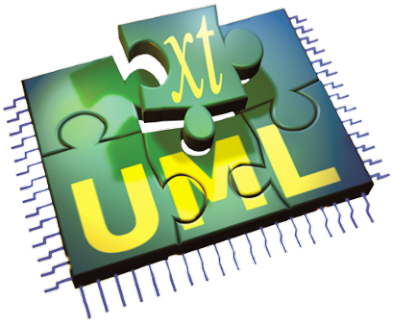
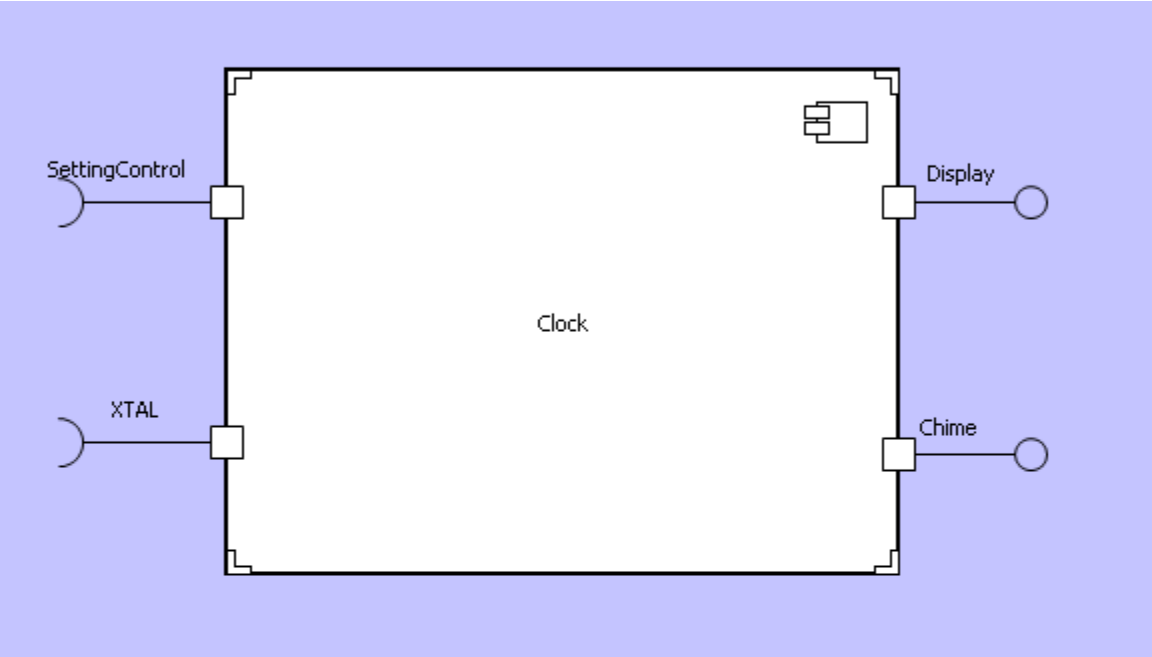
The xtUML method - Building Component Diagrams

- **Analysis** – questioning, thinking, sketching...
  - Descriptive UML diagrams
    - use case, sequence, ...
- **Executable Modeling** – formalizing the analysis:
  - **Component Diagrams (partitioning/interfaces)**
  - Class Diagrams (data)
  - State Machines (control)
  - Activities (processing)
- **Verification**
  - Interpretive Model Execution
- **Code generation**
  - Template and Rule-Based Translation



UML 2.0 Component Definition

- A modular part of a system design that hides its implementation behind a set of external interfaces
- Within a system, components satisfying the same set of interfaces may be substituted freely



