AARHUS
UNIVERSITY
DEPARTMENT OF ENGINEERING

Architecture & Design of Embedded Real-Time Systems (TI-AREM)

Architectural Design Patterns 2.

Components & Connectors and Component Architecture Patterns and UML 2.0 Ports

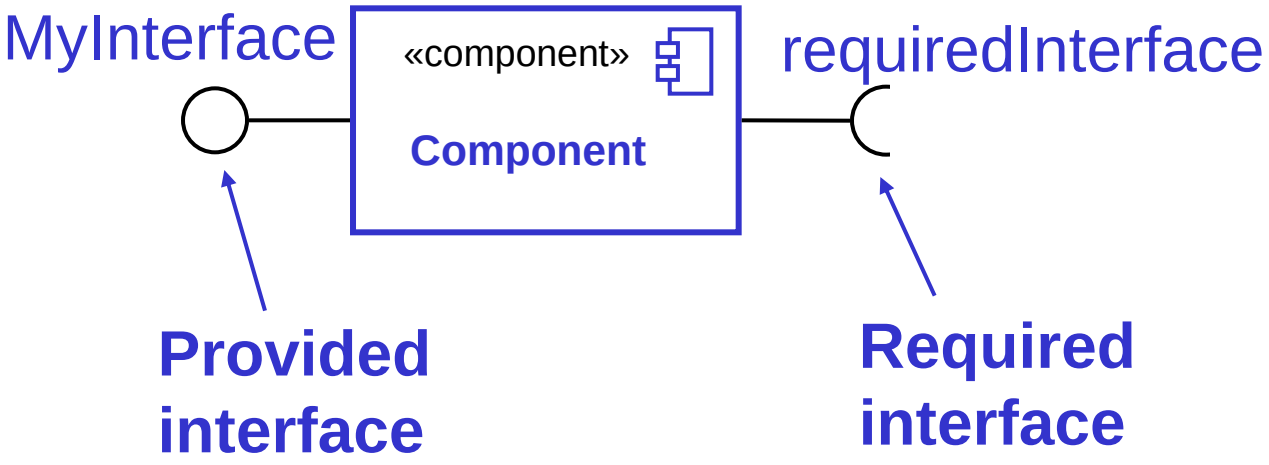
(BPD. Chapter 4.8+4.9. p. 184-201)

Version: 27-2-2015

Agenda

- Component Notation
- Component-based architecture Pattern
- Component & Connectors
- ROOM Pattern
- Design with UML 2.0 Ports

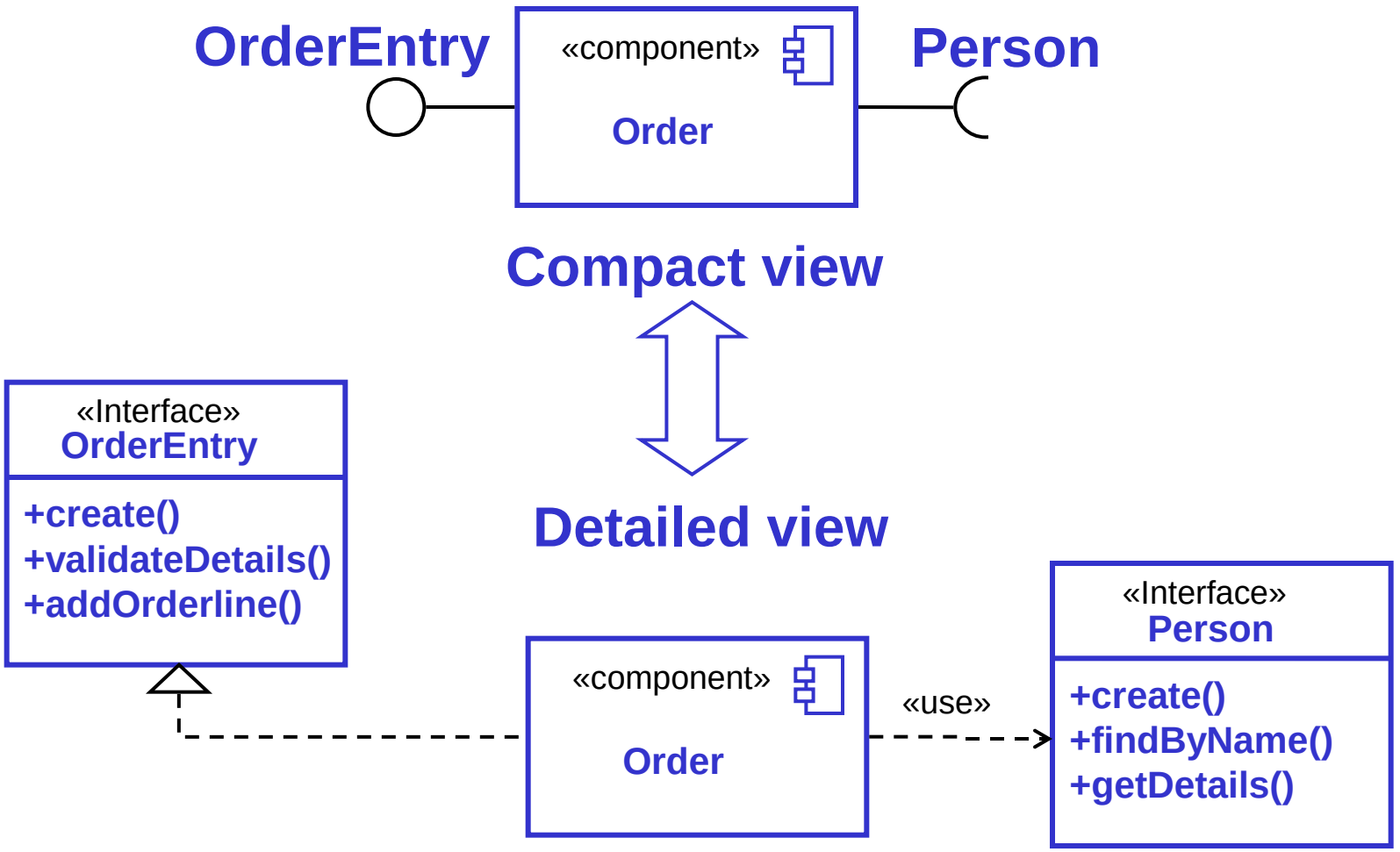
UML Component Notation (1)



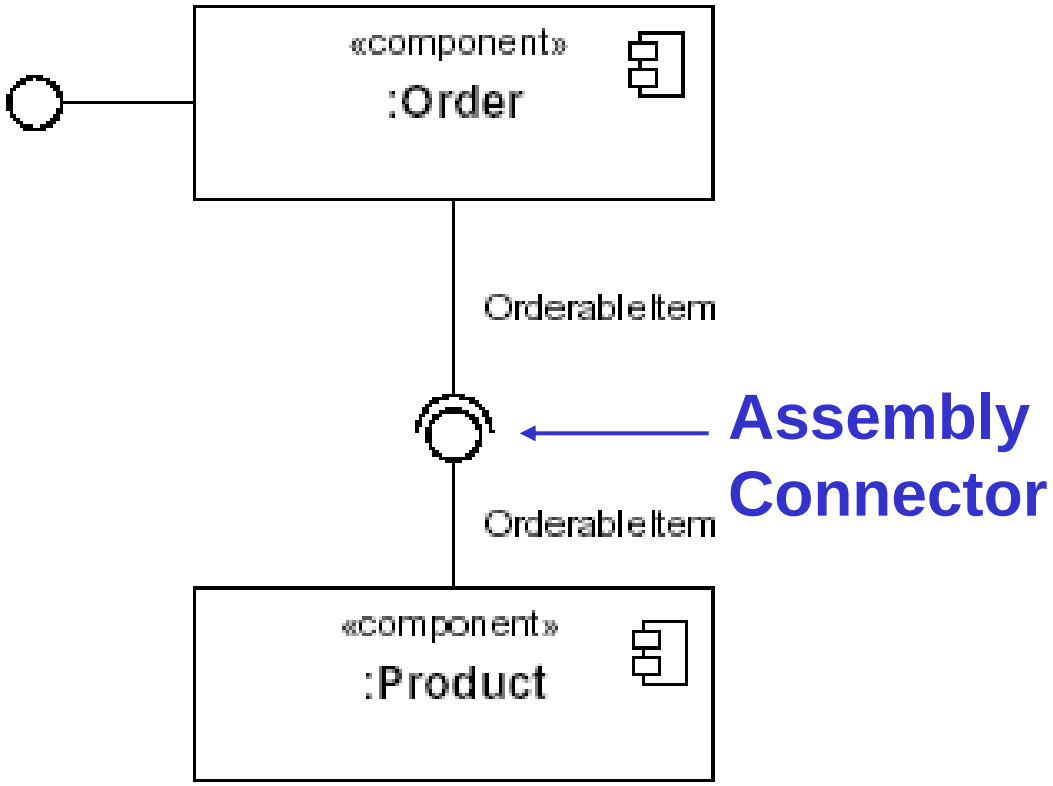
An interface:
defines operation signatures
(name, parameters and return type)

NB! UML 2.0 Component notation and concepts

UML Component Notation (2)



UML Component Notation (3)

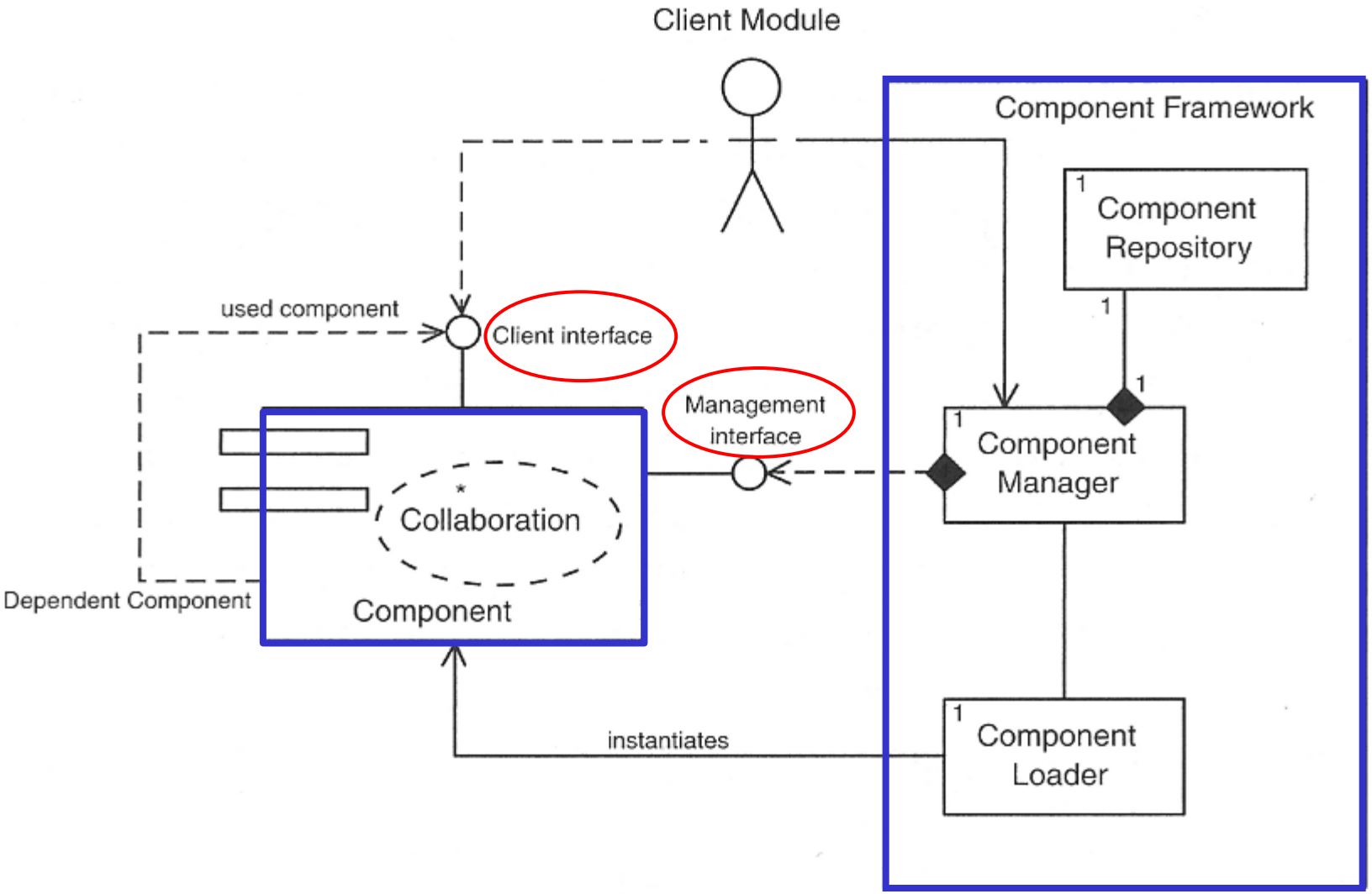


Component-Based Architecture Pattern (BPD 4.8)

