



IBM Software Group

## Building component-based architectures using UML2 ports and Ada 2005 interfaces

**Fraser Chadburn,**  
**Technical Professional**  
**IBM Rational UK**



Rational.

The Rational logo is displayed in white text within a blue rectangular box. Below the box, a series of concentric blue ripples, resembling water, are centered on the right side of the slide.

→ Go to IBM

A horizontal decorative bar at the bottom of the slide contains several small icons: a vertical bar with colored segments (cyan, green, yellow, red, purple), a crane, a circular arrow, a person's head, a globe, a network of nodes, and a square with four arrows pointing outwards. To the right of these icons is a button with a right-pointing arrow and the text 'Go to IBM'.

# Why use visual modeling?





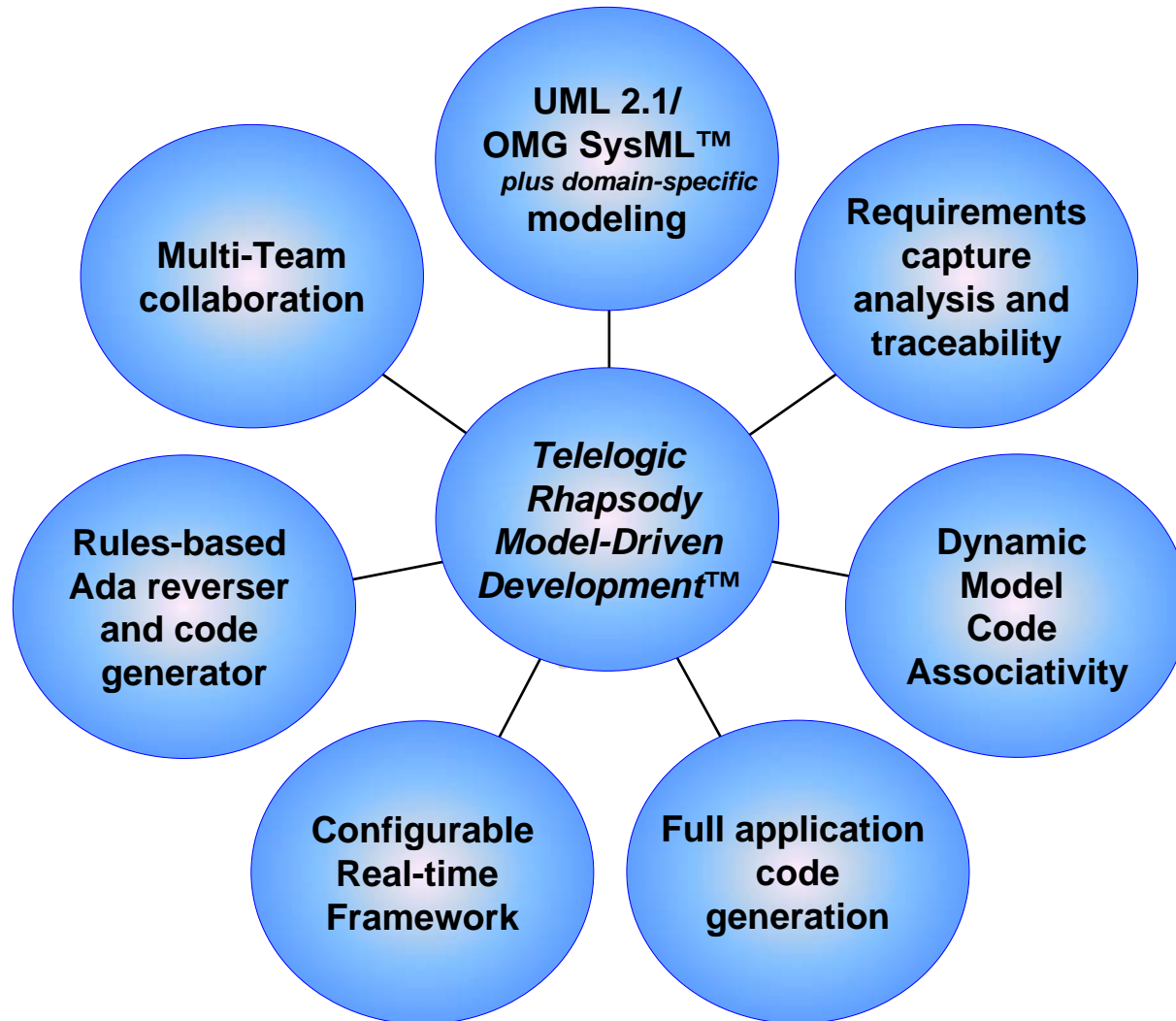
# Overview

- Introduction to Telelogic® Rhapsody® for Ada.
- What were the strengths and weaknesses of UML1?
- What are the advantages of UML2?
- What are the benefits of using ports and interfaces?
- How can these new UML2 constructs be used with Ada?
- Summary.