Component Packages

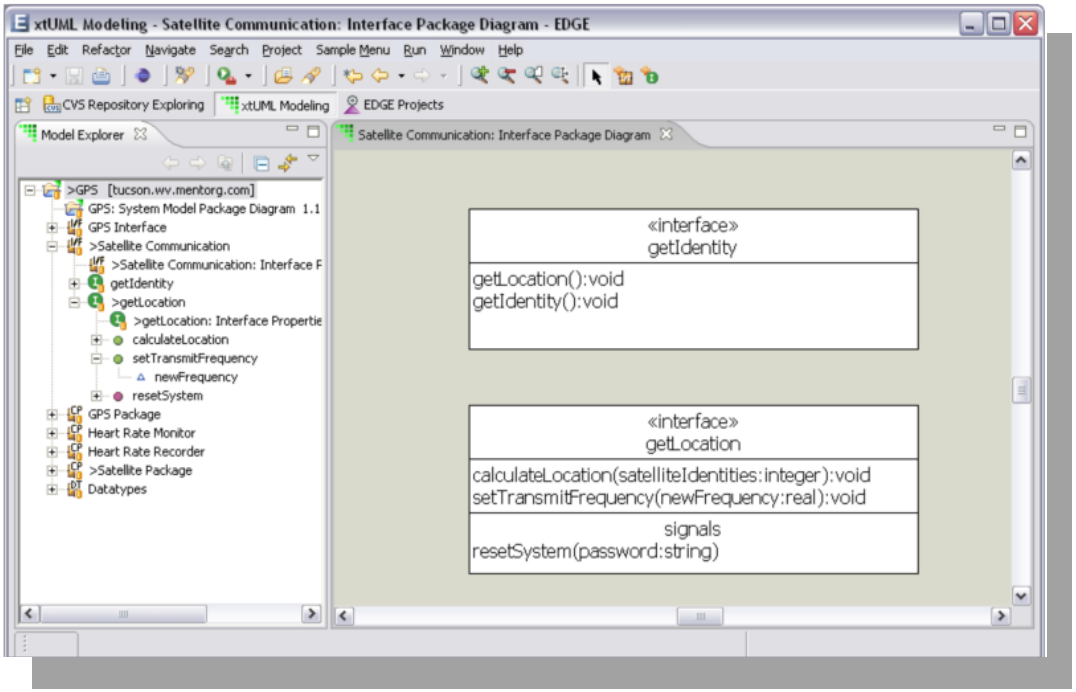
- Packages are a diagrammatic way to group modeling elements together and manage their hierarchy.
- Components are defined in packages.
- Packages can be nested.
- Component packages may contain other component packages and/or interface packages.
- Interfaces also are organized in packages.
- This allows a level of organizational association between components and their interfaces.

Interfaces

- An interface is a declaration of a collection of synchronous and asynchronous messages
- Managed separately from components
- Components implement Interfaces
- The separation also allows more than one component to implement (require or provide) any particular interface.
- Interfaces can be resolved from a different hierarchical path in the project and reused.
- Implementation can be specified at the individual component port and is unique for each component.

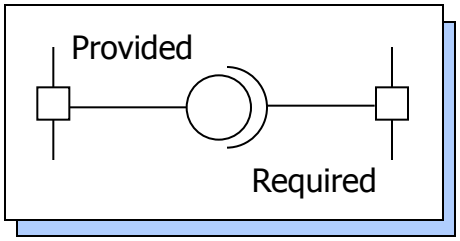
Interface Editor

- Three sections in graphic
  - Name
  - Operations
    - Synchronous message
    - Operation completes before further execution
    - Can carry return values
  - Signals
    - Asynchronous message
    - Execution resumes immediately after signal sent
    - No return value