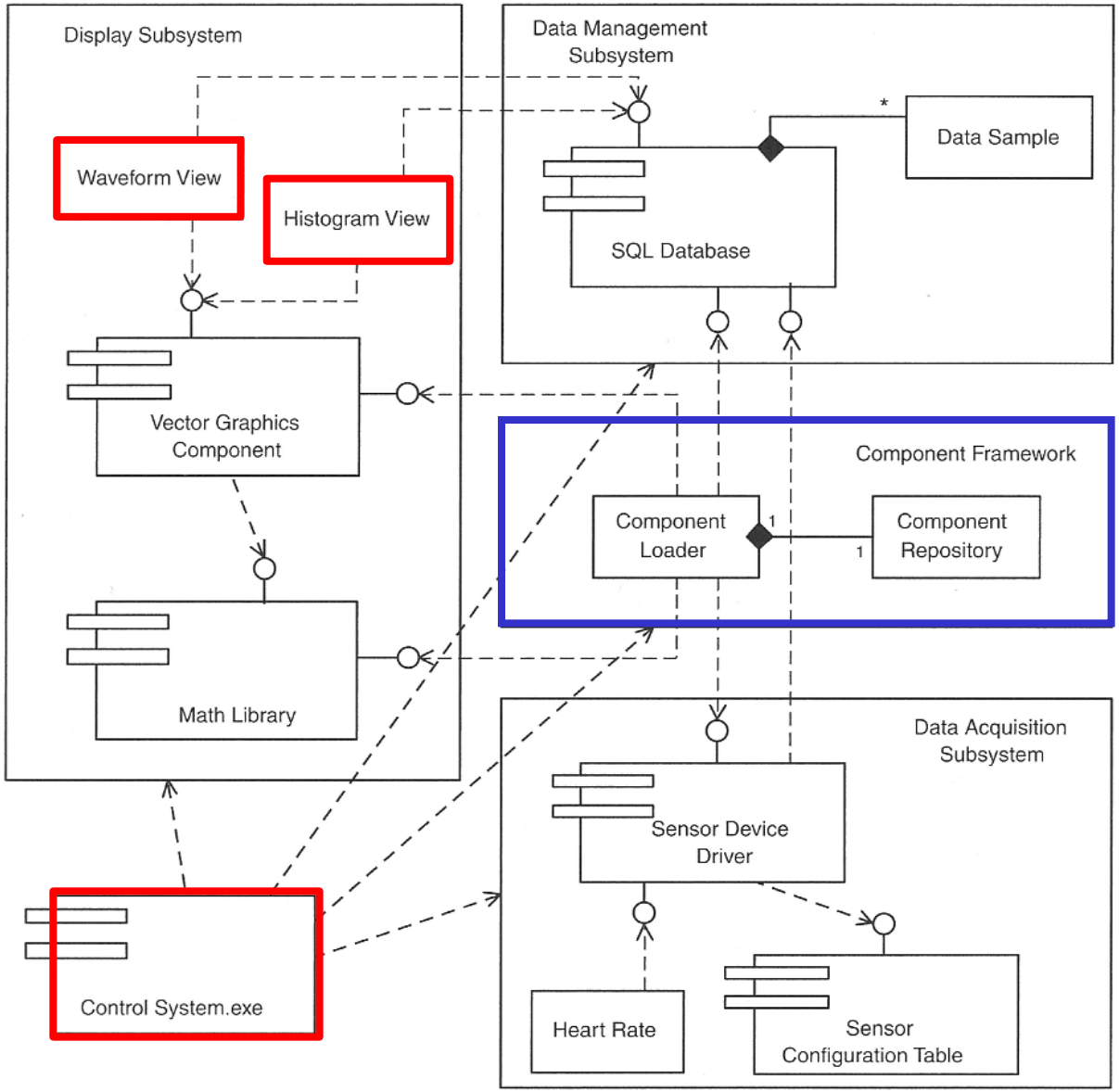
A component in UML is a run-time artifact that forms the basic replaceable unit of software

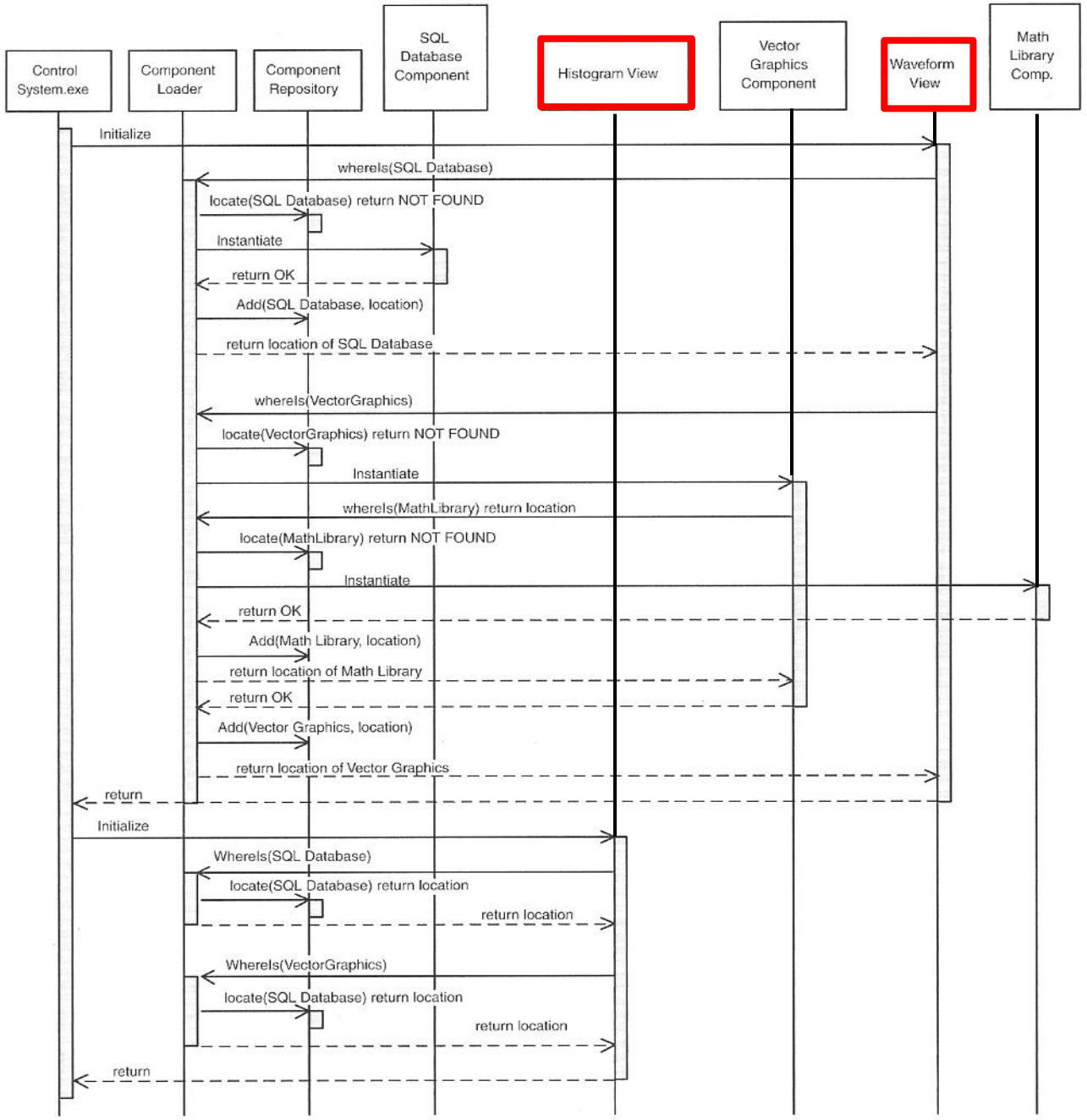
Component examples: static libraries, dynamic link libraries (DLLs), OCX and ActiveX components

Component Based Pattern Structure



Component Based Pattern Example




Control System.exe
Component Loader
Component Repository
SQL Database Component
Histogram View
Vector Graphics Component
Waveform View
Math Library Comp.
Initialize
whereIs(SQL Database)
locate(SQL Database) return NOT FOUND
Instantiate
return OK
Add(SQL Database, location)
return location of SQL Database
whereIs(VectorGraphics)
locate(VectorGraphics) return NOT FOUND
Instantiate
whereIs(MathLibrary) return location
locate(MathLibrary) return NOT FOUND
Instantiate
return OK
Add(Math Library, location)
return location of Math Library
return OK
Add(Vector Graphics, location)
return location of Vector Graphics
return
Initialize
WhereIs(SQL Database)
locate(SQL Database) return location
return location
WhereIs(VectorGraphics)
locate(SQL Database) return location
return location
return

Component & Connectors

- Existing research has focused on
  - Component structure
  - Component Interfaces
  - Component functionality
- More focus needed on component interactions embodied in the notation of **software connectors**

Connectors

- Connector definition:
  - *”Connectors mediates interactions among components, that is, they establish the rules that govern component interaction and specify any auxiliary mechanism required”*
- Connectors should be ***first-class*** modeling constructs

Ref. Article:
“*Towards a Taxonomy of Software Connectors*”

Taxonomy for Software Connectors

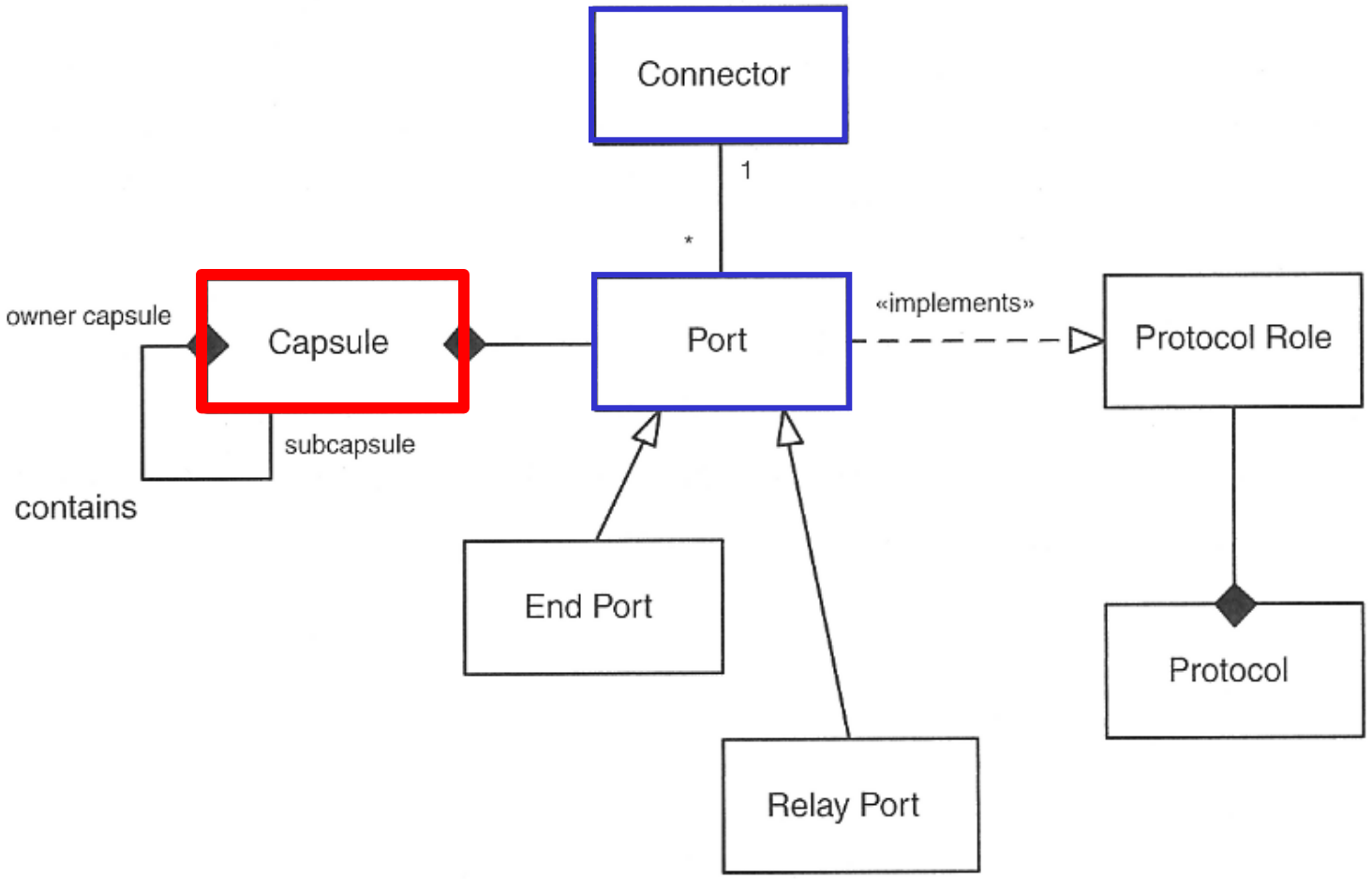
- 4 Service Categories (connector roles)
  - Communication
  - Coordination
  - Conversion
  - Facilitation
- 8 Connector types
  - Procedure call, event, data access, linkage
  - Stream, arbitrator, adaptor, distributor

Ref. Article:
"*Towards a Taxonomy of Software Connectors*"