## What is Telelogic Rhapsody?

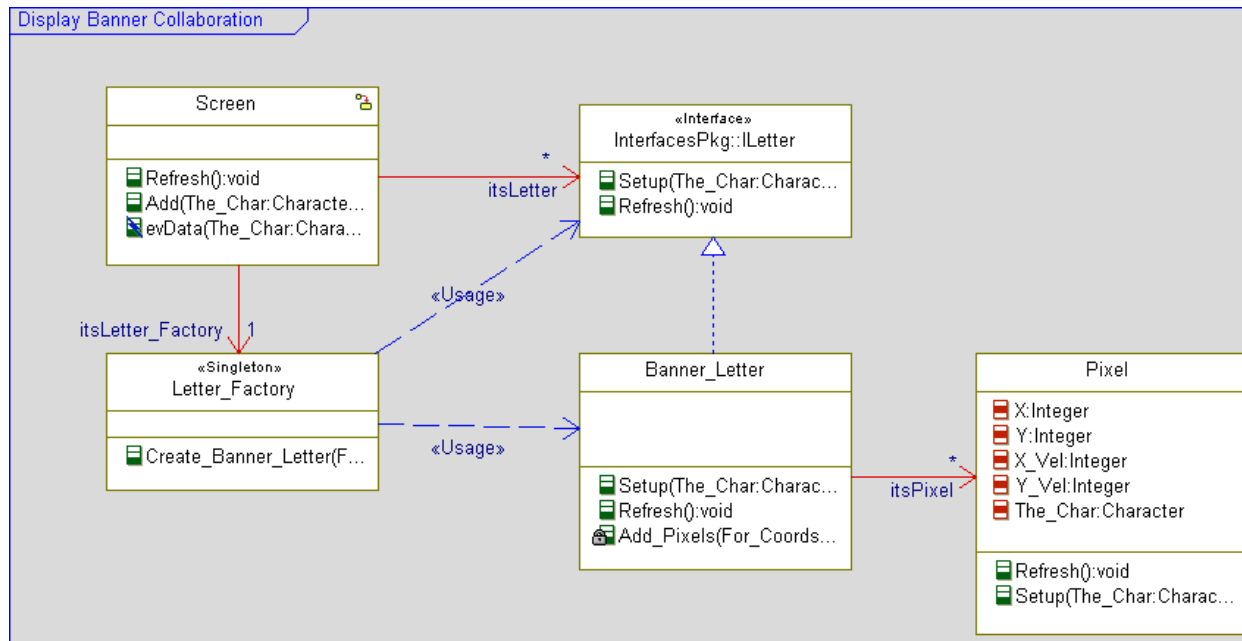
- Telelogic Rhapsody is a leading UML/OMG SysML™ tool focused on complex system design and validation.
- Origins:
  - ▶ I-Logix formed in 1997 focused on complex system design and validation – Statemate®.
    - Dr. David Harel – Behavior modeling/Israeli Prime Minister's Aware.
    - Dr. Amir Pnuelli – Formal Verification/Turing Award.
  - ▶ 1998 – New generation Unified Modeling Language™ (UML™) compliant platform for seamless systems design and development – Rhapsody®.
    - Eran Gery – UML methodologist/technologist.
    - Dr. Peter Hoffman – Systems methodologist.
    - Dr. Bruce Douglass – Software methodologist/author.
  - ▶ 2006 – I-Logix acquired by Telelogic.
  - ▶ 2008 – Telelogic acquired by IBM Rational.