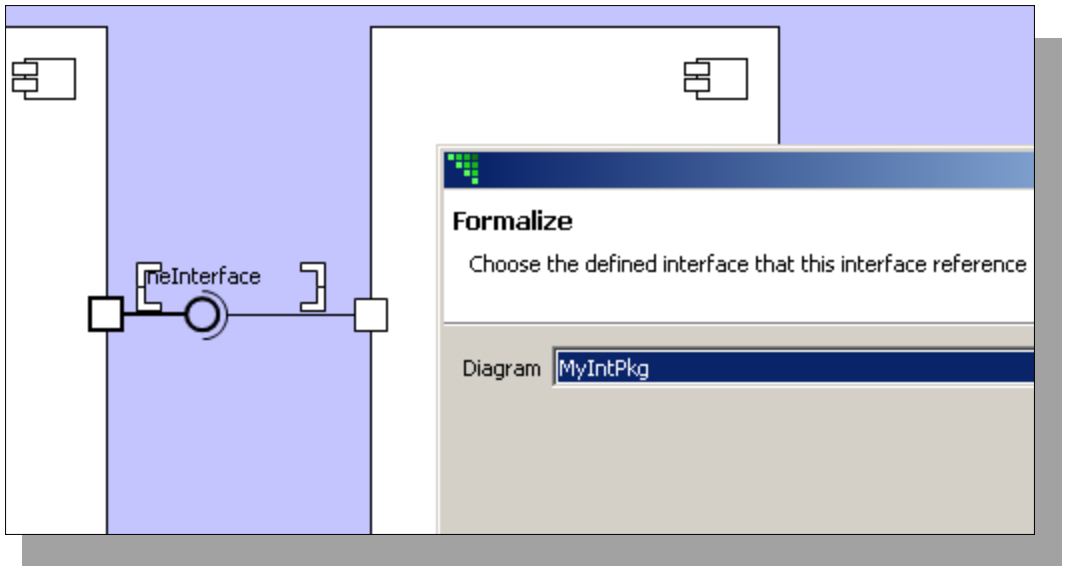
Provided vs. Required Interfaces

- **Provided Interface**
  - **“The Ball”**
  - **Allows a component to provide services to other components**
- **Required Interface**
  - **“The Cup”**
  - **Allows a component to demand services from another component**
- **Ports provide a name.**
  - **Necessary if an interface is multiply used.**



Formal Interfaces

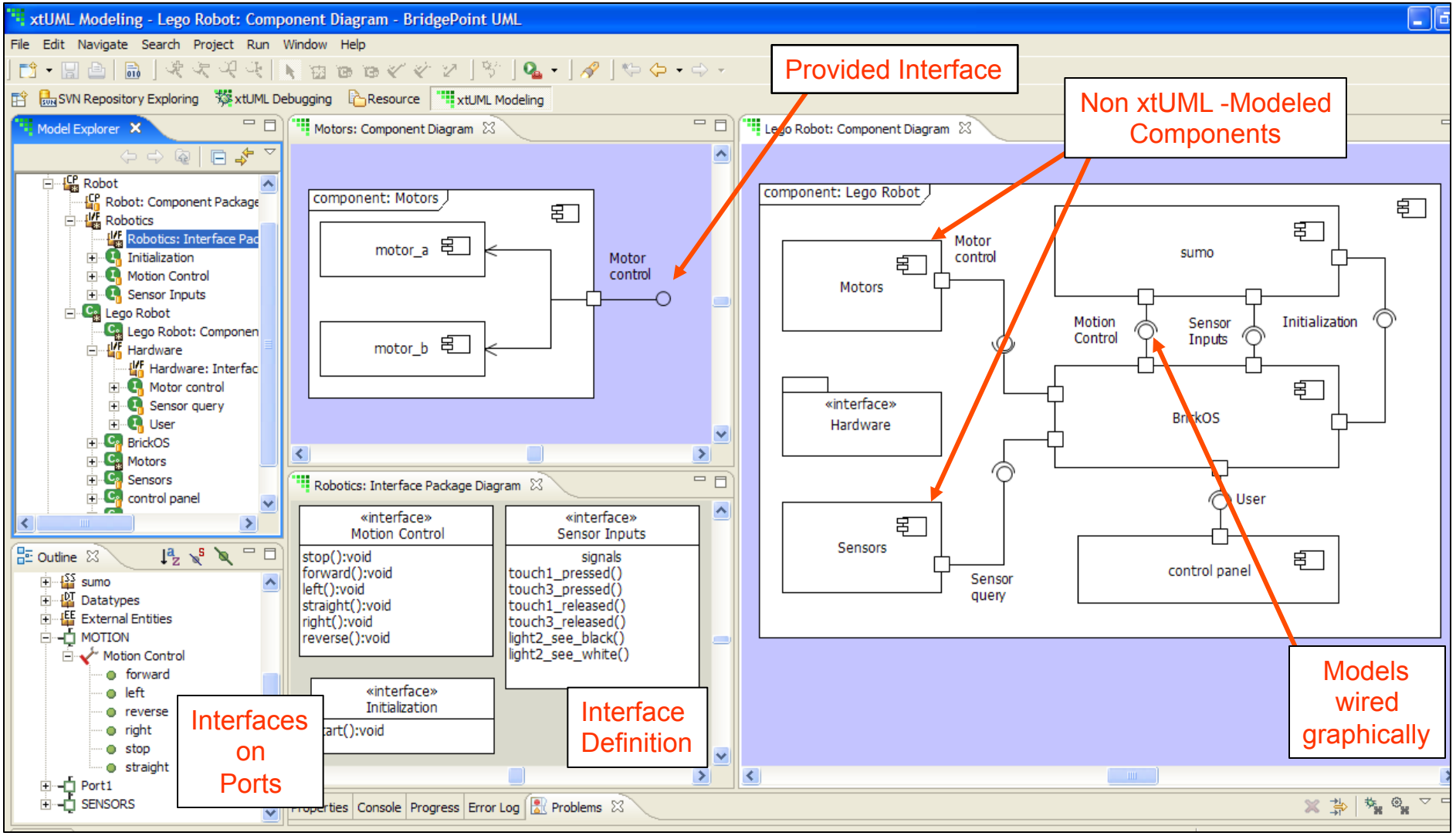
- When an interface is added it is not named or formalized.
- Name interface for clarity
- Formalize to interfaces declared in interface packages.
- Interfaces must be formalized to pass messages.



