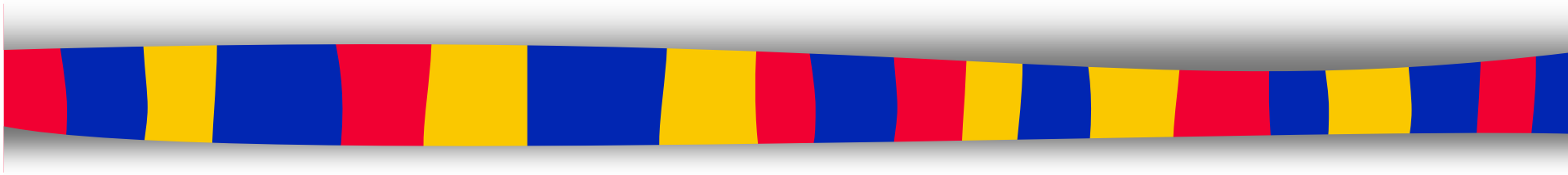
CISC 322
Software Architecture



Lecture 07:
Architecture Styles (2)

Emad Shihab

Adapted from Ahmed E. Hassan and Spiros Mancoridis

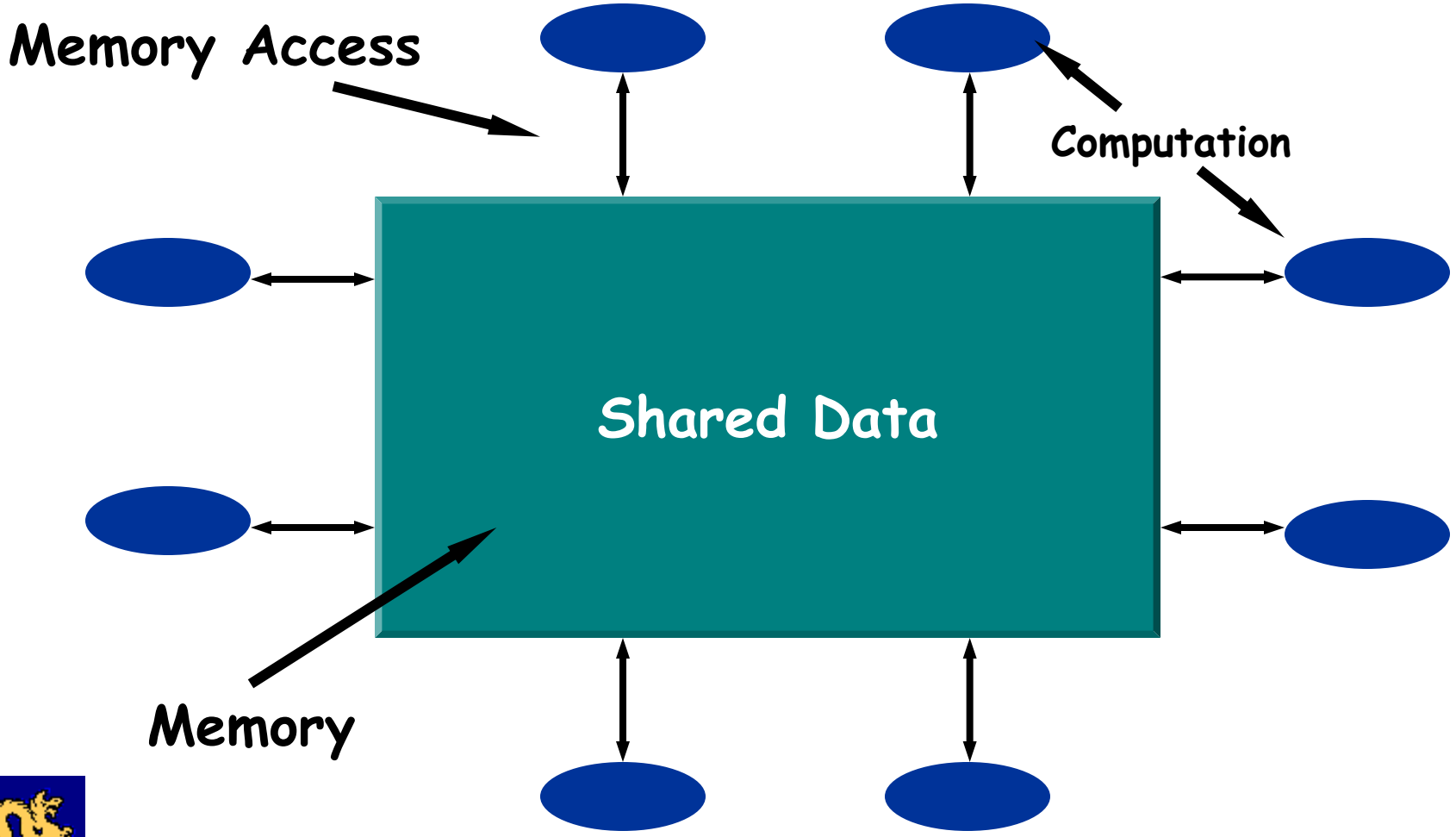
Announcement

- Professional Internship Program
  - Sep. 22 @ 5:30 in Goodwin Hall 247
- Quizzes and teaching style

Last Class Recap

- Architectural styles are used to:
  - Communicate between stakeholders
  - Document design decisions
  - Support sharing of styles for similar software systems
- Repository – e.g. World of War Craft
- Pipe and Filter – e.g. Traditional compilers