# Key enabling technologies for Rhapsody in Ada



## What were the strengths of UML1?

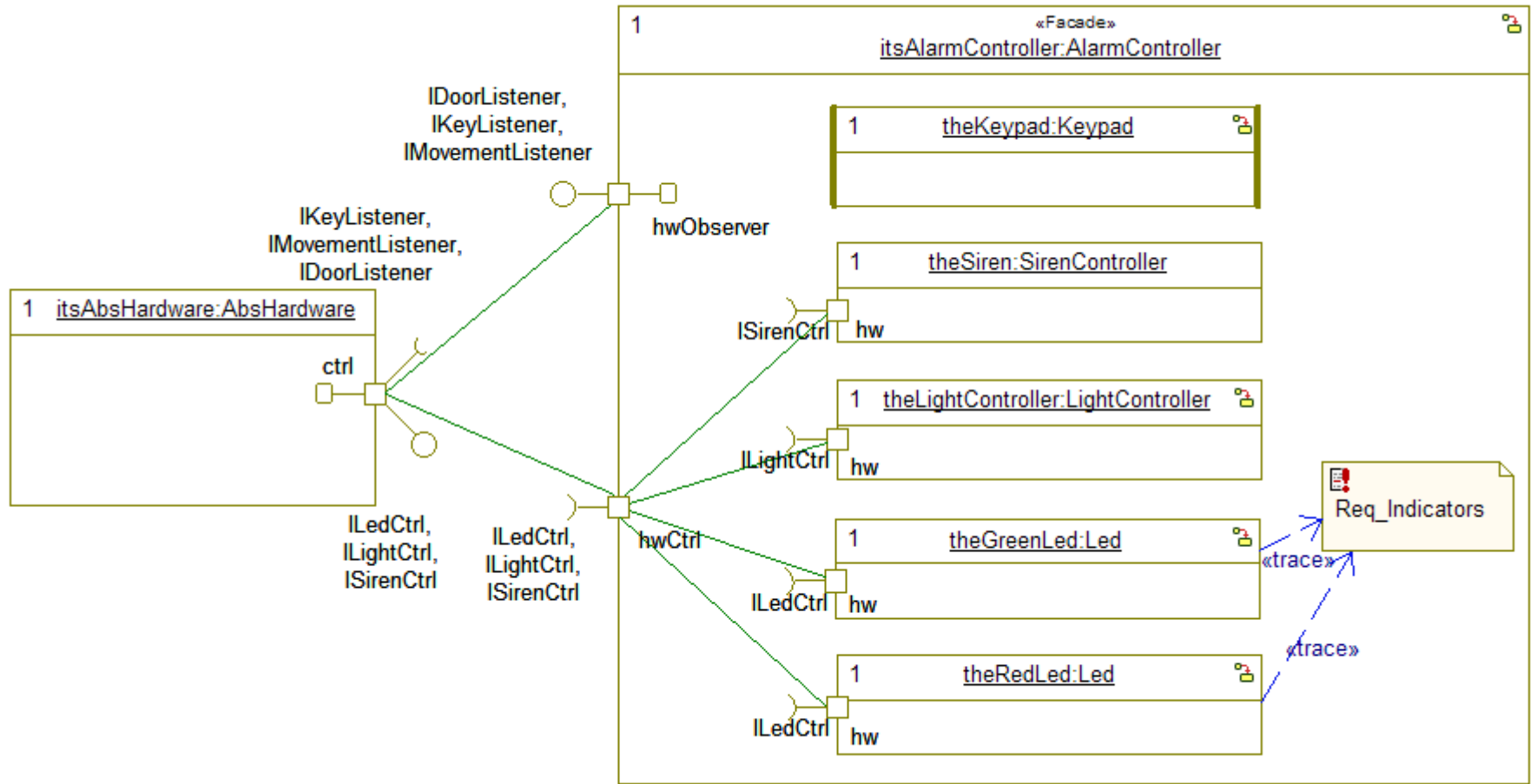
- The initial versions of the UML (UML1), an international standard, originated from three leading object-oriented methods (Booch, OMT, and OOSE).
- UML1 incorporated a number of best practices for object-oriented design. In particular for modeling object-orientated relations between classes.



## Modeling large-scale systems using UML2

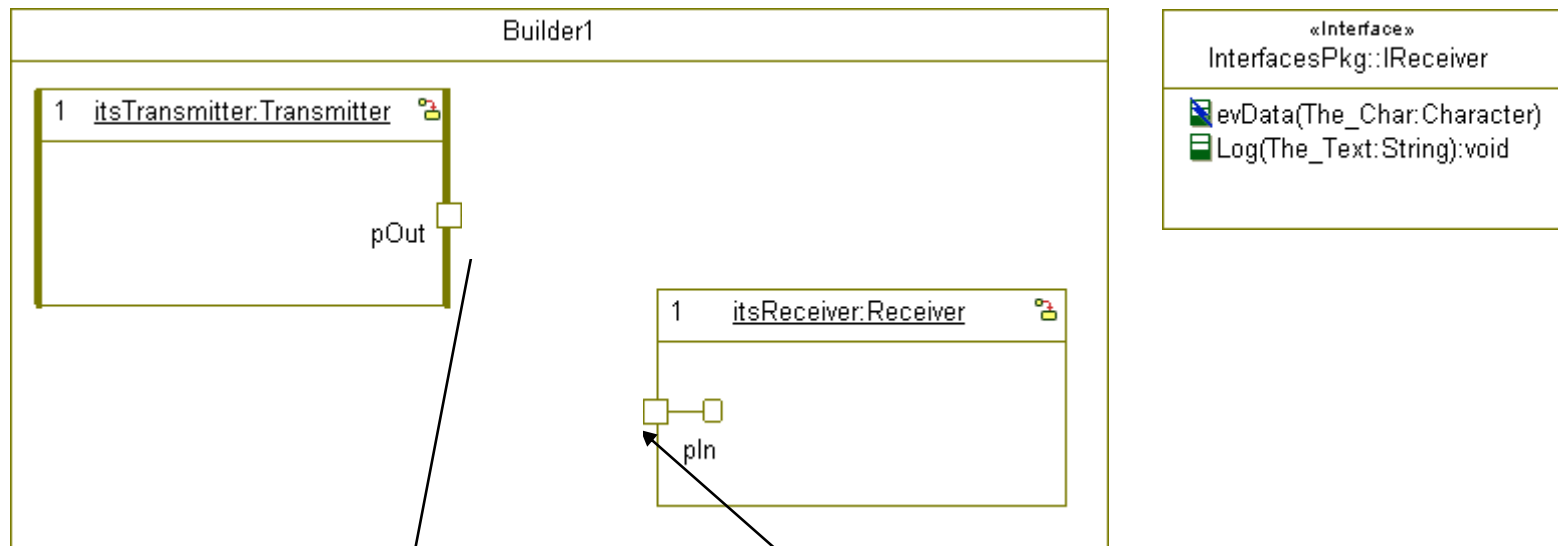
- When building large systems, we want to focus not only on how classes interrelate in terms of cohesion, but also how they are separated.
- The main enhancements in UML2 were to support modeling large-scale systems by introducing a new concept called **composite structures**.
- A “structured” class is one which contains other classes. The instances of classes inside a class are known as **parts**.
- Classes can be composed of classes which themselves have parts, supporting the hierarchical decomposition of a system.