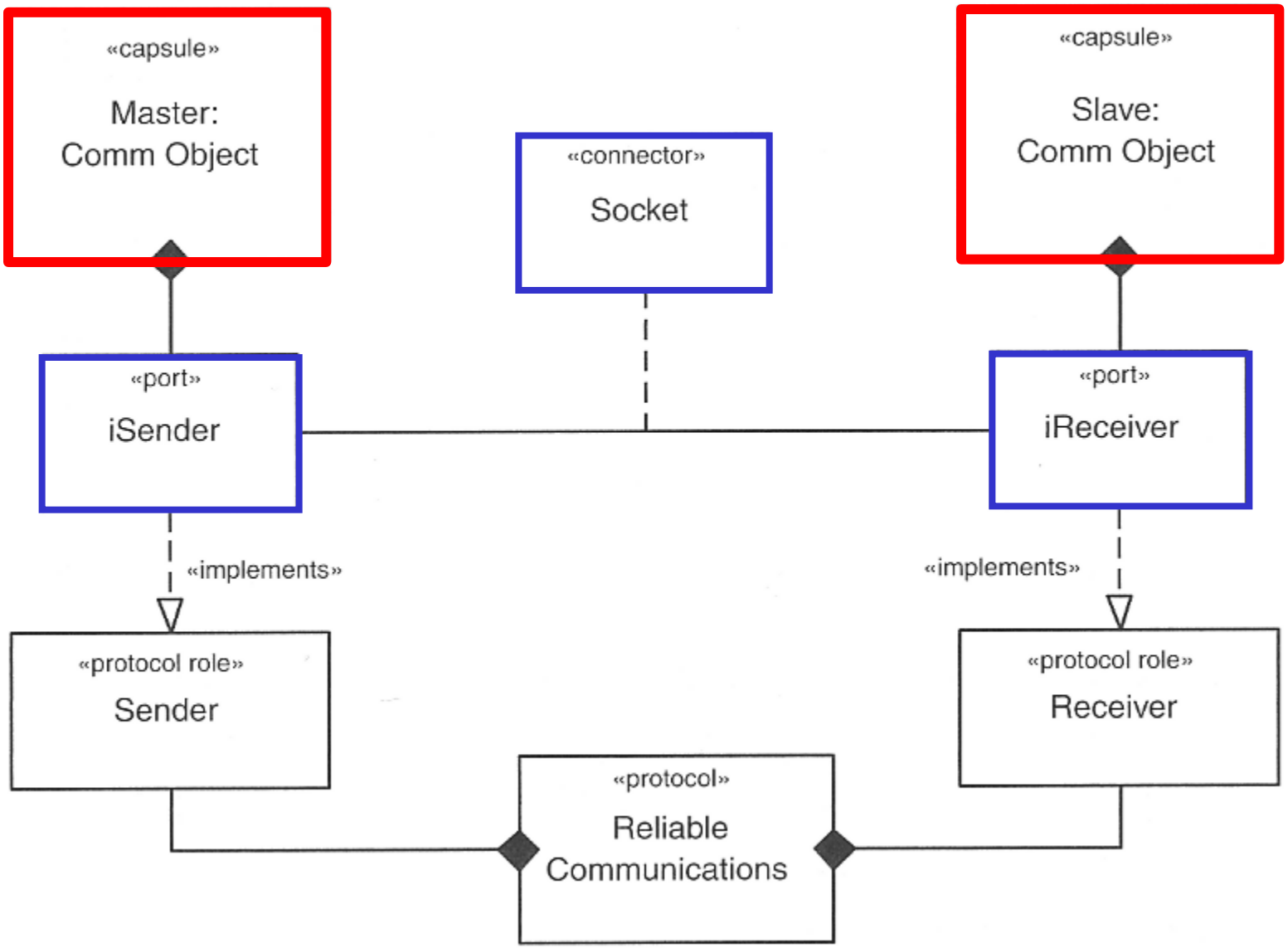
ROOM Pattern (BPD 4.9)

ROOM (Real-Time Object-Oriented Methodology).
The ROOM Pattern is appropriate when the interaction of some large-scale objects is complex and requires special means to mediate and control.

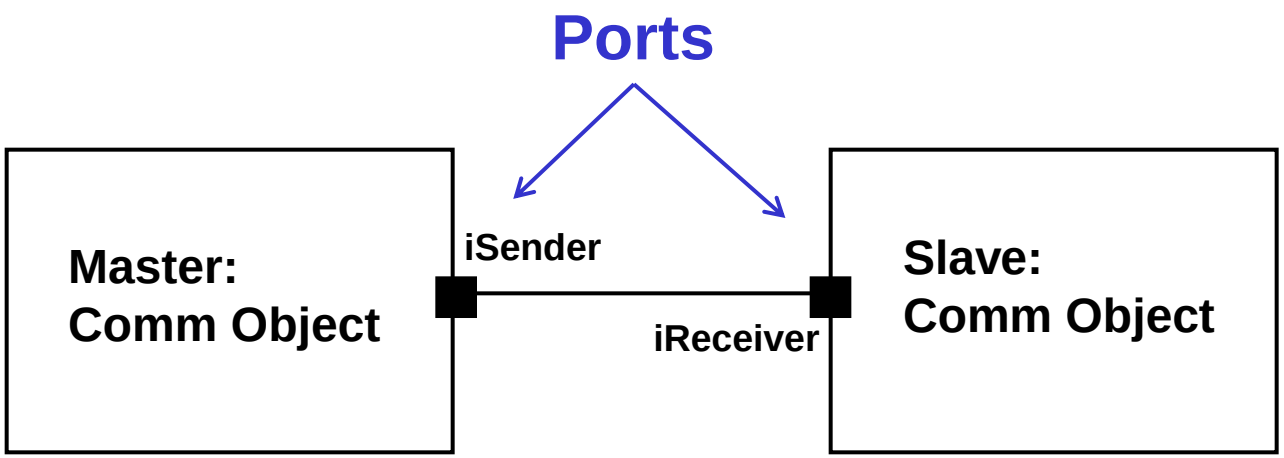
ROOM Pattern Structure



ROOM Pattern Example



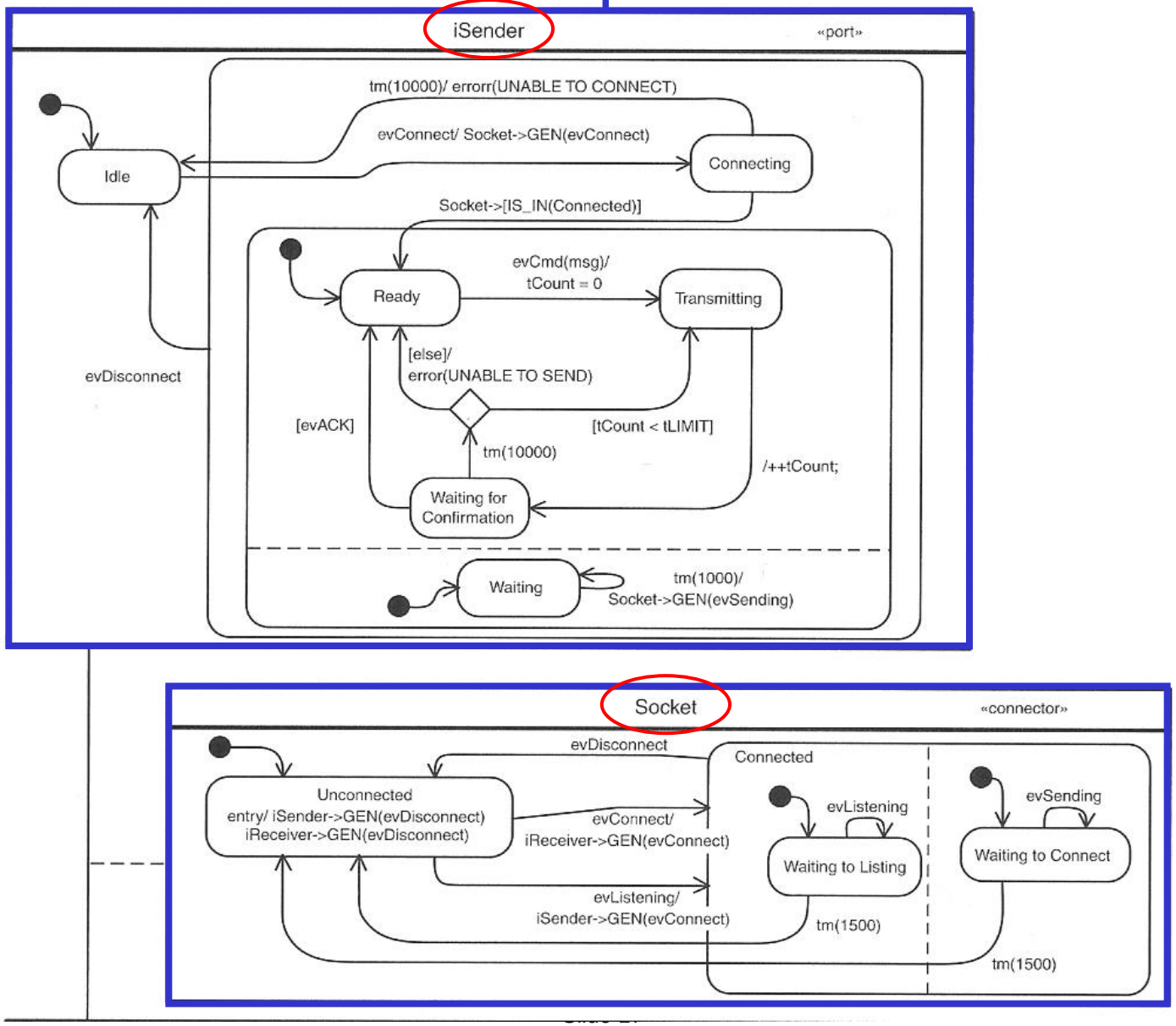
ROOM Ports (~UML Ports)



ROOM Ports has evolved into UML 2.x Ports

End Port= UML behavioral port
Relay Port= UML delegation port

ROOM Pattern Example State Chart Model



UML 2.0 Ports

Ports

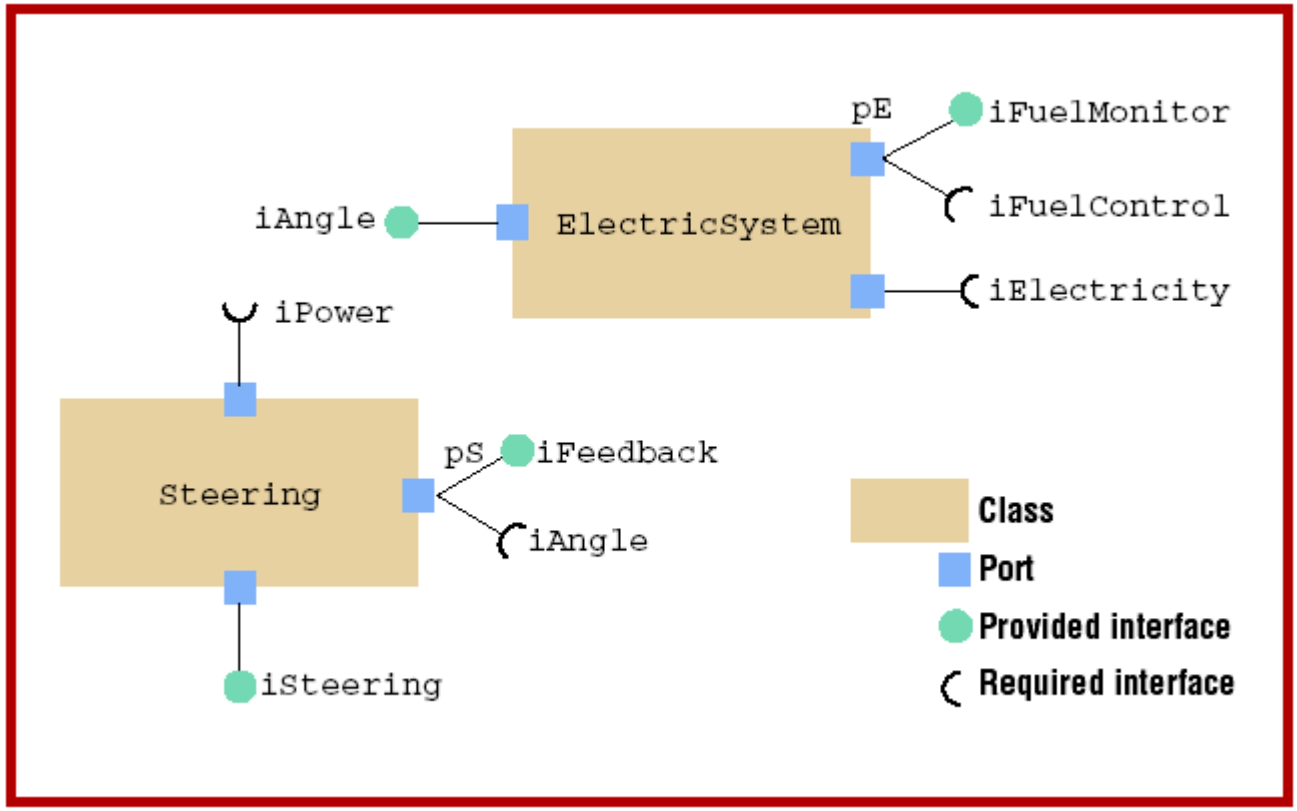
*"The Ports subpackage provides mechanisms for **isolating a classifier from its environment**. This is achieved by providing a point for conducting interactions between the internals of the classifier and its environment. This interaction point is referred to as a **"port."***

Multiple ports can be defined for a classifier, enabling different interactions to be distinguished based on the port through which they occur.

By decoupling the internals of the classifier from its environment, ports allow a classifier to be defined independently of its environment, making that classifier reusable in any environment that conforms to the interaction constraints imposed by its ports".

Ref. UML 2.0 Superstructure spec.

New UML 2.0 Concepts for Classes

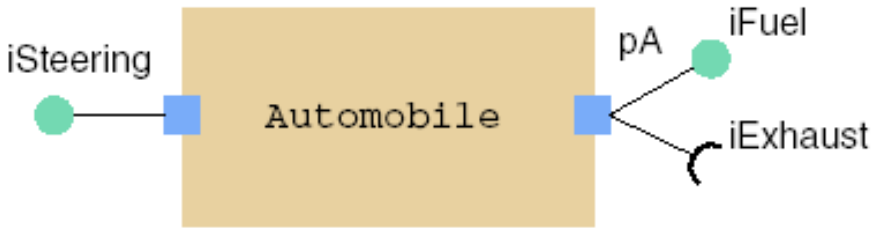


A port groups interfaces belonging to a particular stakeholder. A port can have as well provided as required interfaces.

Ref. Article "Architecting Systems with UML 2.0", IEEE Software jul/aug. 2003.