Repository Style



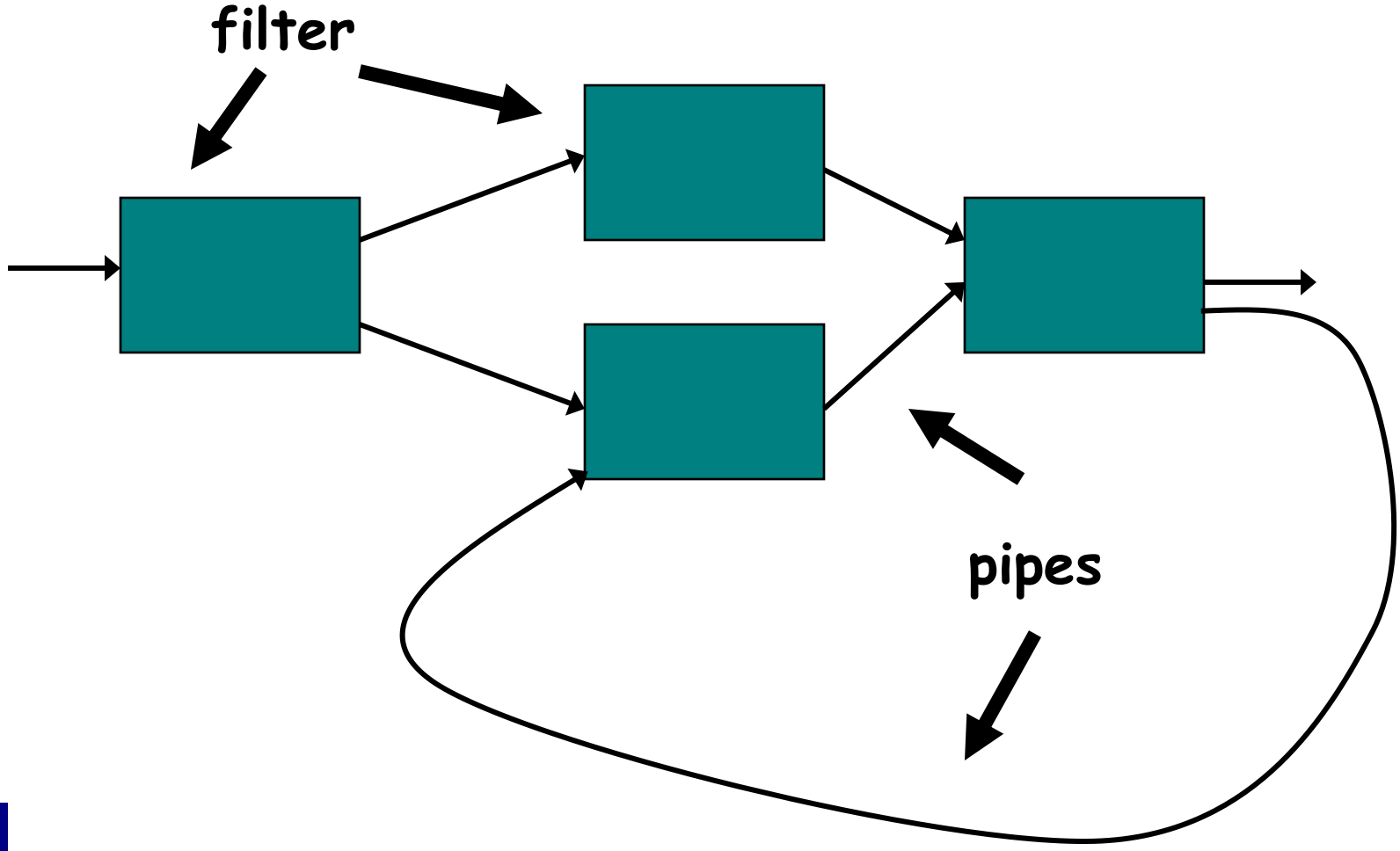
Summary of Repository Style

- Independent components (programs) access and communicate exclusively through global repository

- Advantages
  - Efficient storage of data
  - Easily manageable
  - Can solve complex problems

- Disadvantages
  - Evolving data is expensive
  - Cannot handle high volume or complex logic