Termination of Interface Messages

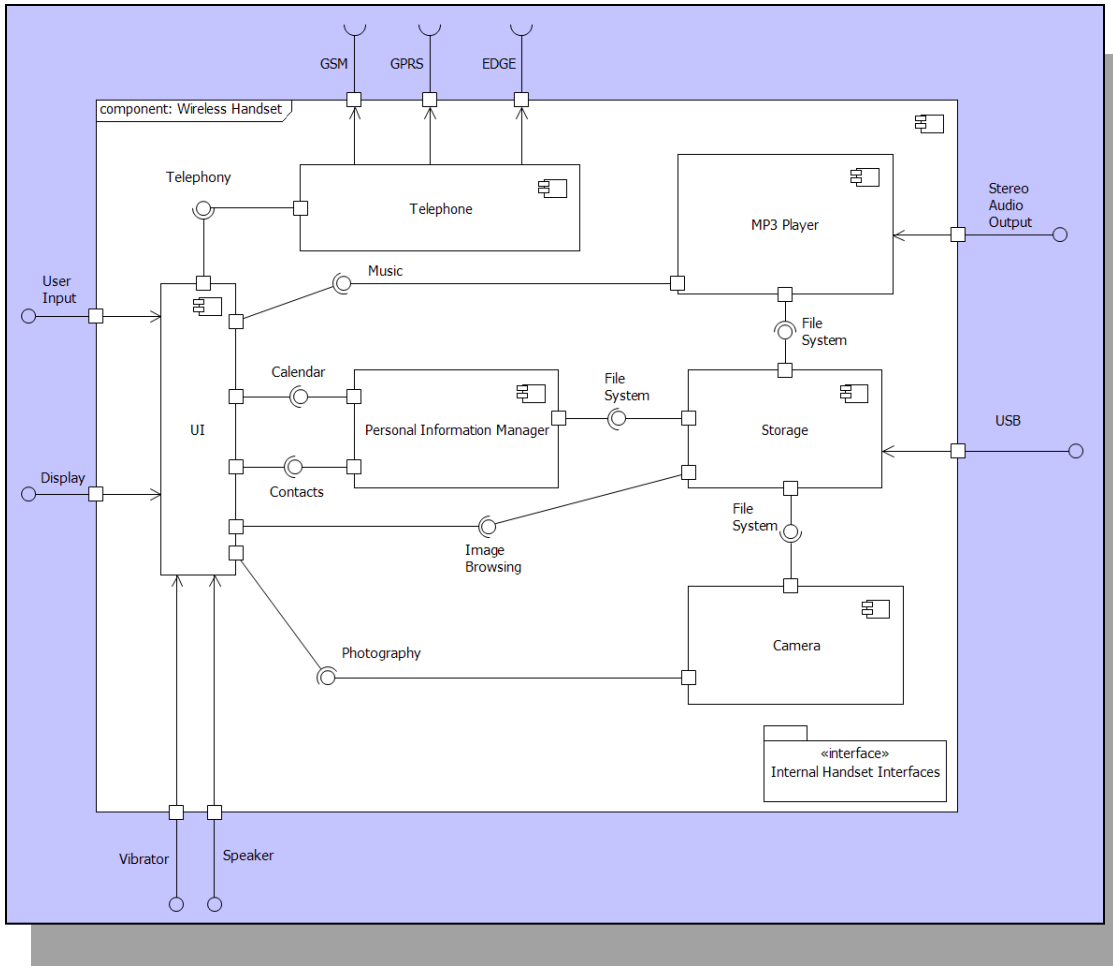
- Signal (async message) between two connected components:
  - Class based state machine (mapped event), or
  - Receiving port
- Signal sent from single component
  - Sending port
- Operation (sync message) between two connected components:
  - Receiving port
- Operation invoked from single component:
  - Sending port