home alarm overview



## UML2 ports and interfaces

- Ports represent named interaction points between an instance of a classifier (e.g. class or actor) and its environment.
- Interfaces can be used to specify the nature of the interactions in terms of a set of operation specifications (signatures).

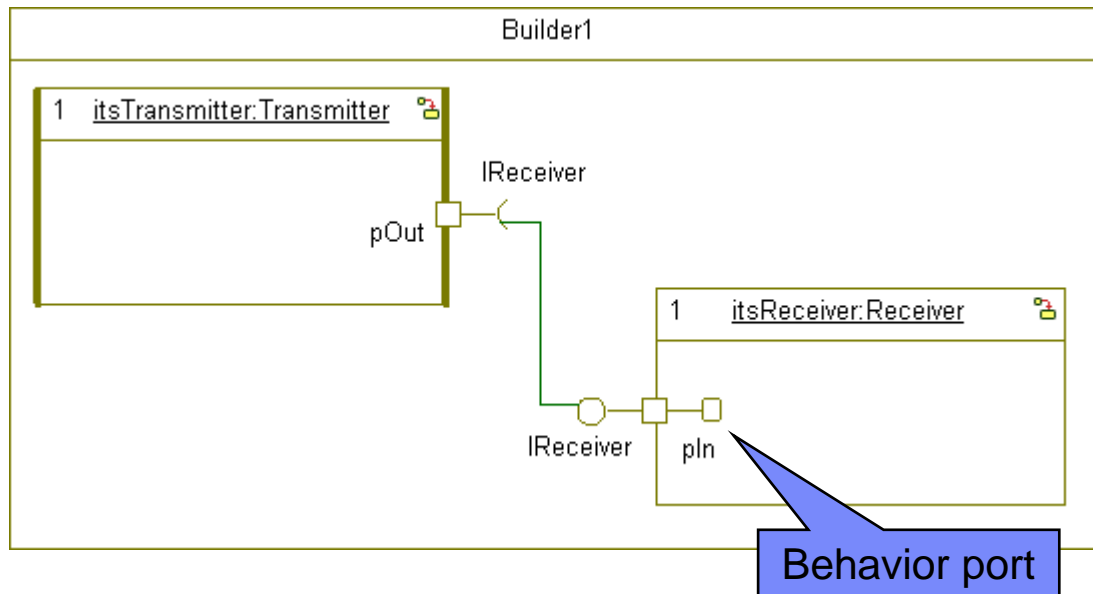


“Required interfaces” characterize outgoing requests that may be made to its environment through the port.

“Provided interfaces” characterize incoming requests that the environment may make through the port.