Pipe and Filter Architectural Style



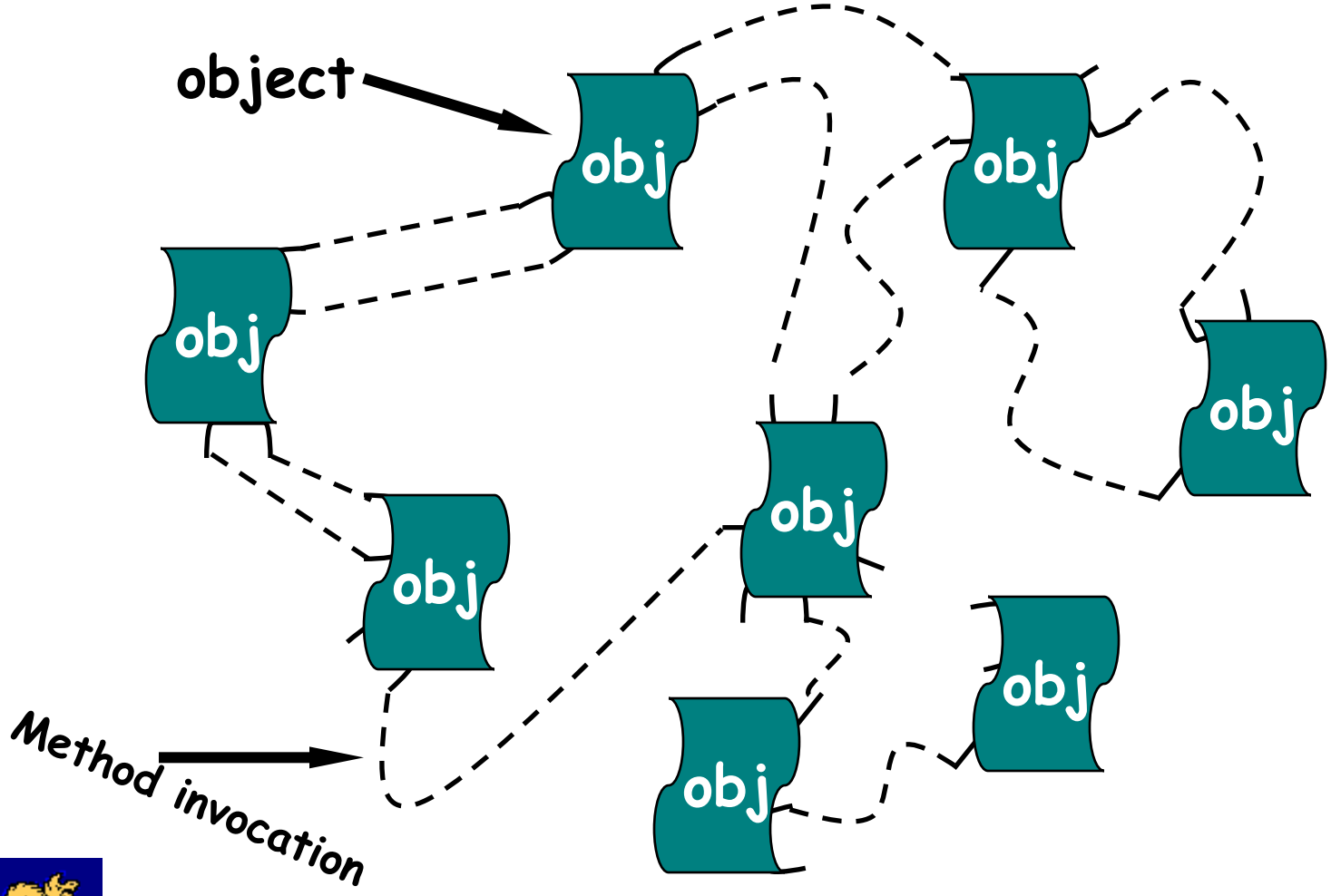
Summary of Pipe-and-Filter Style

- Independent components connected by pipes that route data streams between filters

- Advantages
  - Easy to understand
  - Easy to maintain and enhance

- Disadvantages:
  - Poor performance
  - Increased complexity

Object-Oriented Style



Object-Oriented Style

- Data representations and their associated operations are encapsulated in an **abstract data type**

- **Components:** are **objects**.
- **Connectors:** are function and procedure invocations (**methods**).

Object-Oriented Style

- Topology: Arbitrary
- Maximize Cohesion
  - Operate only on your own data
- Minimize Coupling
  - Minimize dependencies between objects

Object-Oriented Invariants

- Objects are responsible for preserving the **integrity** of the data
  - Data only manipulated by appropriate functions

- The data representation is **hidden** from other objects (information hiding)

Object-Oriented Advantages

- Object can change **the implementation without affecting its clients**.

- Can **design** systems as collections of autonomous interacting agents.
  - Since accessing routines bundled with data