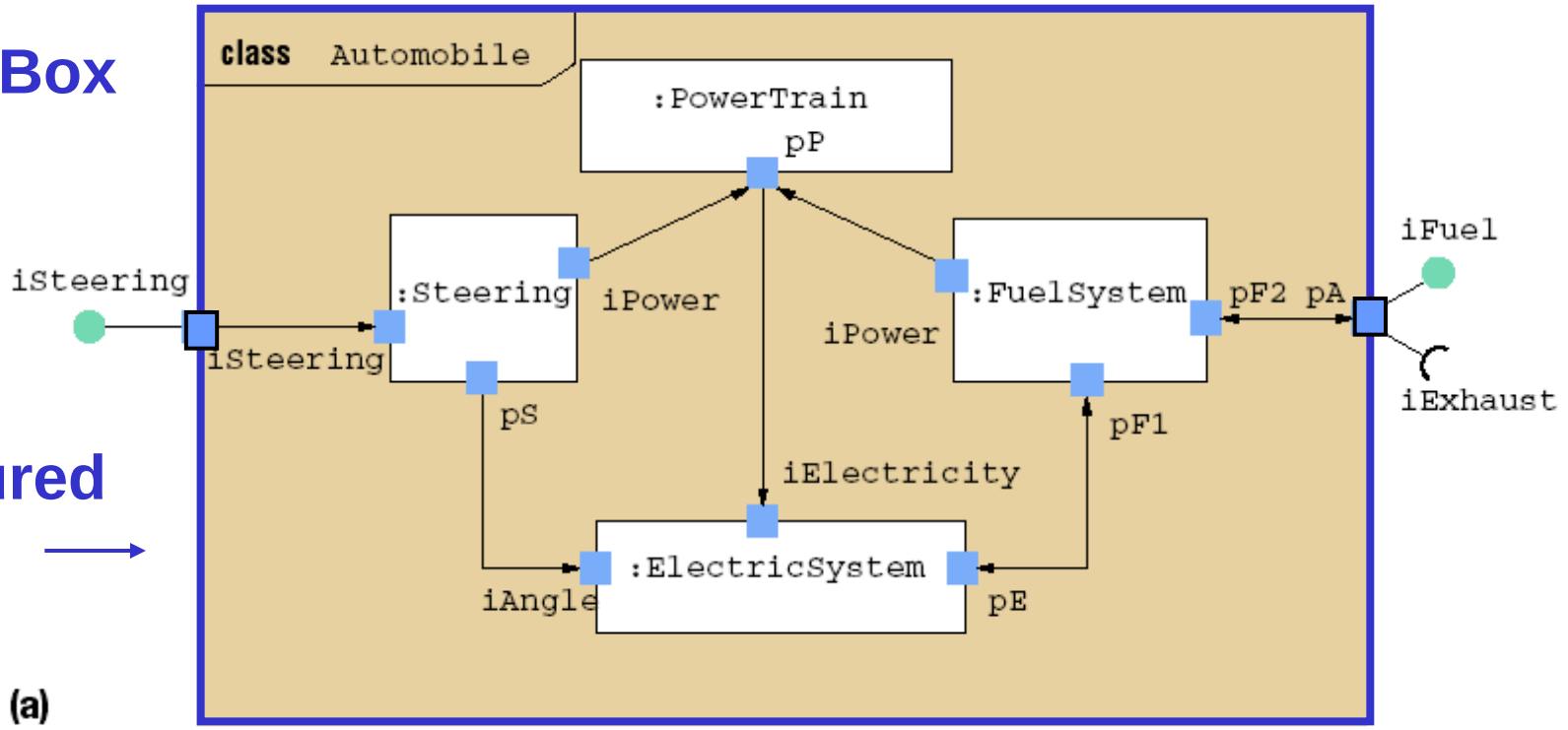
Example with Internal Structure

Black Box View

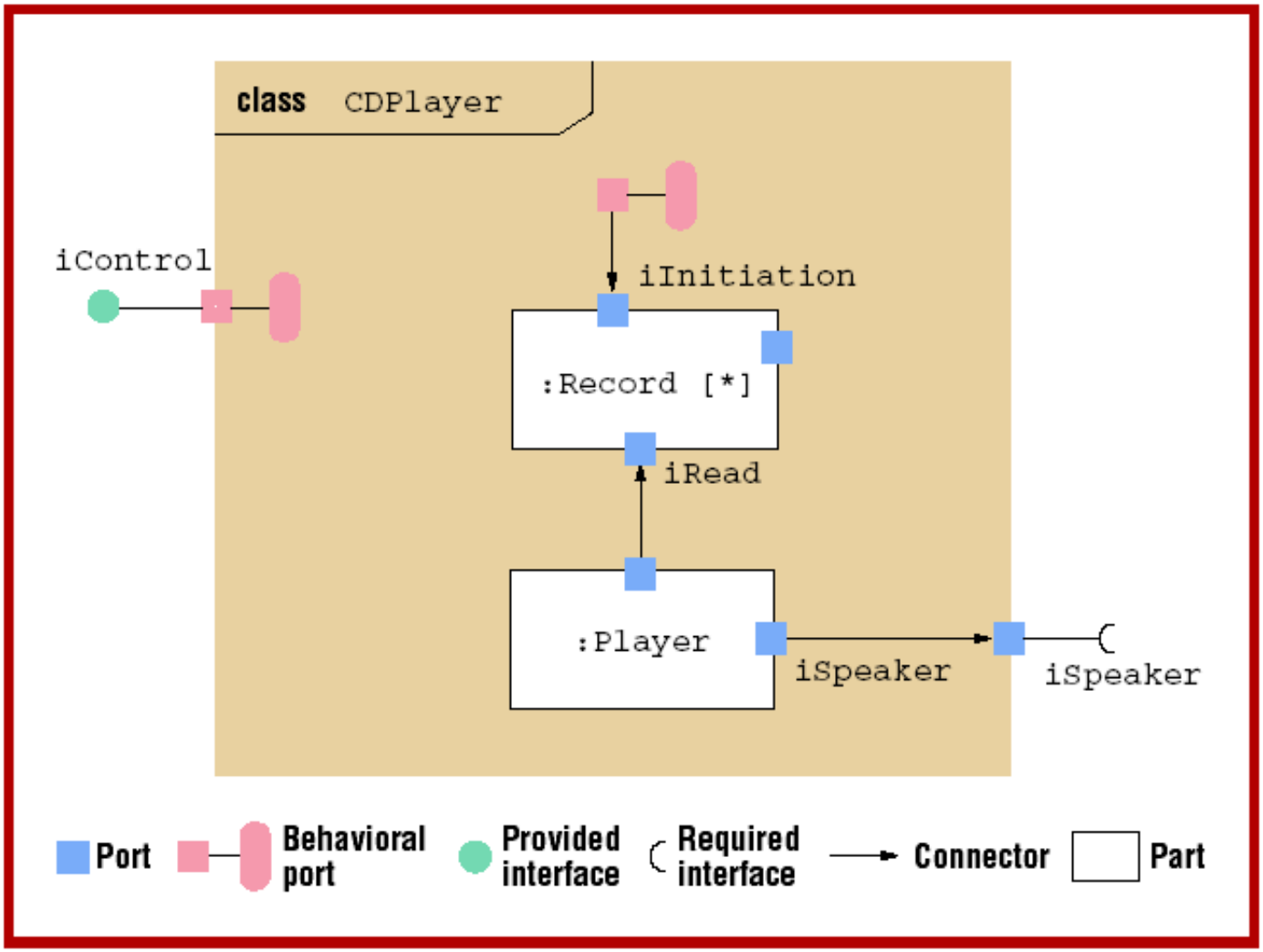


White Box View

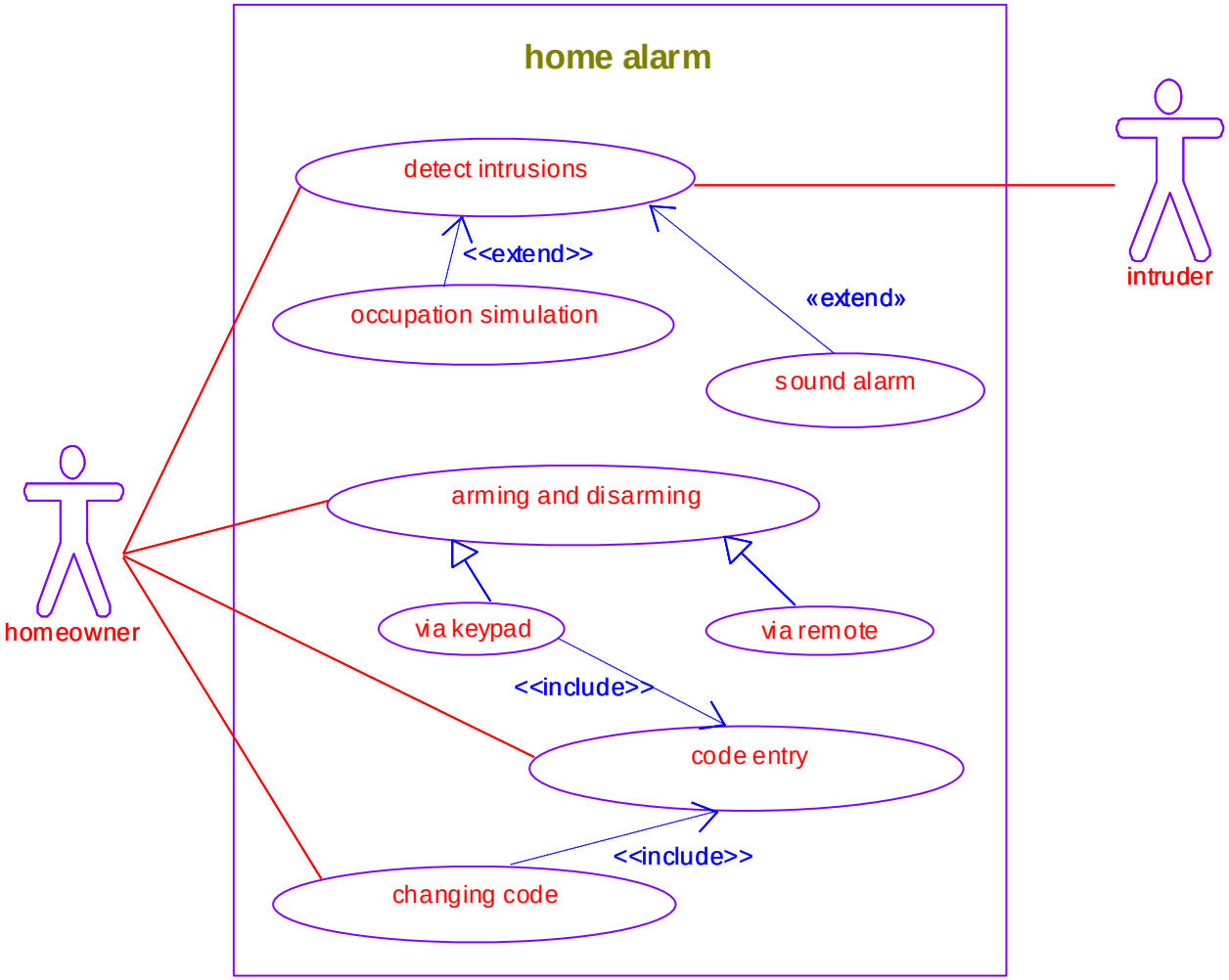
Structured Class ⟶



Example with Behavioral Port

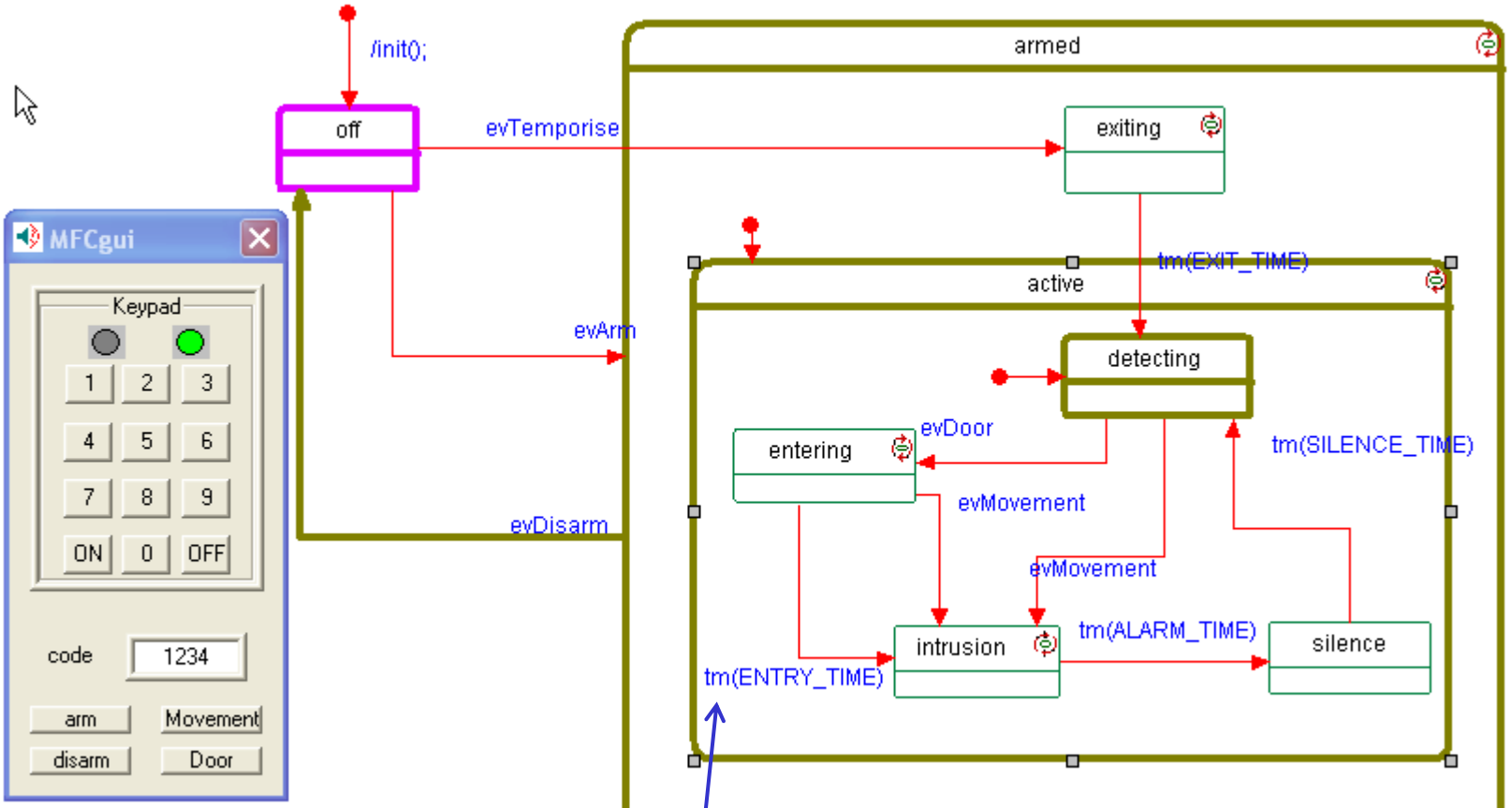


Example: Home Alarm System – version 1



Ref. Rhapsody 8.x

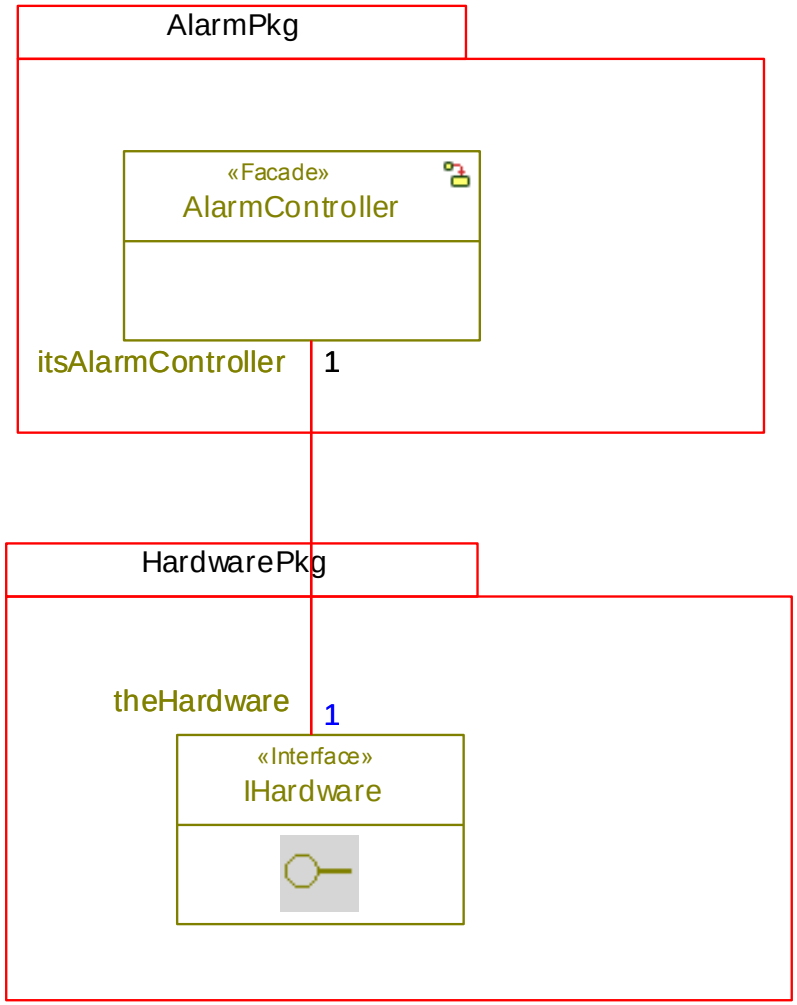
Home Alarm System – version 1 without Ports