Component Views



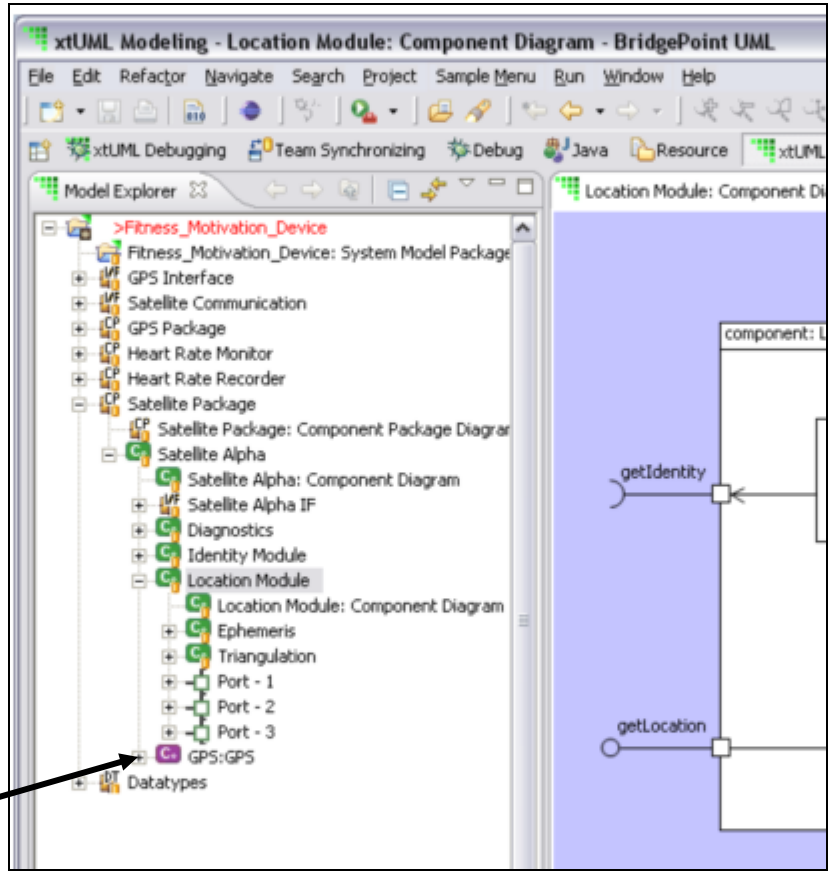
Nesting Components

- **Internal interfaces**
- **Delegation**
  - **Provided and required interfaces to Parent Component**
- **Leaf components:**
  - **xtUML models**
    - **Class,**
    - **State Machine**
    - **Actions**
  - **Legacy code**



Component References (vs. Definitions)

- Placeholder for a component.
- Allows for multiple implementations to be swapped conveniently.
- Multiple configurations with different components in Verifier.



Purple Glyph

Lab 1: Exercises 2, 3

- **Continue embellishing the model with components and interfaces to capture communication boundaries.**