## UML2 connectors

- Connectors specify a link that enables communication between two or more instances. This can be realized automatically in a code-generator rule set.
- You can get the Rhapsody code generator to automatically synthesize the wiring code and build files to create an executable.



```

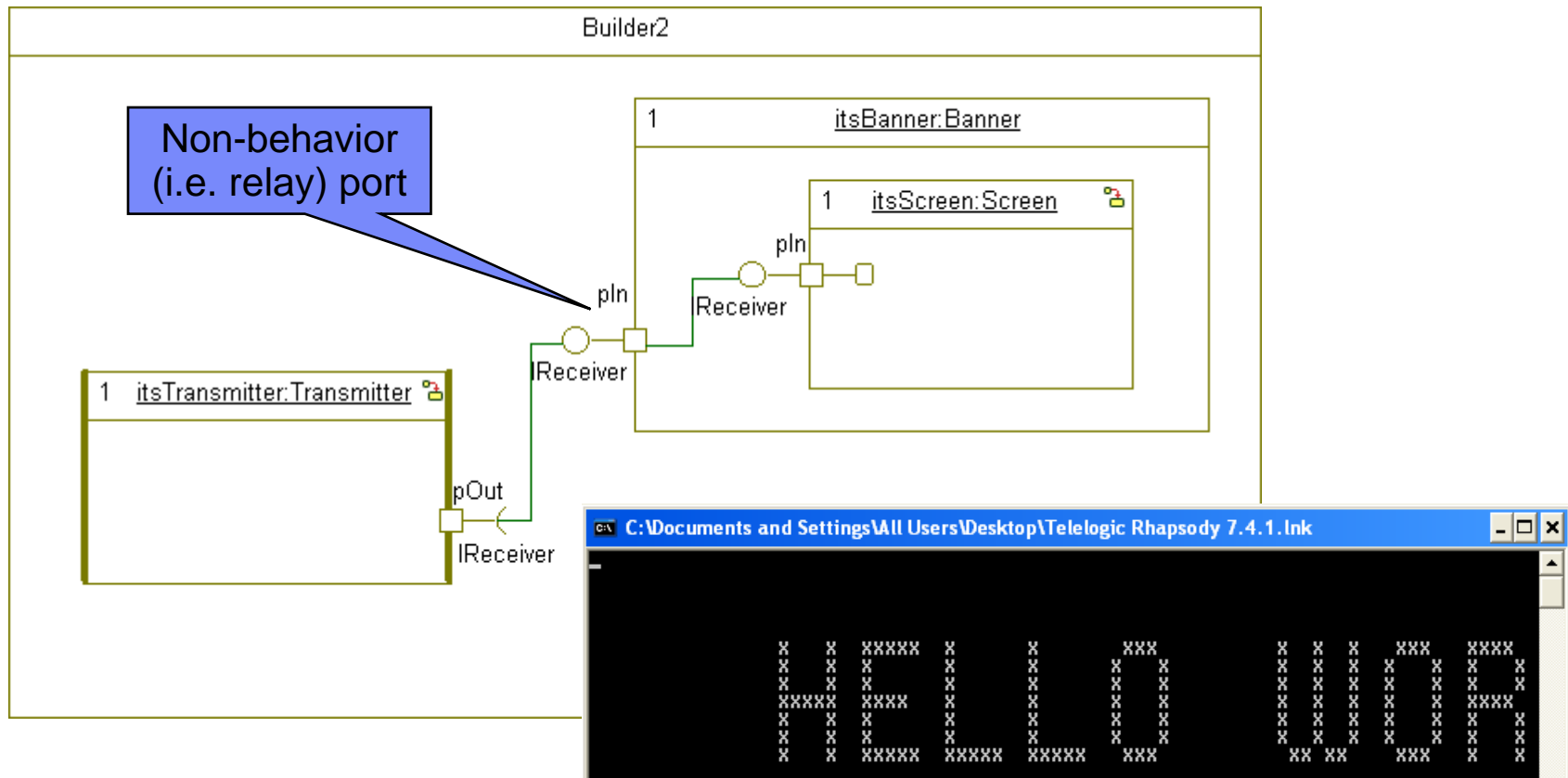
C:\Documents and Settings\All Users\Desktop\
Tx: 'H'      Rx:'H'
Tx: 'e'      Rx:'e'
Tx: 'l'      Rx:'l'
Tx: '1'      Rx:'e'
Tx: 'o'      Rx:'l'
Tx: ' '      Rx:'l'
Tx: 'W'      Rx:'l'
Tx: 'o'      Rx:'o'
Tx: 'r'      Rx:'o'
Tx: 'l'      Rx:' '
Tx: 'd'      Rx:' '
Tx: '?'      Rx:' '
  
```

## Benefits of ports and interfaces

- Supports clear separation of interfaces at the component level.
- Decouples components that “require” services from the “provider”.
- Supports the separation of concern across multiple teams.
- Allows parallel development of components prior to assembly.
- Powerful construct for subsystem definition.
- Eases subsequent integration.
- Supports re-use of components in different contexts.

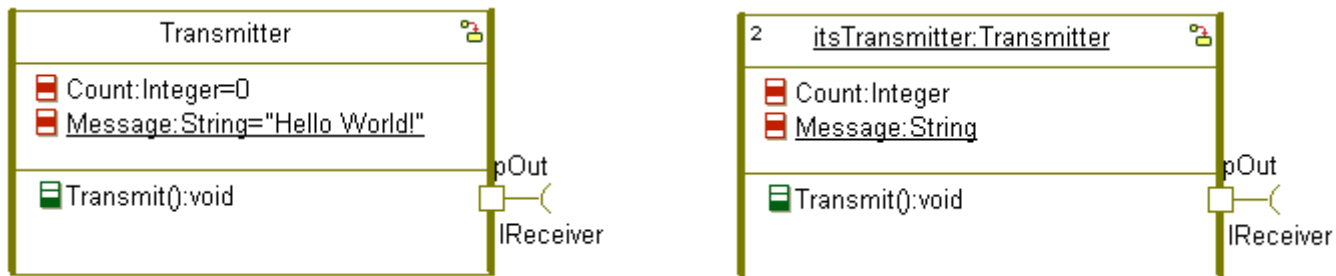
# What are the benefits of ports?