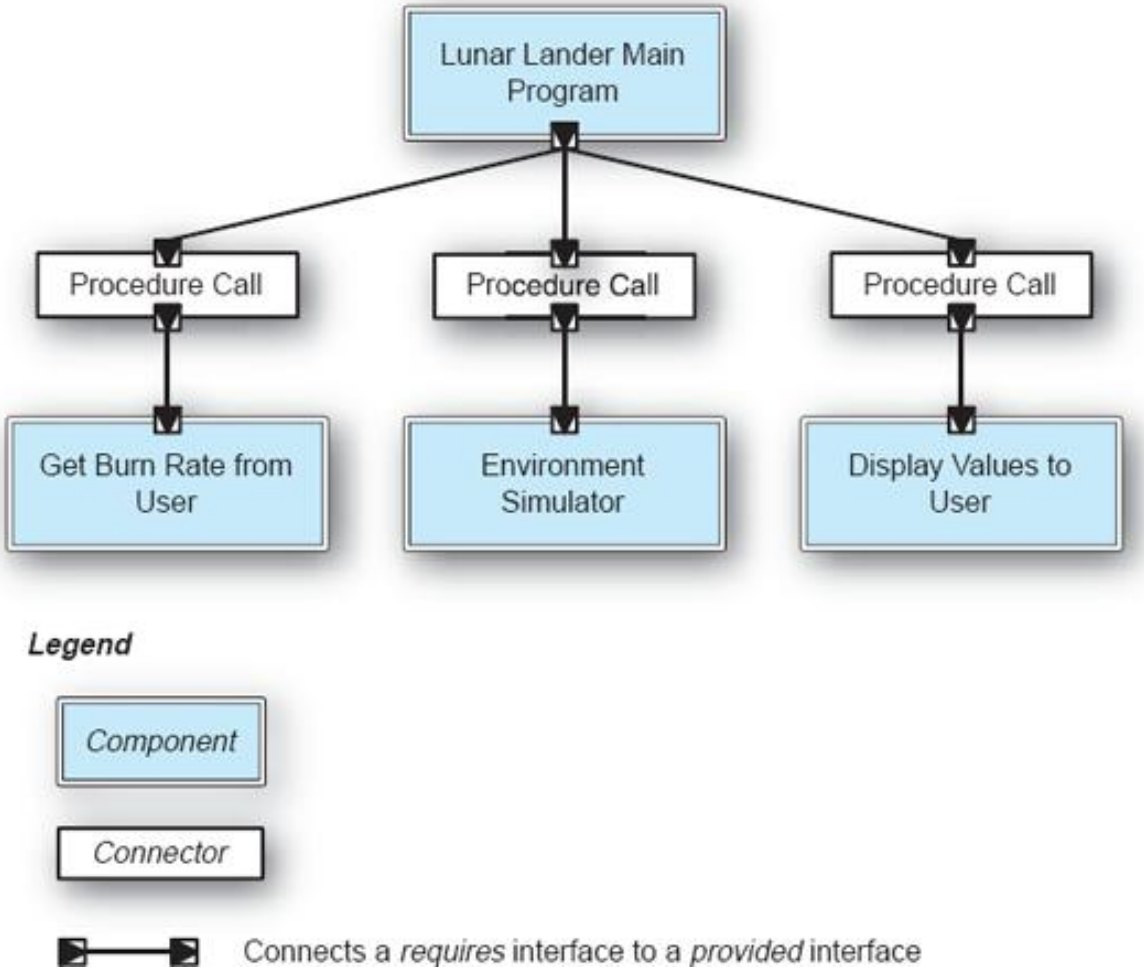
Object-Oriented Disadvantages

- Objects need to identify other objects they want to interact with
  - Contrast with *Pipe and Filter* Style
  - What if identity of an object changes?