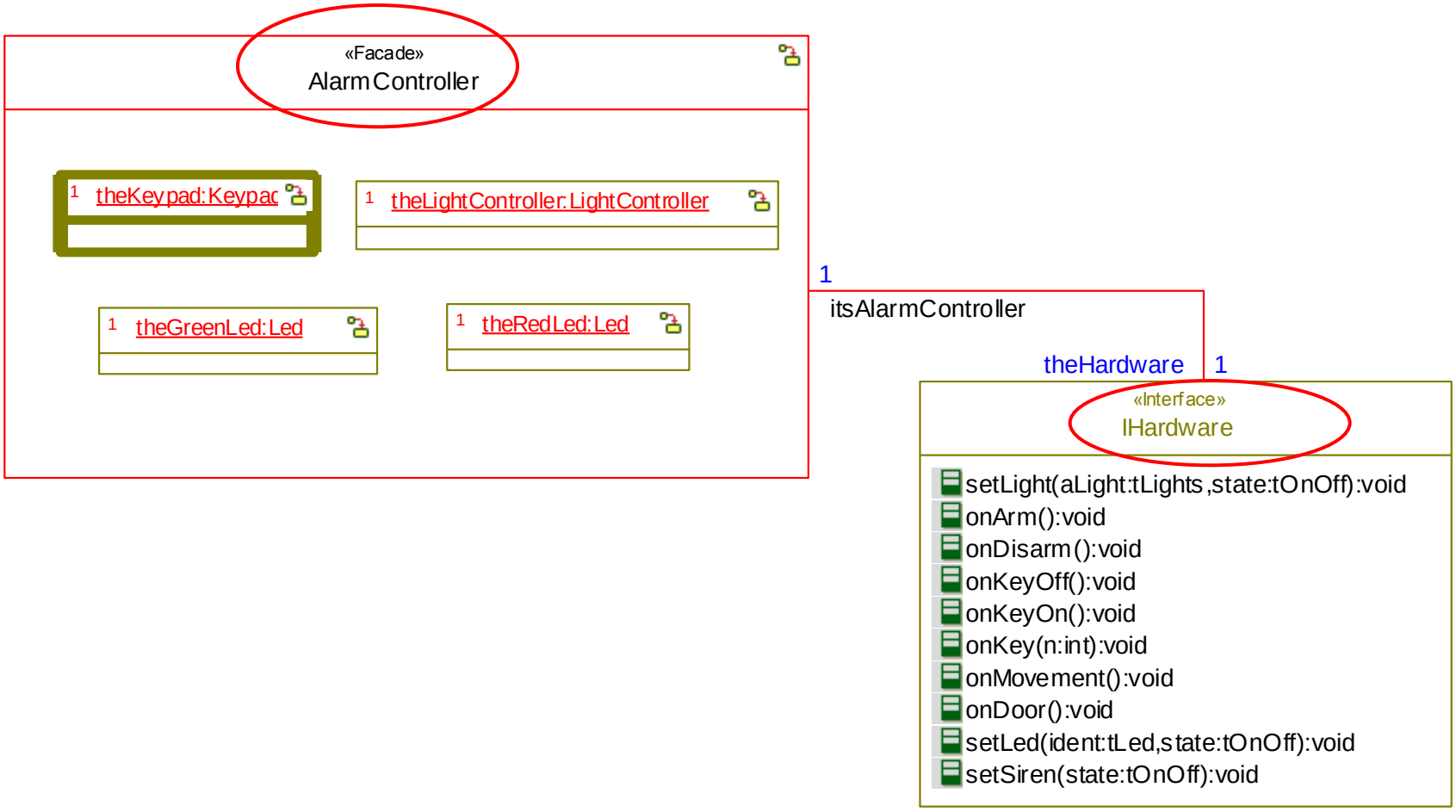
tm= timeout macro

Home Alarm System – version 1

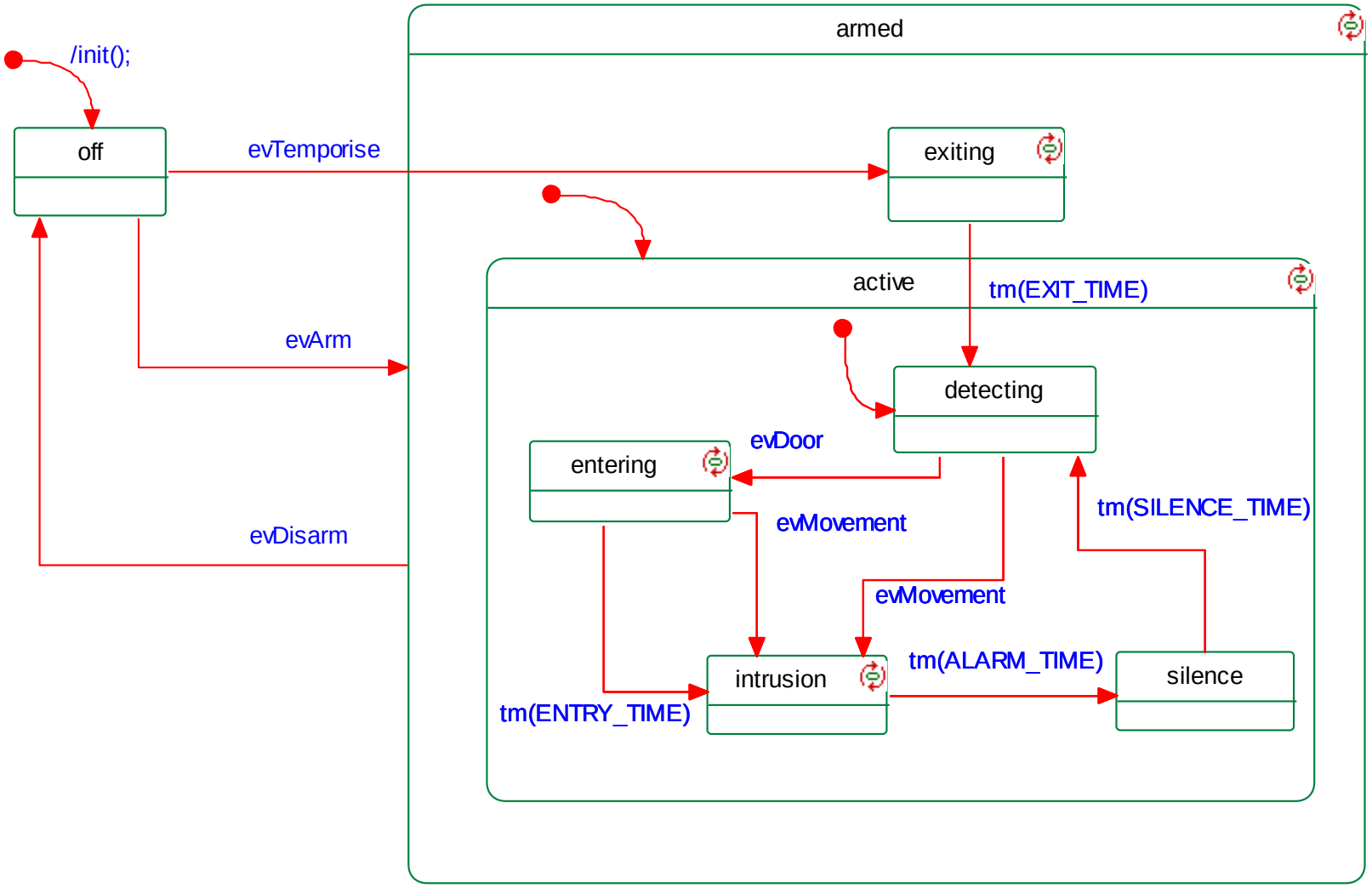
Without Using ports



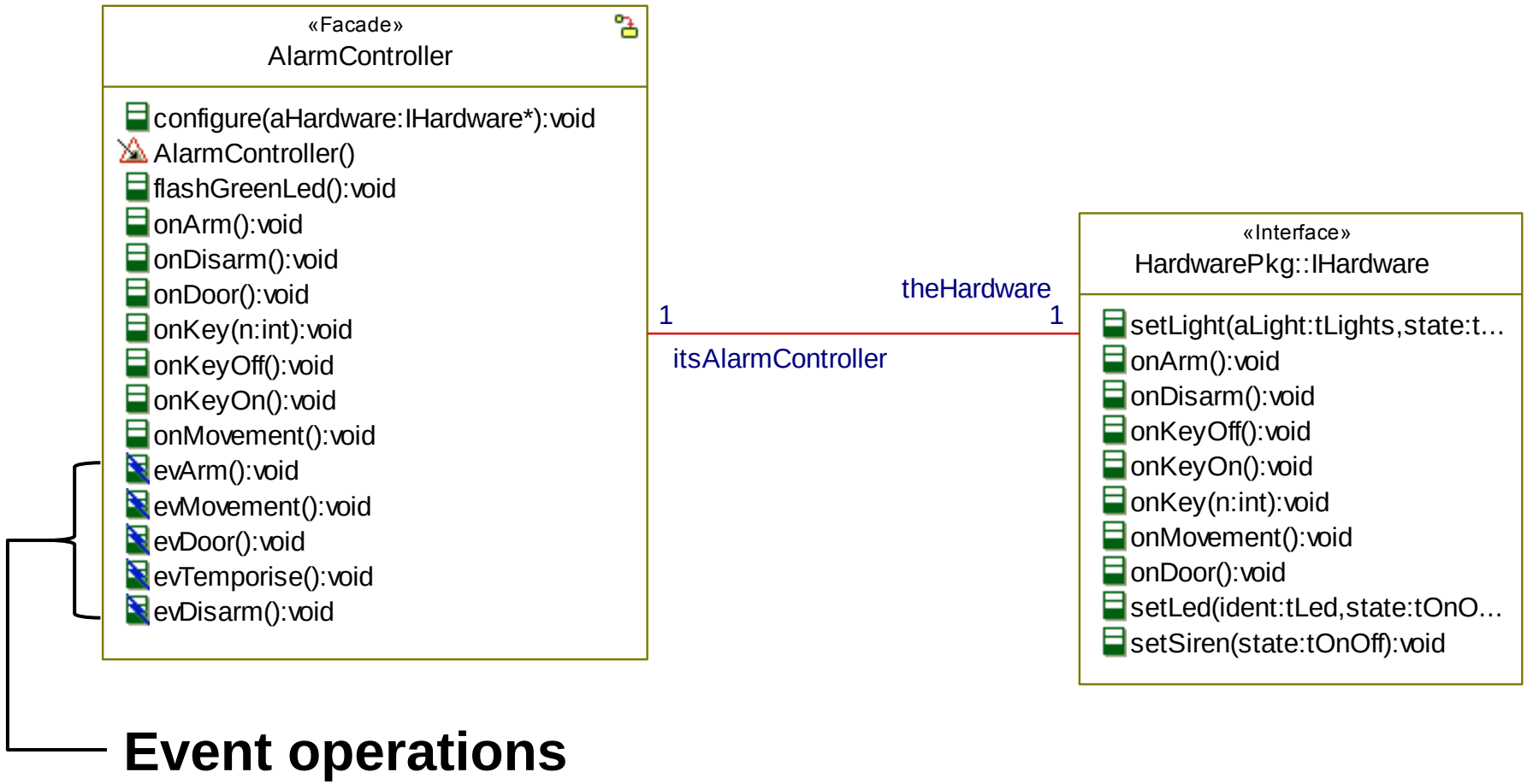
AlarmController – a Composite Class



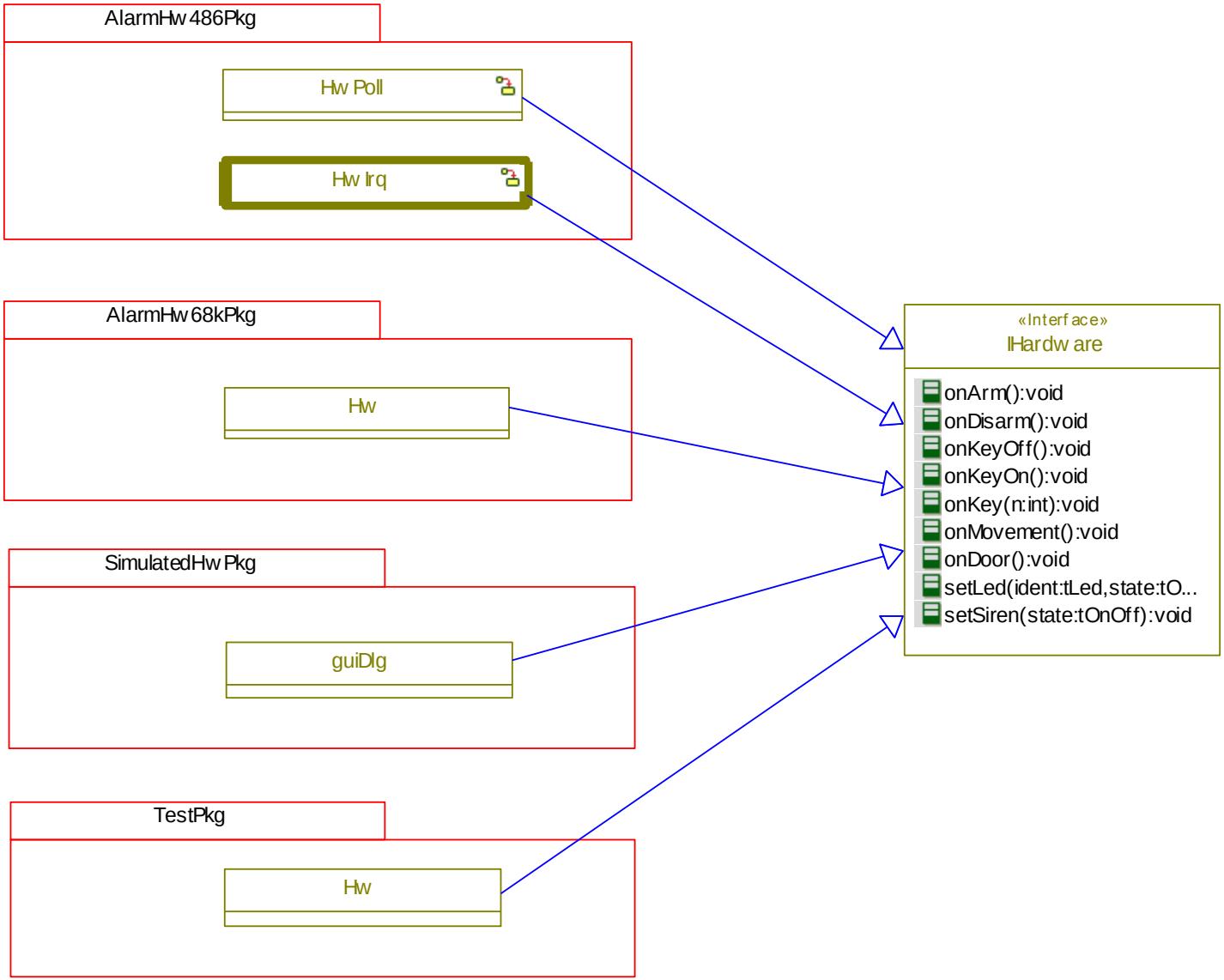
AlarmController State Diagram



AlarmController – Public Operations



Possible Hardware Implementations



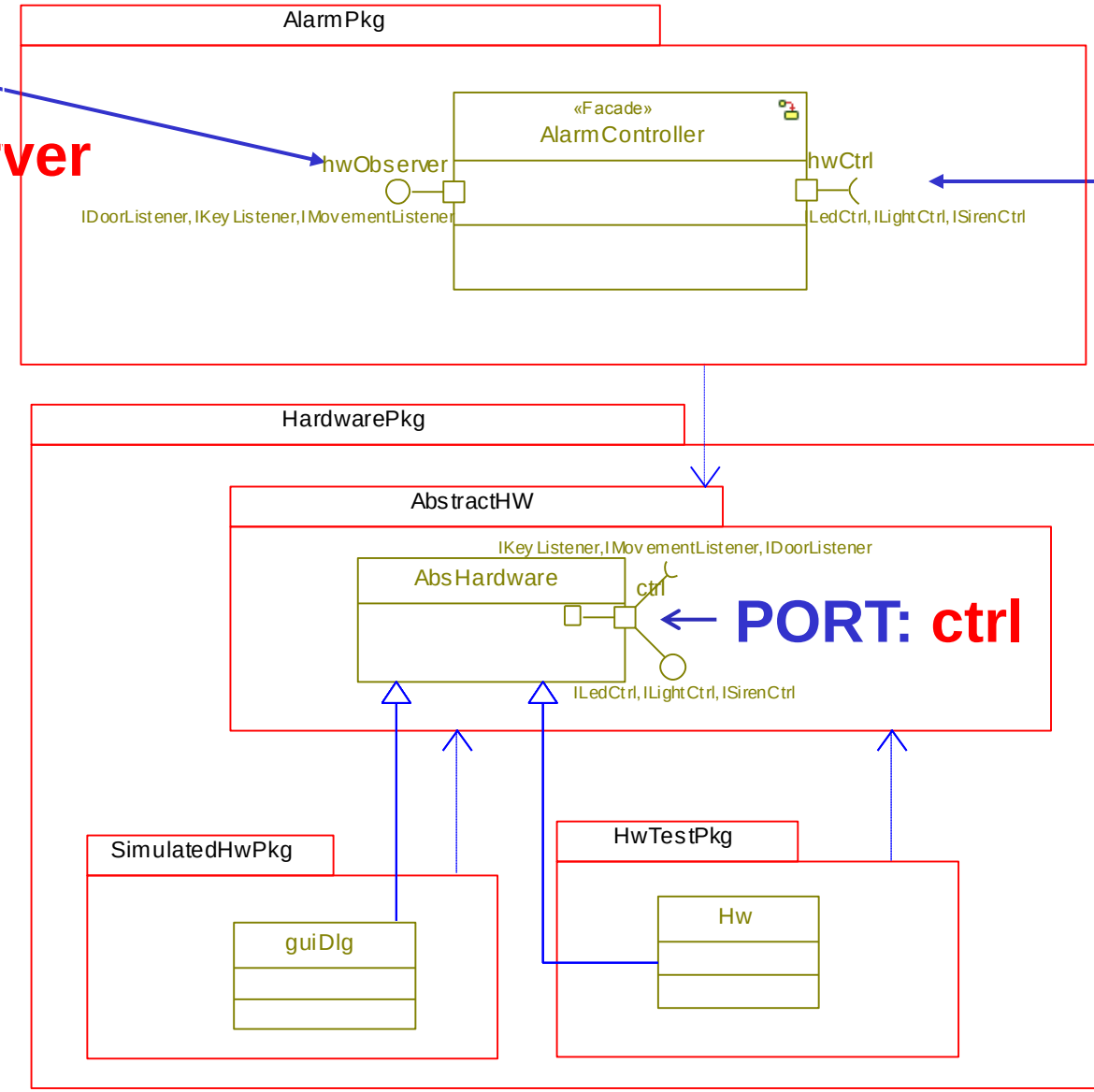
Home Alarm System – version 2 with Ports

