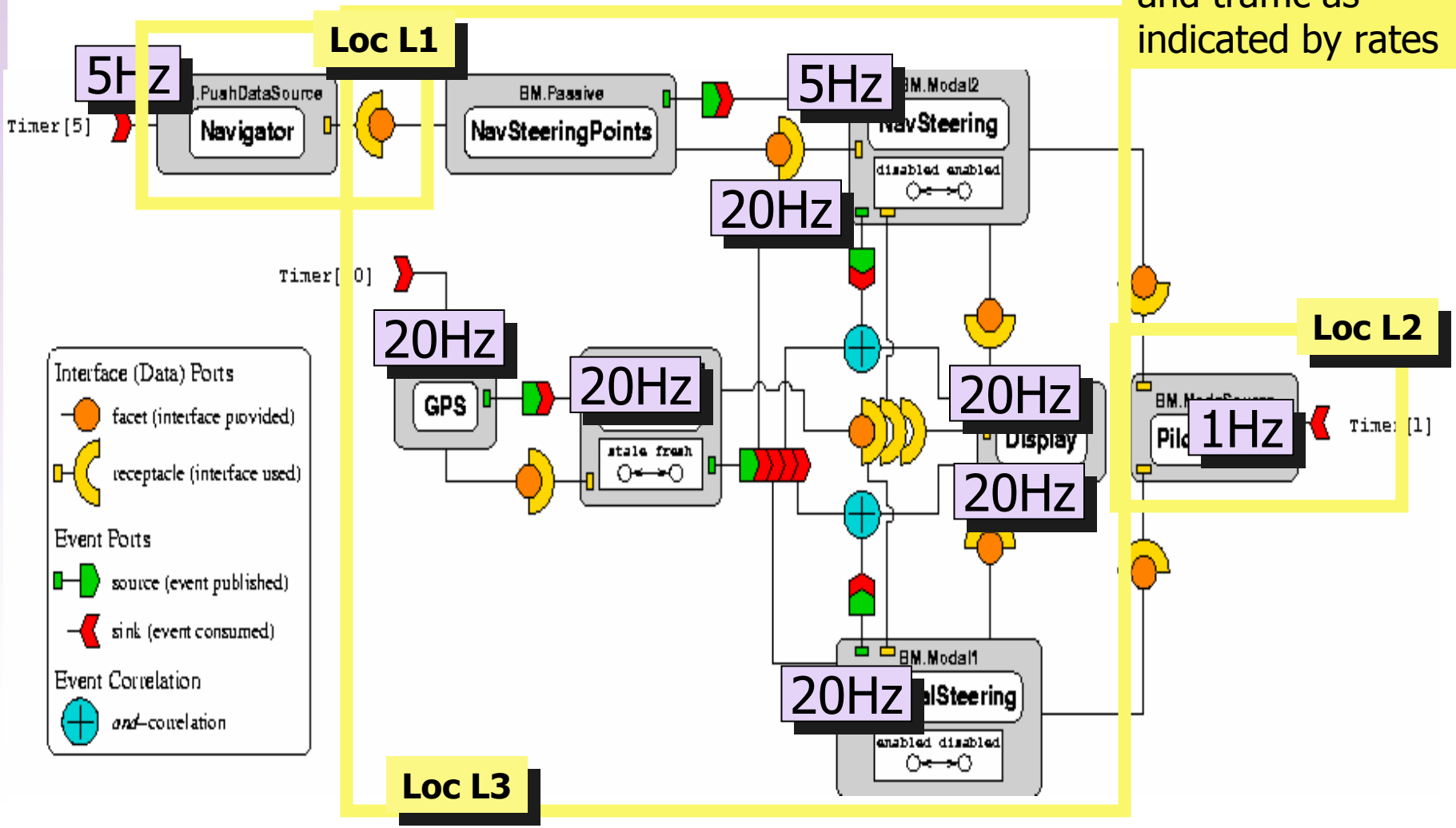
Spreadsheet View



Automated Design Advice

Synthesis of distribution information

Look at coupling and traffic as indicated by rates

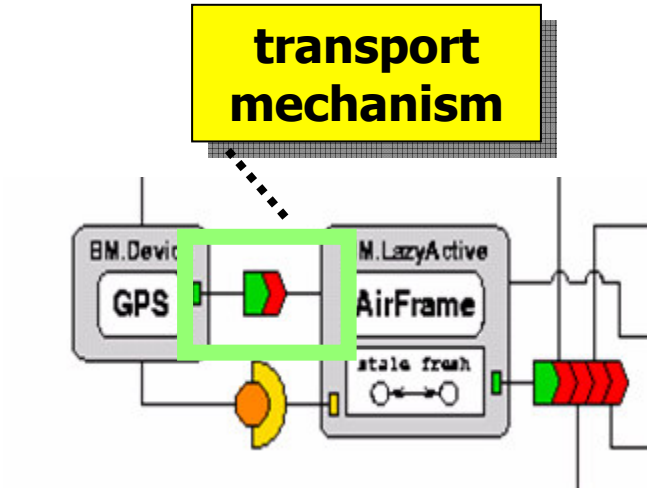


Automated Design Advice

Selecting event transport facility

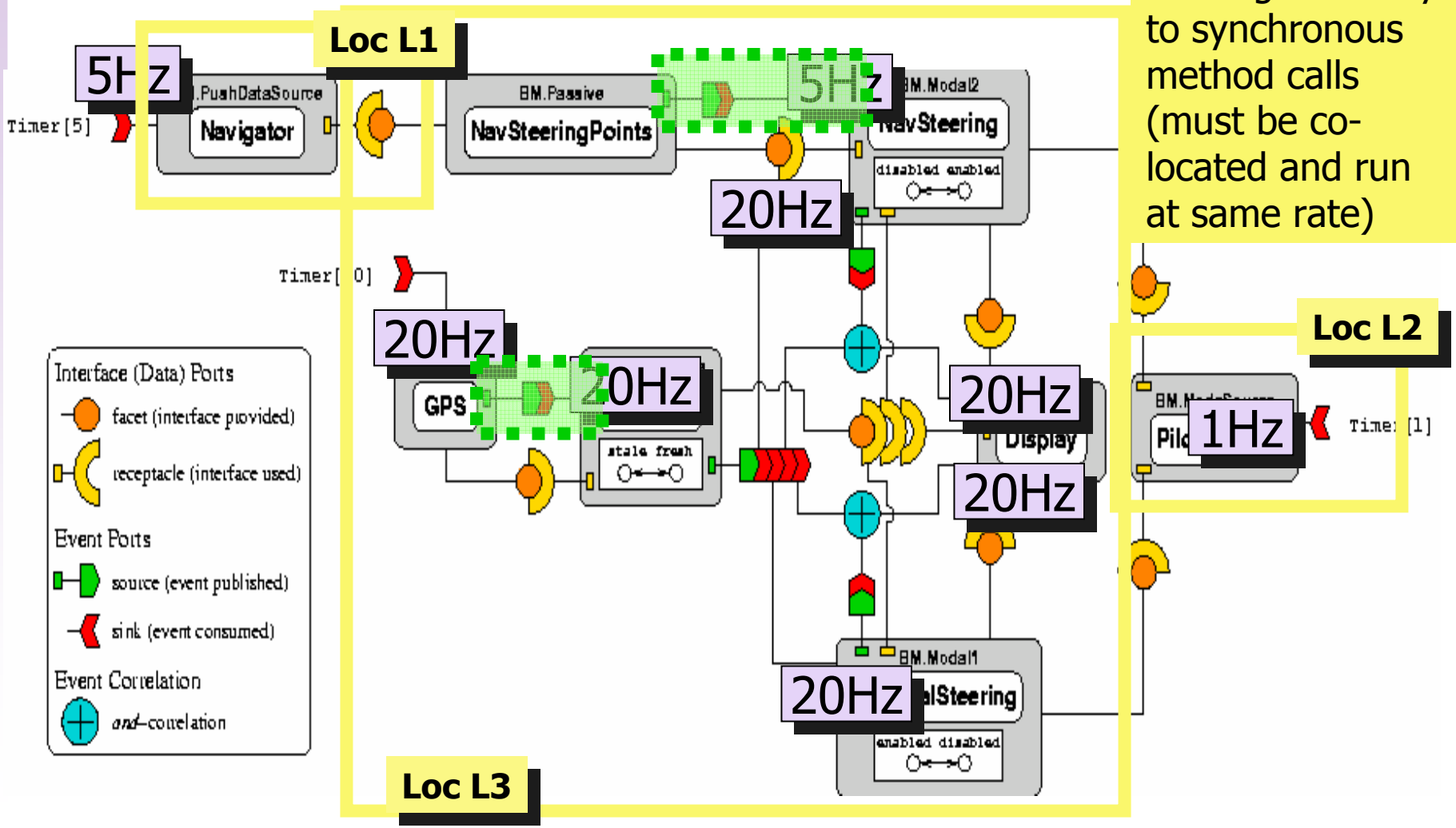
- Many systems may have multiple mechanisms/services or protocols for propagating events
- For example…
  - through full real-time event channel
  - through ORB
  - …optimizations of the above
- Direct dispatch (bypass ORB and event channel) optimization conditions
  - publisher and subscriber
    - run at same rate
    - co-located
  - no correlation involved