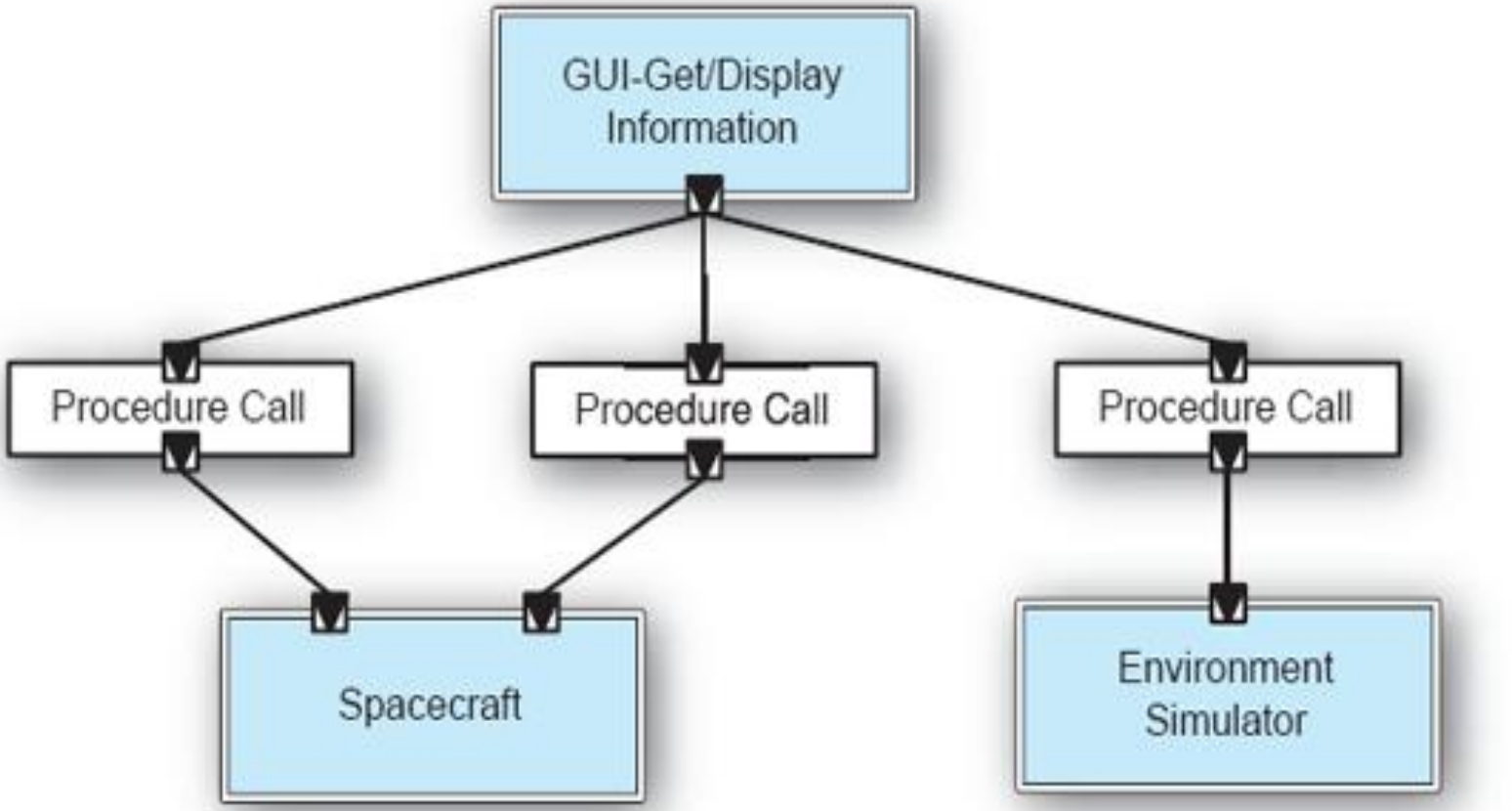
- Objects cause **side effect problems**:
  - *E.g.*, *A* and *B* both use object *C*, then *B*'s effects on *C* look like unexpected side effects to *A*.