- Different concrete implementations can be used providing they “realize” the same interfaces.



## How can classes and objects be realized in Ada?

- The concept of a class is often synthesized in Ada using an Ada **record** to define the attributes of the class and an Ada **package** to define the operations.
- Since operations are performed on objects, we pass the Ada record as the first parameter in the operation call, e.g., a “this” parameter.
- This can be synthesized automatically using a code generator to ensure a consistent approach. Note: It is possible to turn off the OO-mapping.



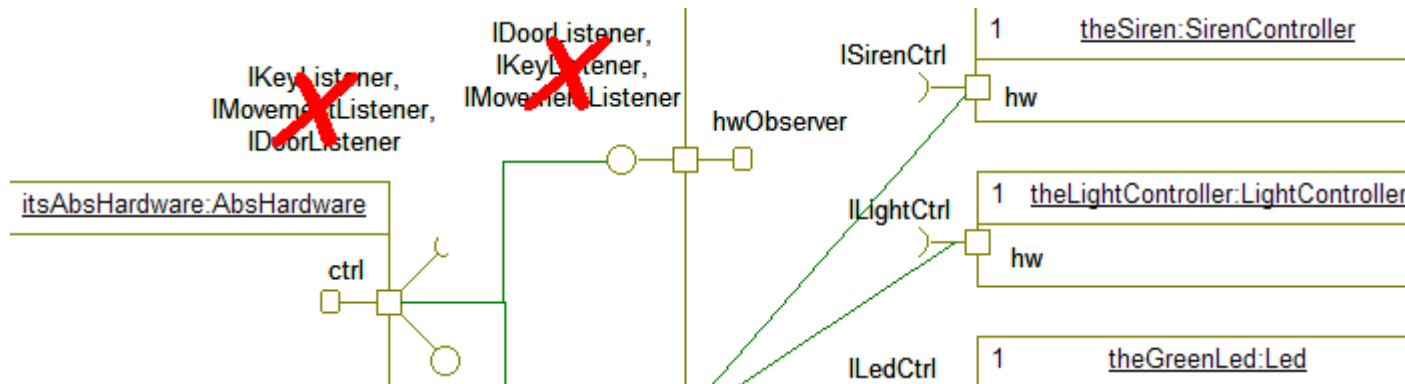
## Constraints associated with Ada 95

- In Ada 95 it is possible to synthesize UML interfaces by use of an Ada **package** and an **abstract tagged null record** type.

**package** IDoorListener is

**type** IDoorListener\_t is **abstract tagged null record**;

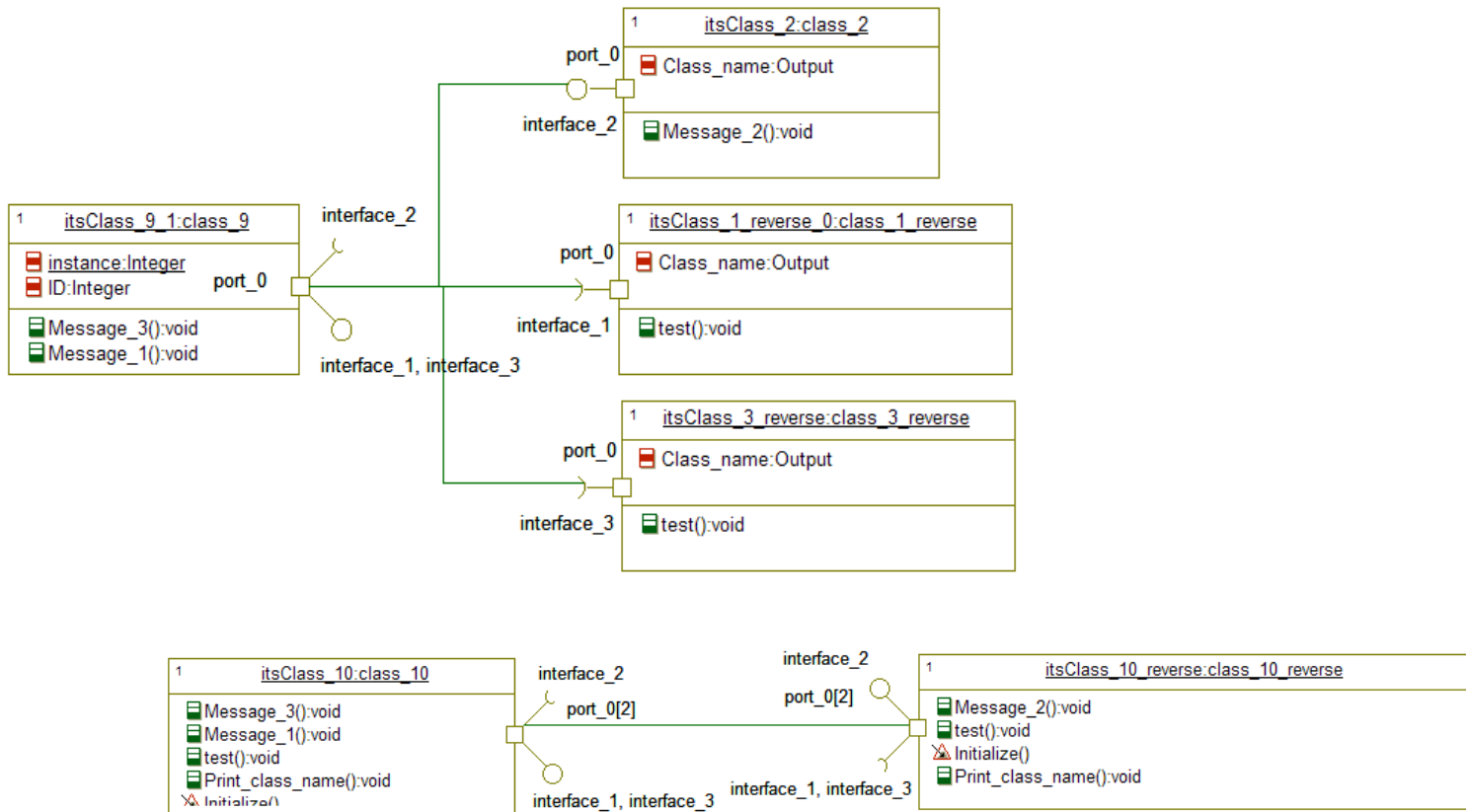
**procedure** On\_Door (This : **in out** IDoorListener\_t) **is abstract**;