developer must select attribute value…



Automated Design Advice

Automatic selection of event transport mechanism



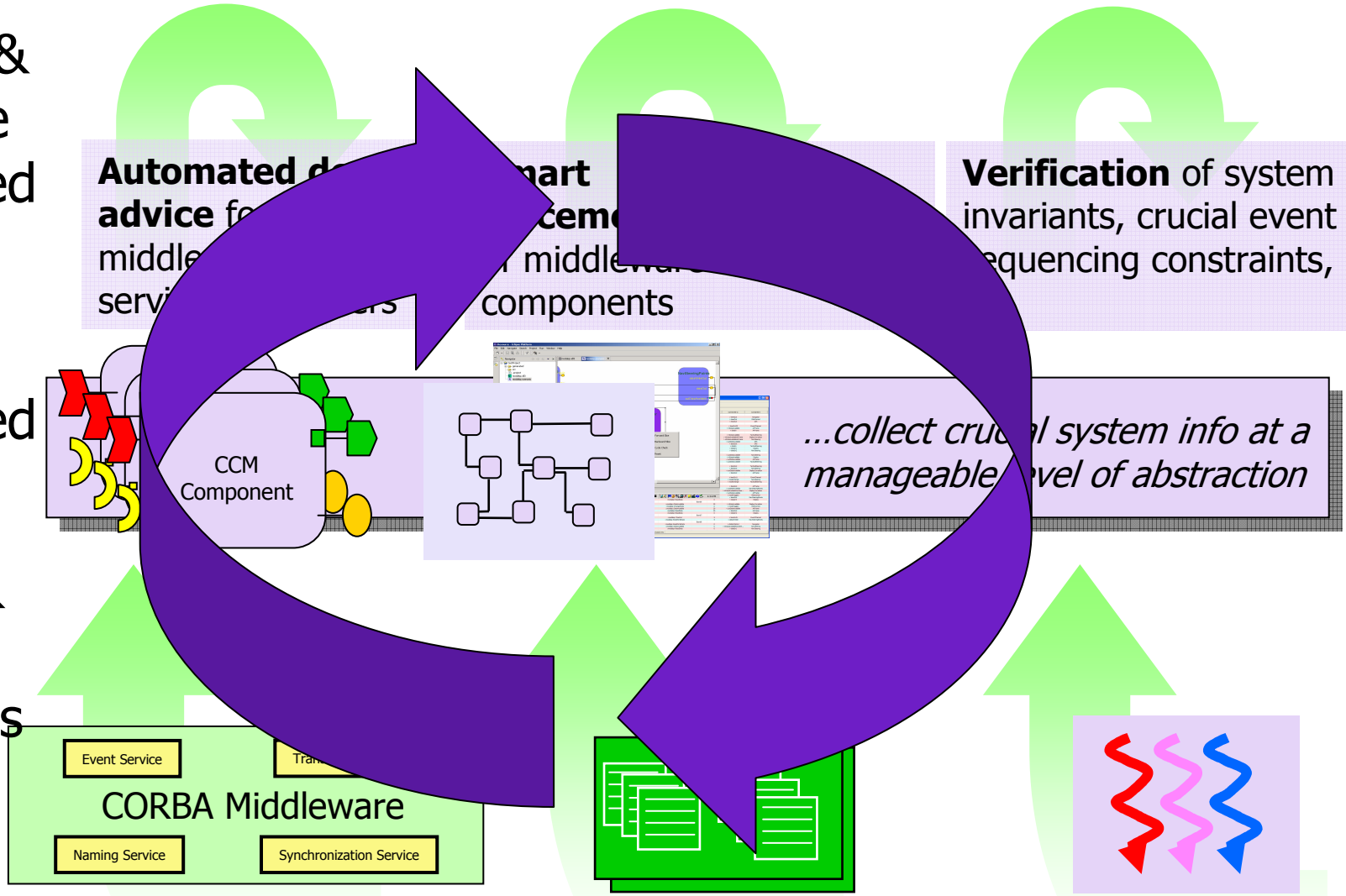
Asynchronous message delivery to synchronous method calls (must be co-located and run at same rate)