Taylor et al. 2010

Main Program Lunar Lander Example



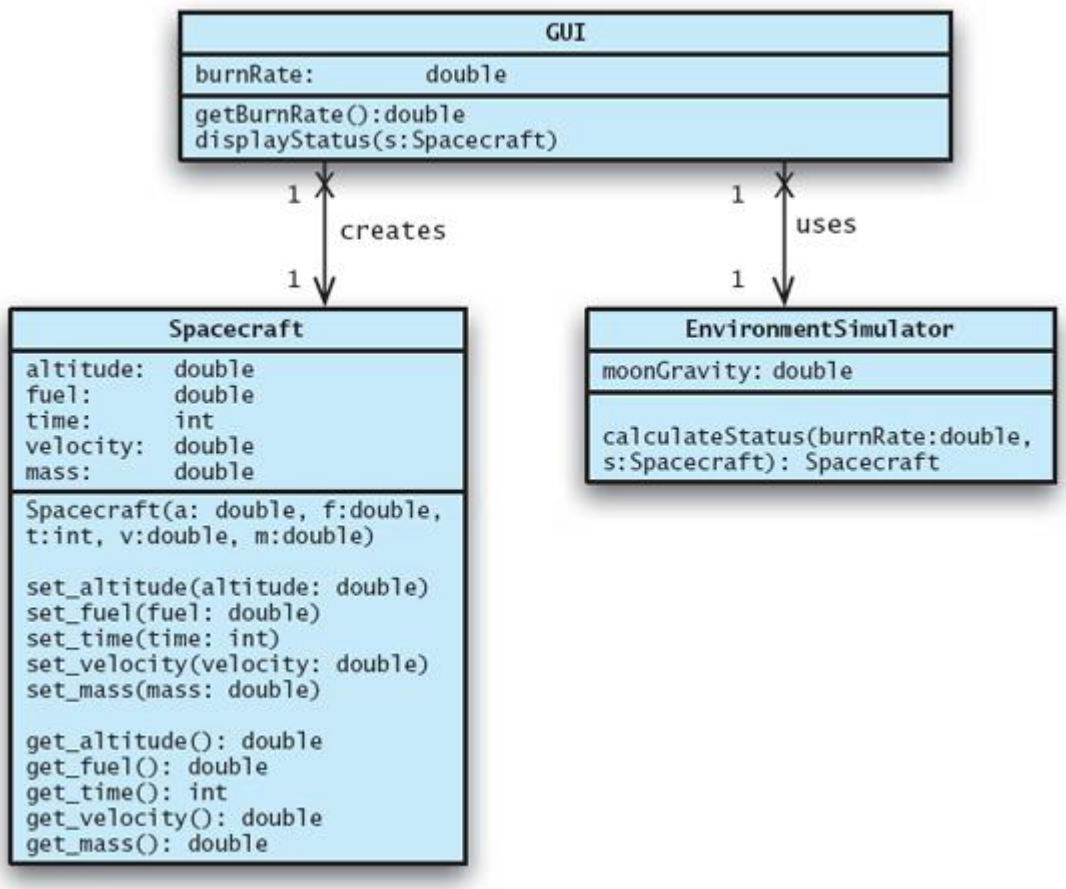
Object-Oriented Lunar Lander Example