- Ports are a new UML2.x modeling concept that allows strong encapsulation of classes from the environment.
- In this case, the **Hardware** and **Alarm Controller** are **decoupled by specifying ports** as their boundary interaction points.

Package Diagram

PORT: hwObserver

With 3 provided interface

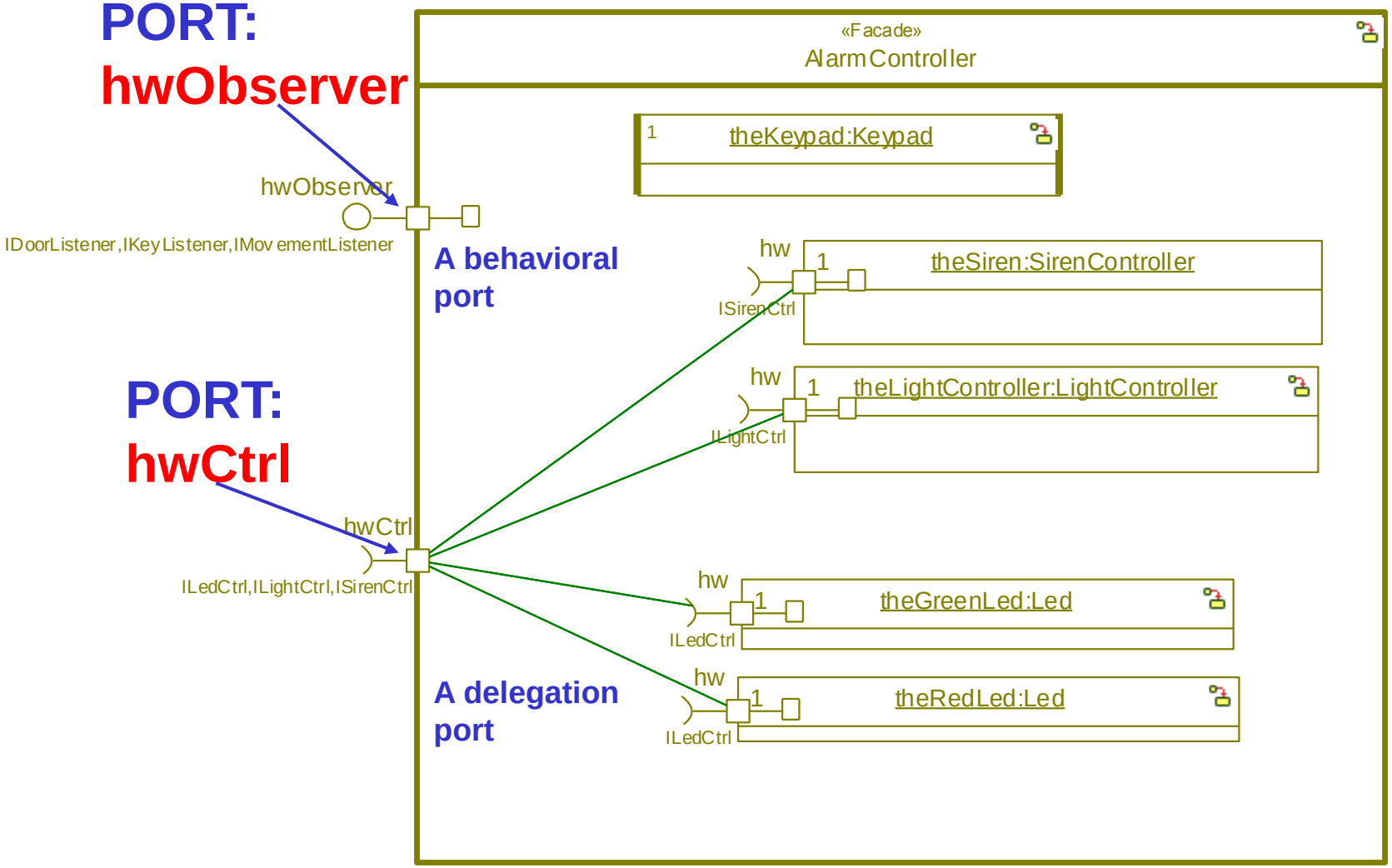


PORT: hwCtrl

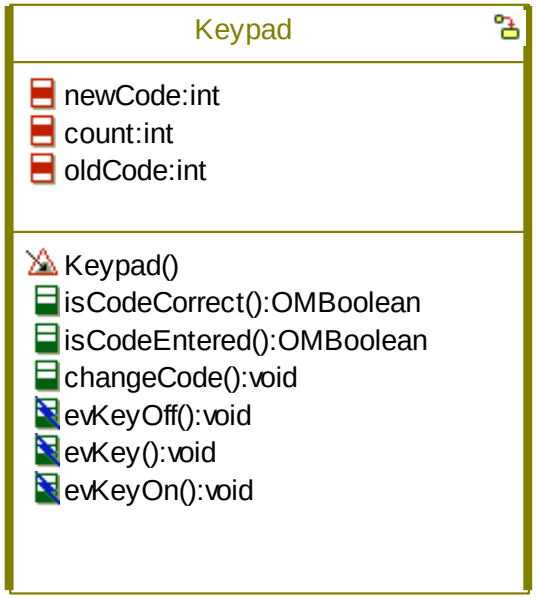
With 3 required interface

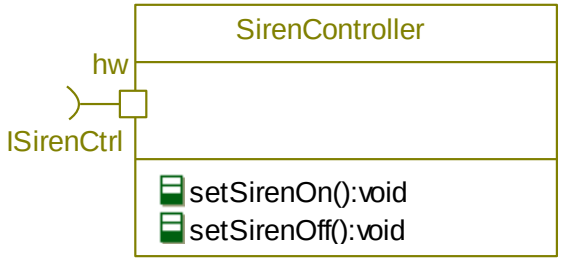
PORT: ctrl

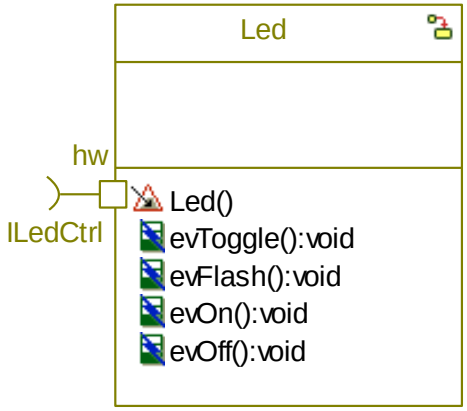
Home Alarm System – version 2 with Ports