Overall Theme

Analyze & Leverage Annotated Models

Annotated Models

Collect & Expose Attributes



Automated d… advice fo… middle… servi… components

Verification of system invariants, crucial event …equencing constraints,

Middleware and service configuration parameters

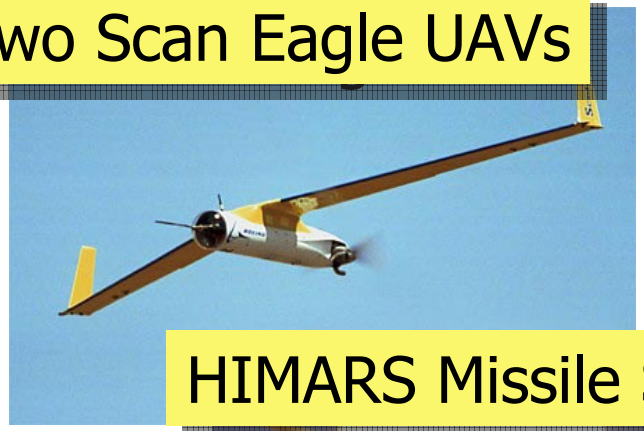
Code level static analysis

Data-collected by run-time monitoring

Boeing Tech Transition

White Sands Missile Range Live Fire Demo (April 2005)

Two Scan Eagle UAVs



HIMARS Missile System

Dynamically Balanced QoS Streaming Video

- Cadena used to develop RT-Java flight control software for Scan Eagles
- Scan Eagles sent streaming video to integrated command/control for launching HIMARS rocket

Concluding Themes

Lifting programming to a higher level of abstraction

- Huge amount of code auto-generated from IDL and model developed structures
- Component designers can be more effectively "compartmentalized" and isolated from complexities of entire system
- *Component integrators* play a key role in correct system construction and need to be supported by a variety of visualizations, structural and semantics analyses

Concluding Themes

"Aspectize" a variety of programming system elements and configure those at the modeling level

- Examples in Cadena…
  - locking strategies, choice of communication layers, configuration/placement of event channel and other service infrastructure, deployment information
- Provide flexible mechanisms to allow component integrators to select from pre-identified values/strategies for these aspects
- Use automated analysis on models of both the system and target platforms (hardware, OS) to automatically synthesize values of these attributes, or at least suggest values as design advice

Concluding Themes

Structural analyses with lightweight semantic information can go a long way

- Many examples in Cadena of model/analysis-driven configuration are based on the simple declaration of intra-component dependences with mode information
- Easy for developers to specify
- Fairly easy to reverse engineer from existing code (using program slicing techniques)
- Easy to check that code conforms to model-based specifications
- Provides a gentle transition path to richer specifications with clear notions of refinement

For More Information…

SAnToS Laboratory,
Kansas State University
http://www.cis.ksu.edu/santos

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

…web-site provides distribution that is integrated with CIAO and OpenCCM, examples, user-manual, tutorial, etc.