Interaction with the user are handled by one object

Taylor et al. 2010

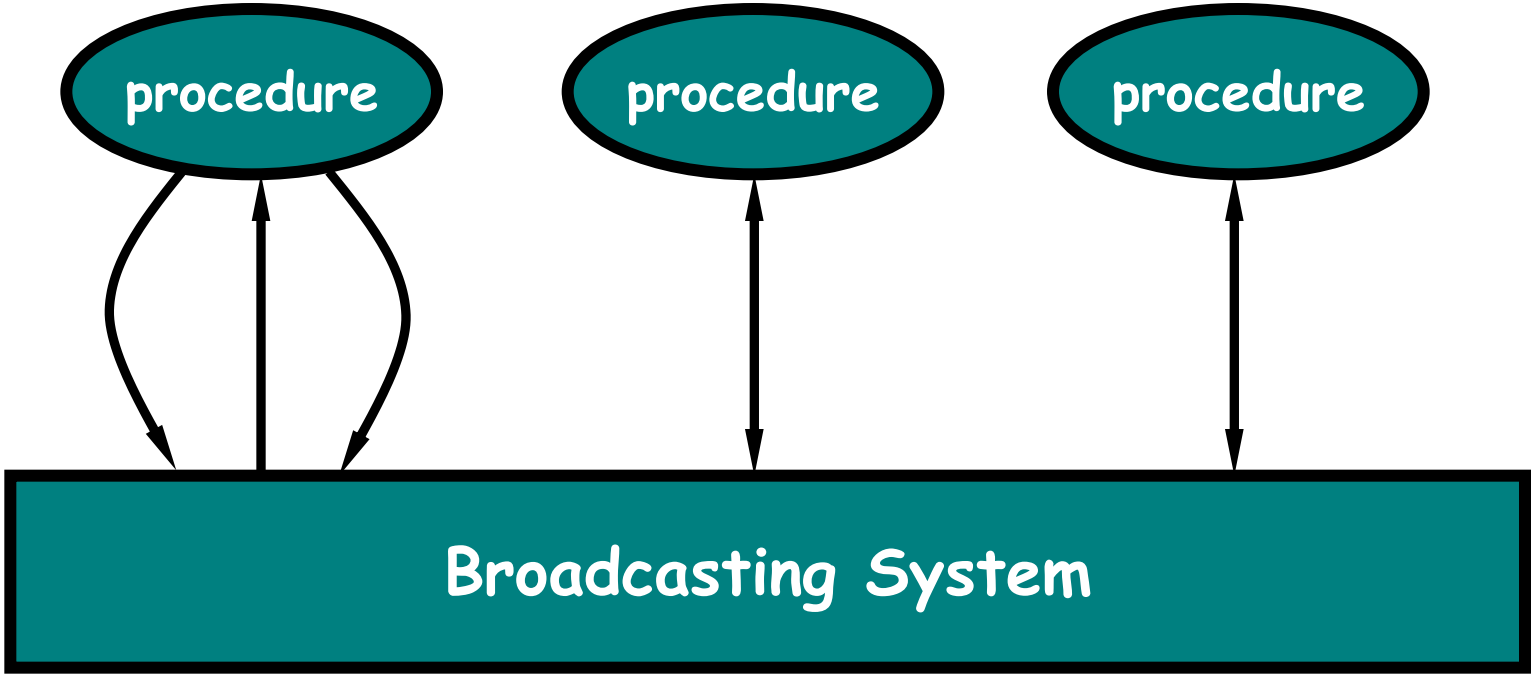
UML representation of Lunar Lander Example