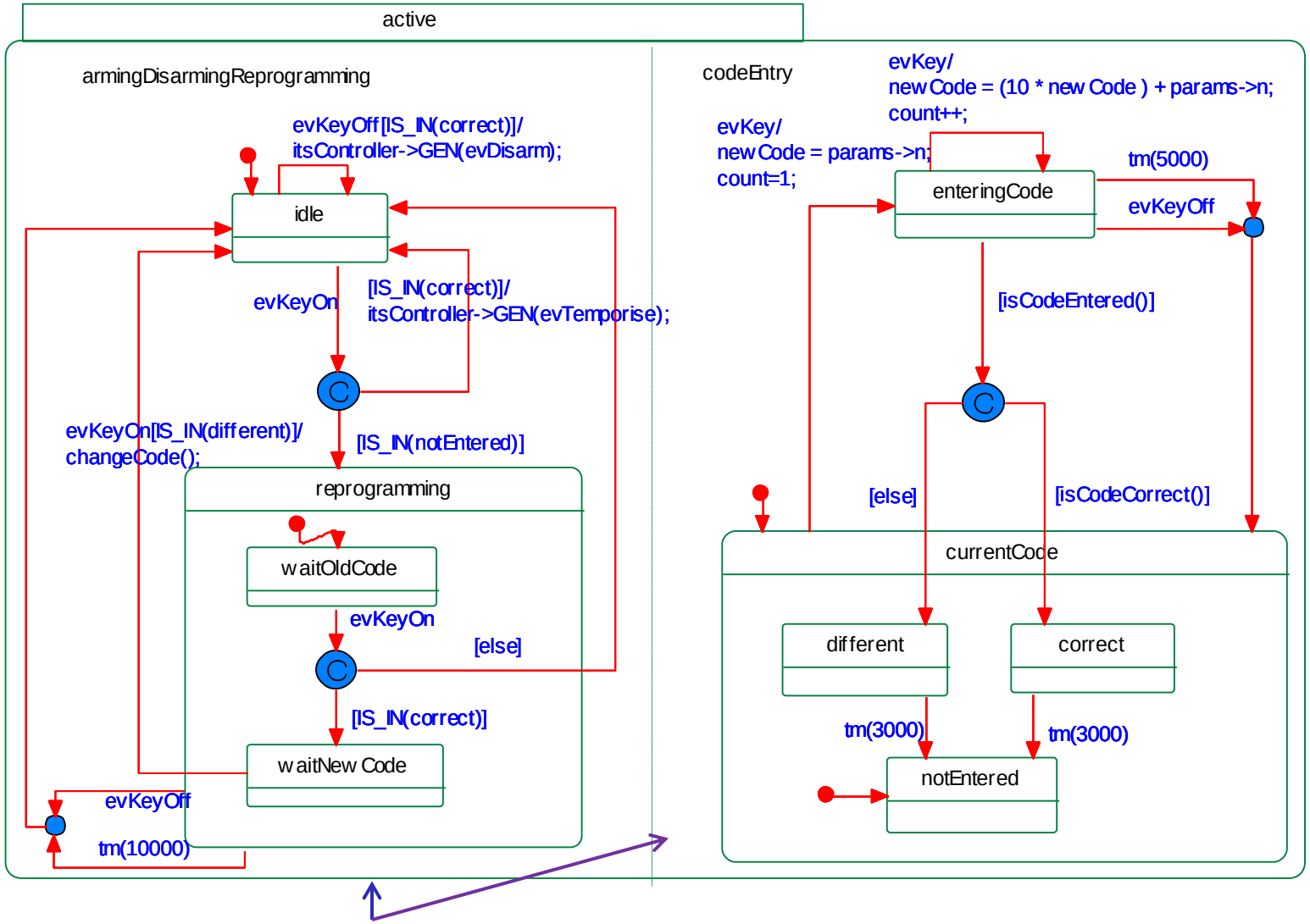
Classes: KeyPad, SirenController, Led





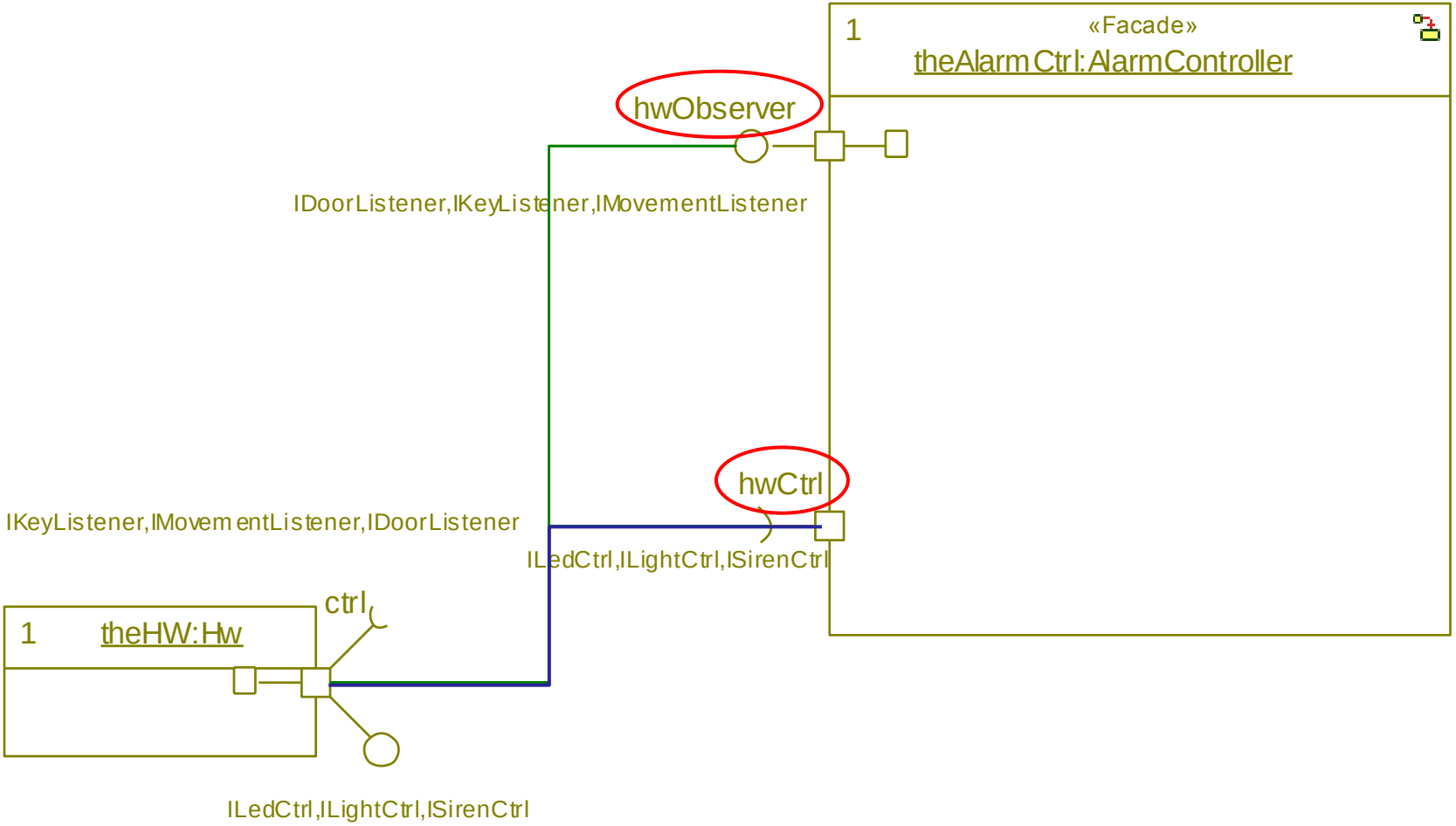


KeyPad – State Diagram

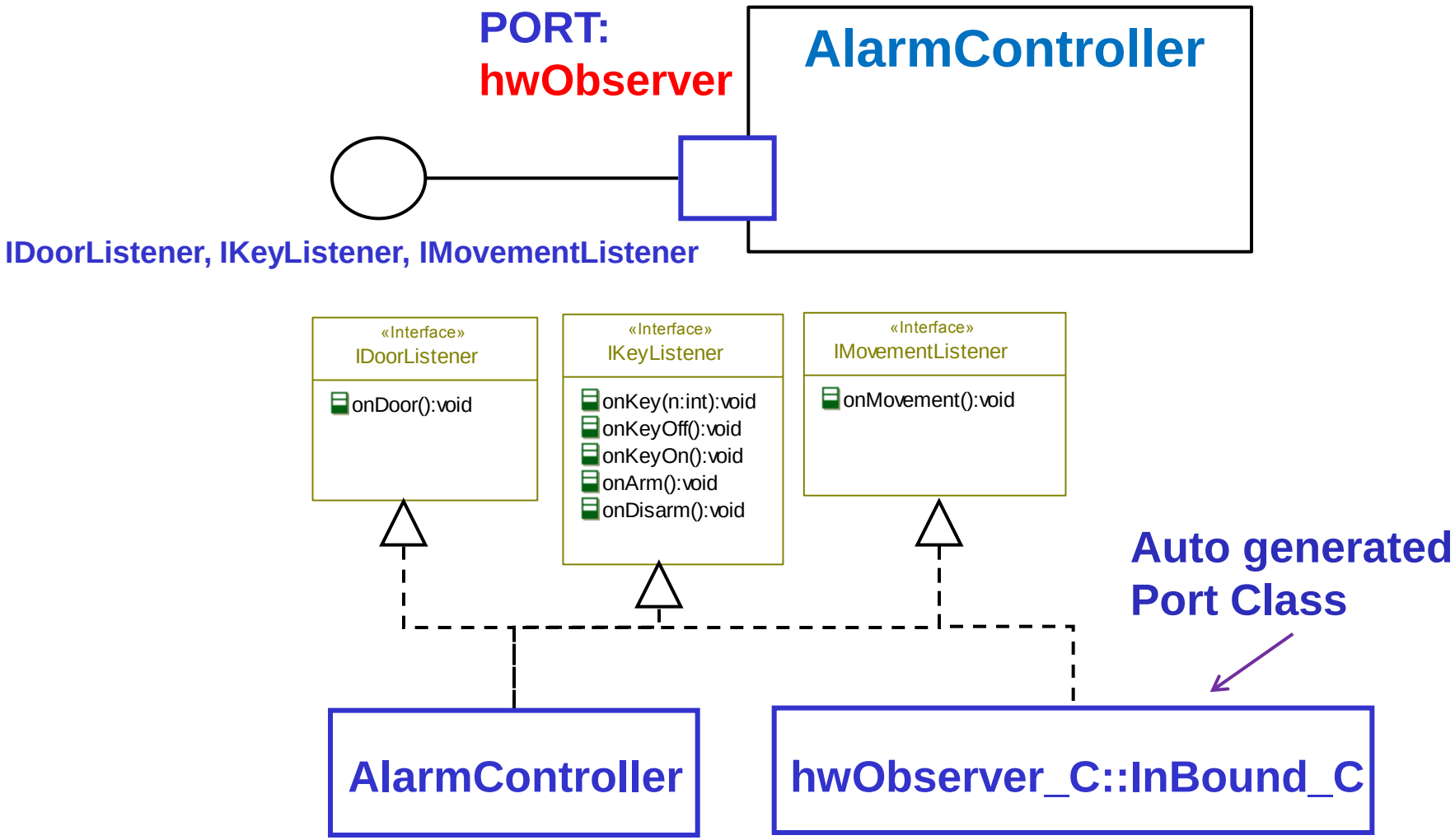


2 AND states

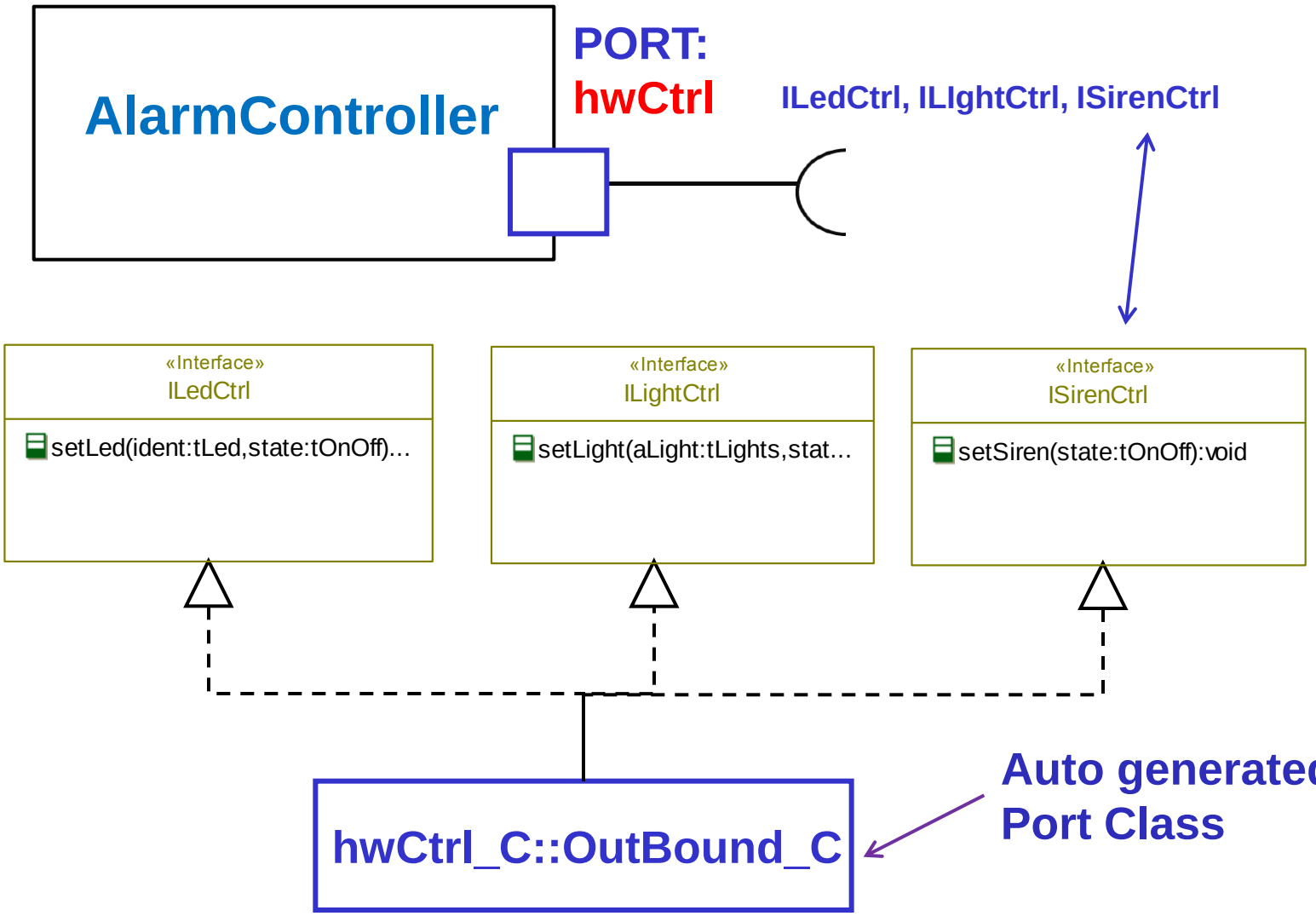
Test Configuration



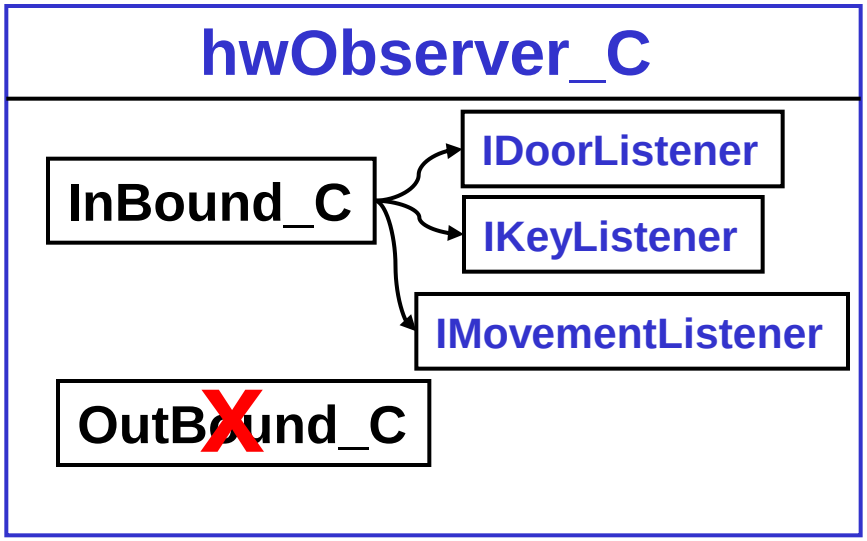
Rhapsody Implementation of a Port (1)

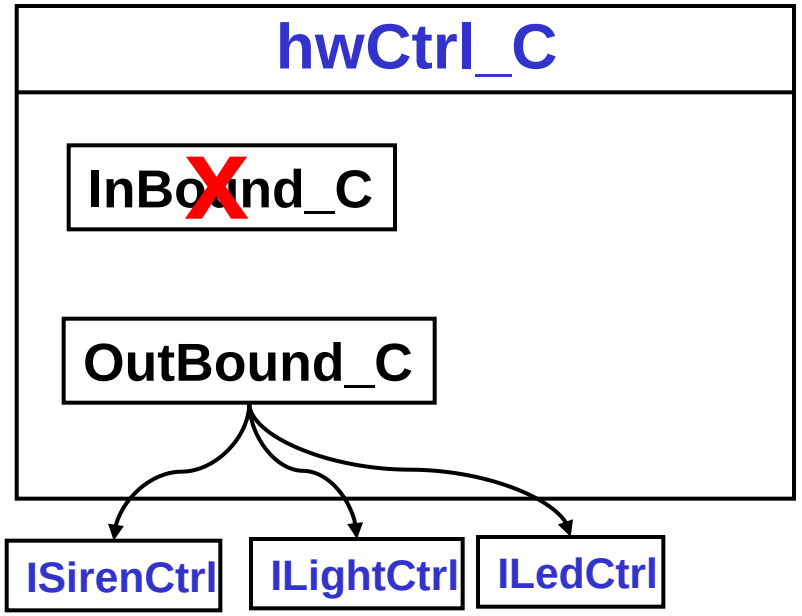


Rhapsody Implementation of a Port (2)

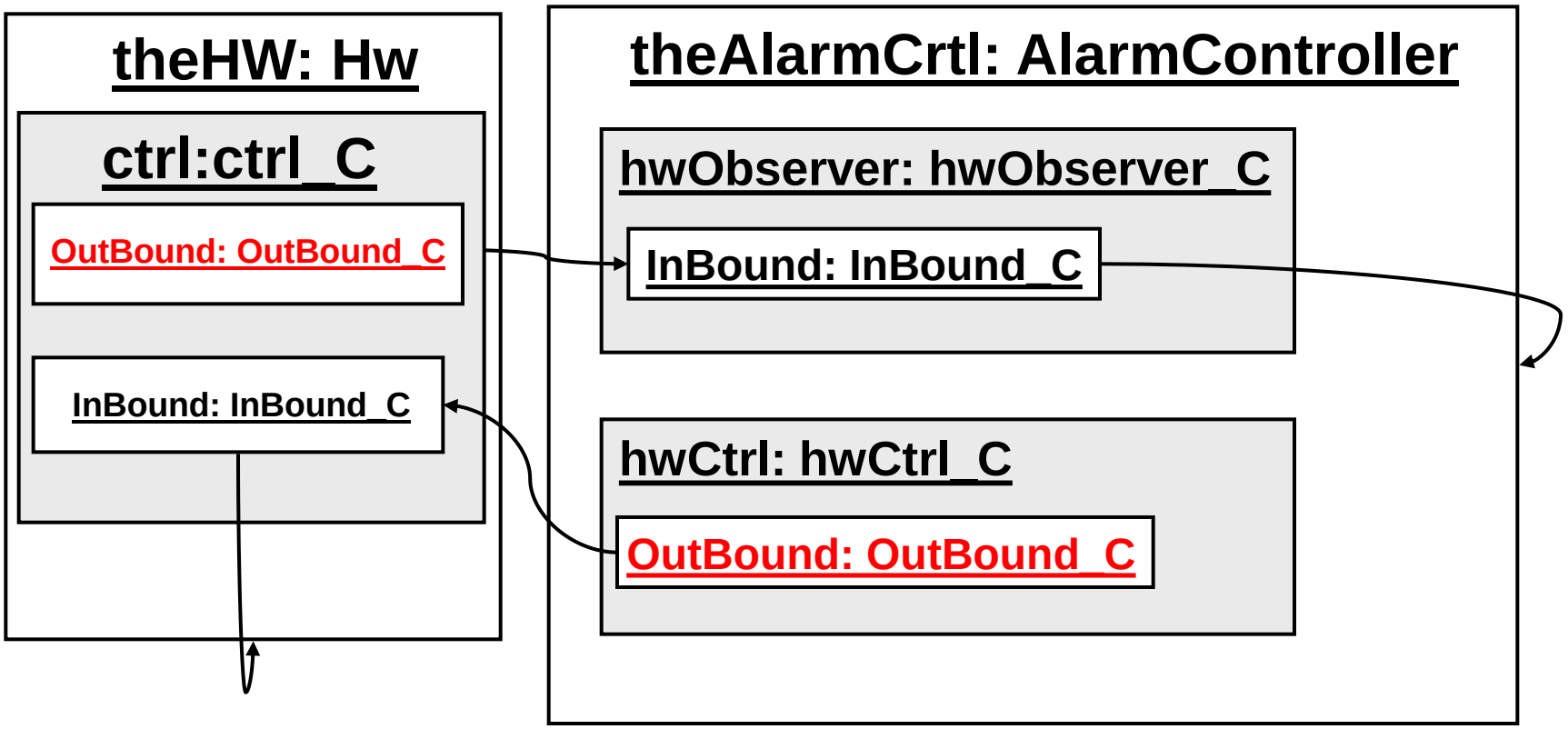


Rhapsody Implementation of a Port (3)





Rhapsody Implementation of a Port (4)



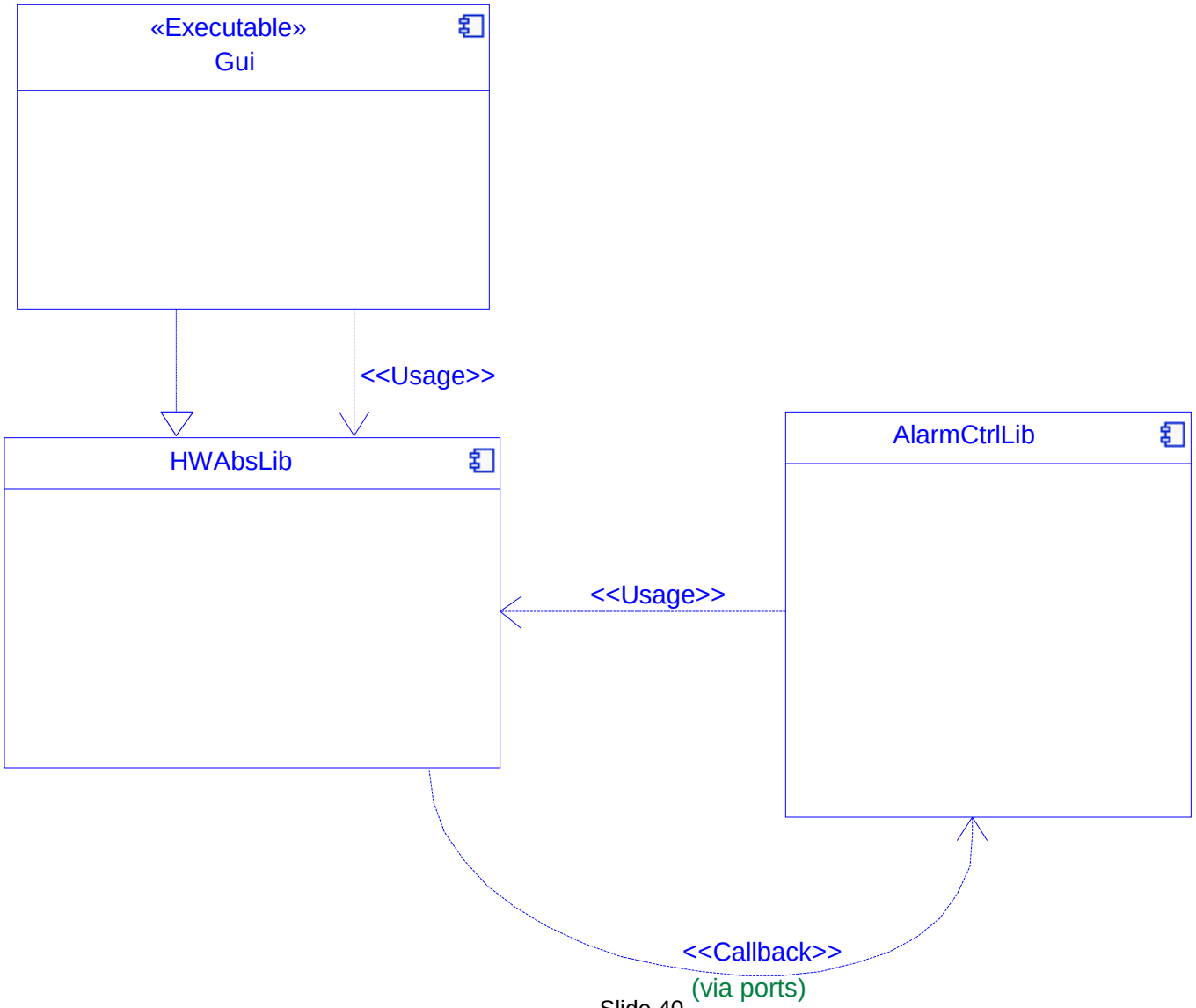
Test Setup

```
AlarmController theAlarmCtrl;
Hw theHW;
```

```
theHW.getCtrl()->setItsIKeyListener(
                    theAlarmCtrl.getHwObserver()->getItsIKeyListener());

theHW.getCtrl()->setItsIMovementListener(
                    theAlarmCtrl.getHwObserver()->getItsIMovementListener());

theHW.getCtrl()->setItsIDoorListener(
                    theAlarmCtrl.getHwObserver()->getItsIDoorListener());
```

```
theAlarmCtrl.getHwCtrl()->setItsILedCtrl(theHW.getCtrl()->getItsILedCtrl());

theAlarmCtrl.getHwCtrl()->setItsILightCtrl(theHW.getCtrl()->getItsILightCtrl());

theAlarmCtrl.getHwCtrl()->setItsISirenCtrl(theHW.getCtrl()->getItsISirenCtrl());
```

UML 2.0 Component Diagram



Summary

Component patterns and ports:

- Component-Based Architecture
- ROOM Pattern (=>UML 2.x Ports)
- Designing with Ports in UML 2.x