- Unfortunately, Ada 95 only permits a derived type to have one immediate ancestor.

## Different behavioral port and interface assemblies

- In UML2, a classifier can have any number of ports and ports may be typed by multiple provided and/or required interfaces.



## Exploiting Ada 2005 interfaces with ports

- Ada 2005 resolves the limitations of single inheritance by introducing Java-like **interfaces**.

```
package IDoorListener is
```

```
  type IDoorListener_t is interface;
```

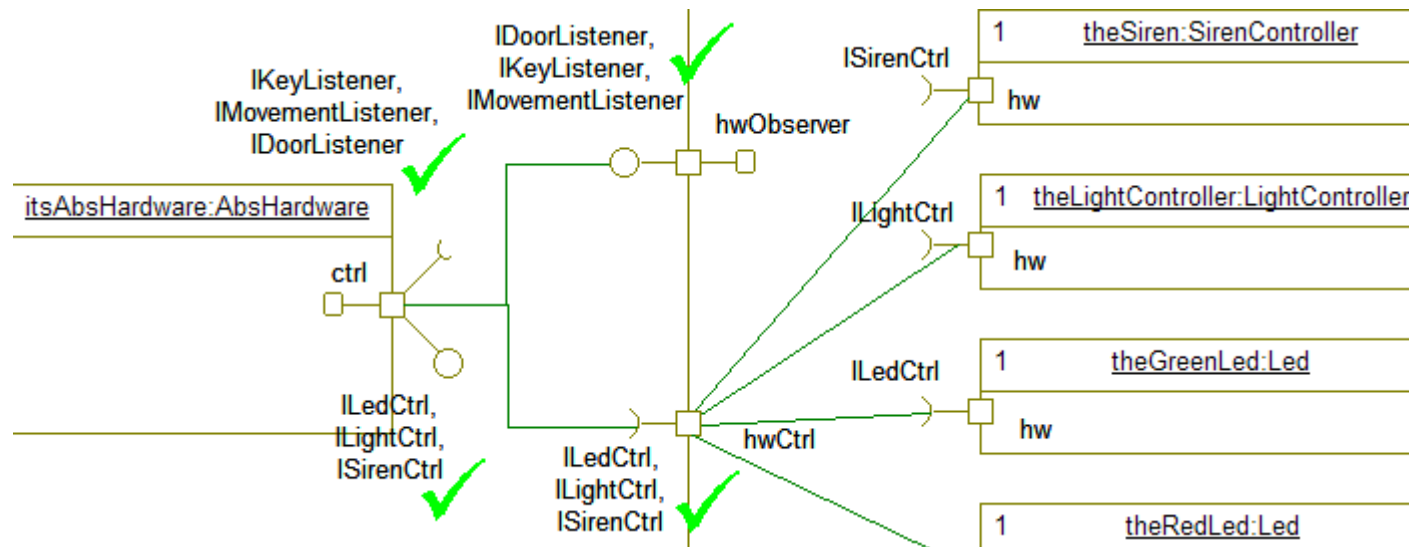
```
  procedure On_Door_Open (This : in out IDoorListener_t) is abstract;
```

```
  procedure On_Door_Close (This : in out IDoorListener_t) is null;
```

- Unlike an abstract type, an Ada **interface** is used solely for defining the subprogram signatures.
- All interface types in Ada 2005 are implicitly tagged types and a type can be derived from multiple parent and progenitor types.