QA evaluation for Object Oriented

- **Performance**
  - In distributed environment, may require middleware to access remote objects
- **Availability**
  - Distributed, if part of the system is impacted, the rest can function
- **Modifiability**
  - Easy to modify implementation without affecting other clients
  - Changing identity of objects may have high impact

Implicit Invocation Style



Implicit Invocation Variants

- **Publish-Subscribe**
  - Subscribers register to receive specific messages
  - Publishers maintain a subscription list and broadcast messages to subscribers
- **Event-Based**
  - ICs asynchronously emit and receive “events” communicated over event bus

Implicit Invocation Style

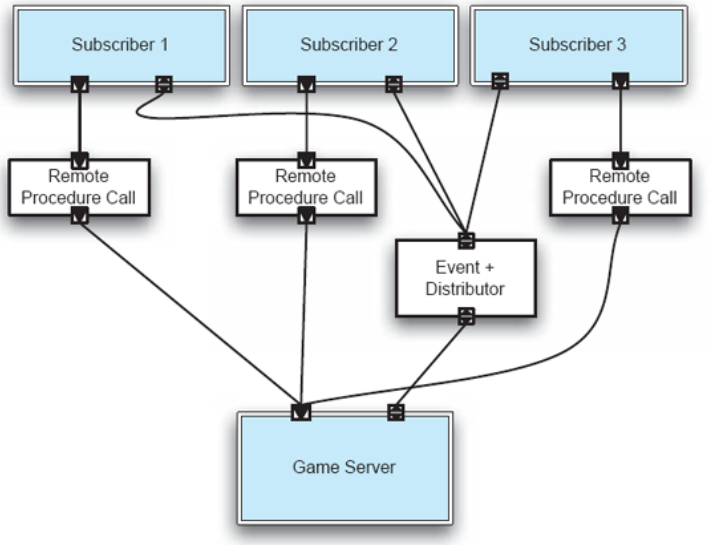
- **Components**
  - Publishers, subscribers
  - Event generators and consumers
- **Connectors**
  - (PS) Procedure calls
  - Event bus

Implicit Invocation Style Topology

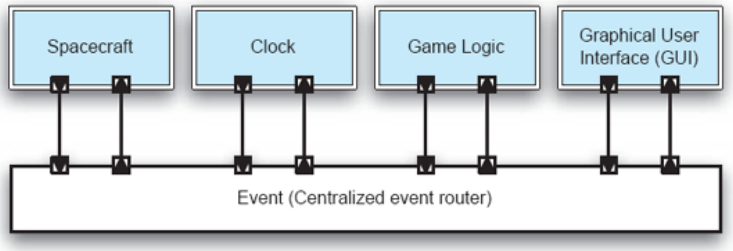
- Subscribers connect to publishers directly (or through network)
- Components communicate with the event bus, not directly to each other

Taylor et al. 2010

Implicit Invocation Style Topology



Publish-Subscribe



Event Based

Implicit Invocation Advantages

- (PS) Efficient dissemination of one-way information
- Provides strong support for **reuse**
  - Any component can be added, by registering/subscribing for events
- **Eases system evolution**
  - components may be replaced without affecting other components in the system

Implicit Invocation Disadvantages

- (PS) Need special protocols when number of subscribers is very large

- When a component announces an event:
  - it has no idea what other components will respond to it,
  - it cannot rely on the order in which the responses are invoked
  - it cannot know when responses are finished

Implicit Invocation Examples

- Used in **programming environments** to integrate tools:
  - Debugger stops at a breakpoint and makes that announcement
  - Editor scrolls to the appropriate source line and highlights it

- Twitter, Google+

QA evaluation for Implicit Invocation

- **Performance**
  - (PS) Can deliver 1000s of msgs
  - Event bus: how does it compare to Repository?
- **Availability**
  - Publisher needs to be replicated
- **Scalability**
  - Can support 1000s of users, growth in data size
- **Modifiability**
  - Easily add more subscribers, change in message format affects many subscribers