## Exploiting Ada 2005 interfaces with ports

- Using Ada 2005 interfaces we can support multiple ports and allow ports to have multiple provided and/or required interfaces.



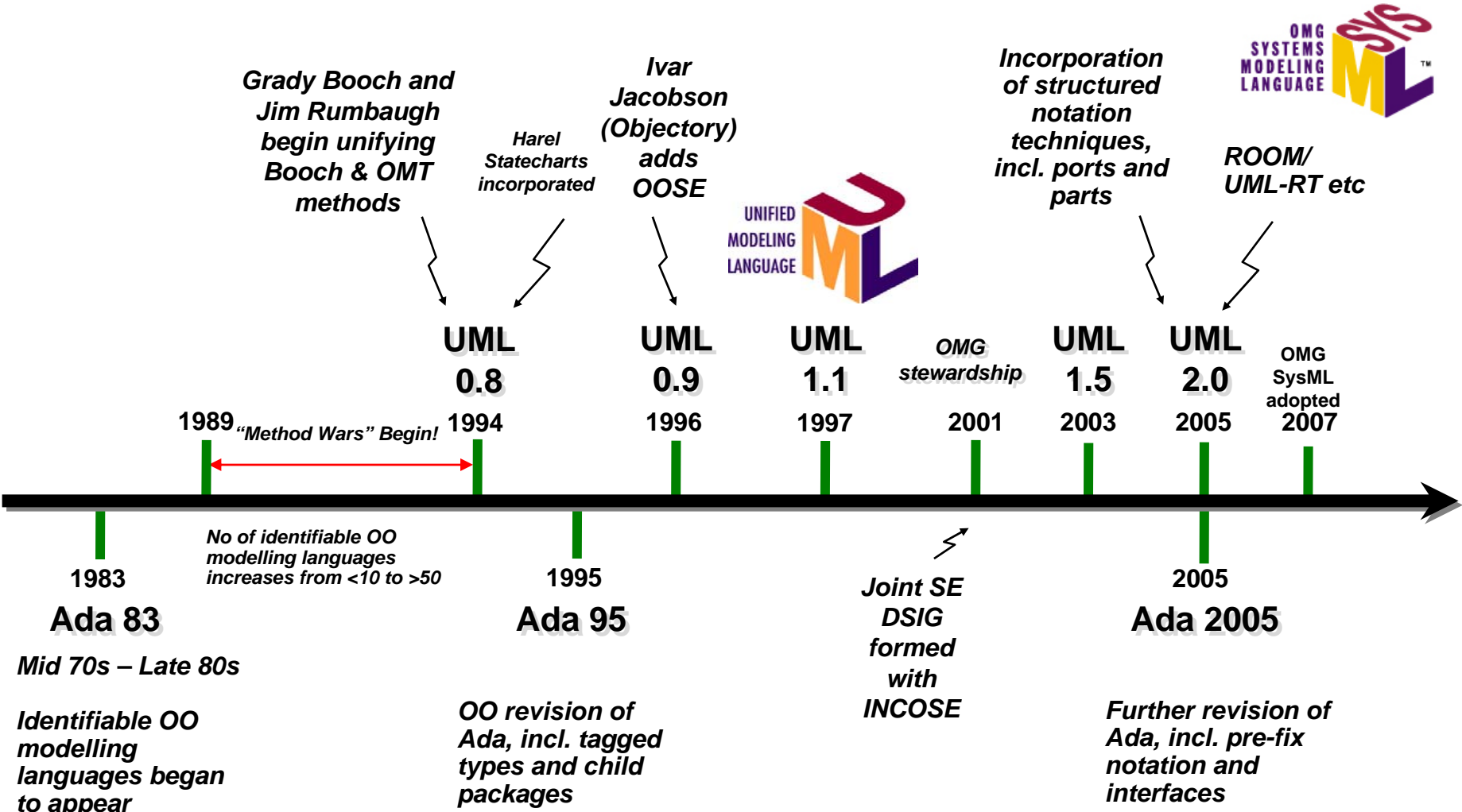
## Key benefits of using Ada 2005 interfaces

- Enables homogenous support for UML2 code-based modeling techniques.
- “Plug and play” components using a graphical notation.
- Re-use of existing components in new assemblies.
- Clear definition of external interfaces for testing.
- Clean and readable code generation.
- Clear and relevant compiler feedback.
- Exploit full application generation.

## Benefits of model-driven development in Rhapsody?

- Create software faster.
- Improve predictability, consistency, quality.
- Manage complexity.
- Improve project oversight.
- Enforce relevance of design.
- Provide compelling evidence for certification/audit.

# Chronology of Ada vs. UML development





[fraser.chadburn@uk.ibm.com](mailto:fraser.chadburn@uk.ibm.com)



© Copyright IBM Corporation 2008. All rights reserved.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo, the on-demand business logo, Rational, the Rational logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.