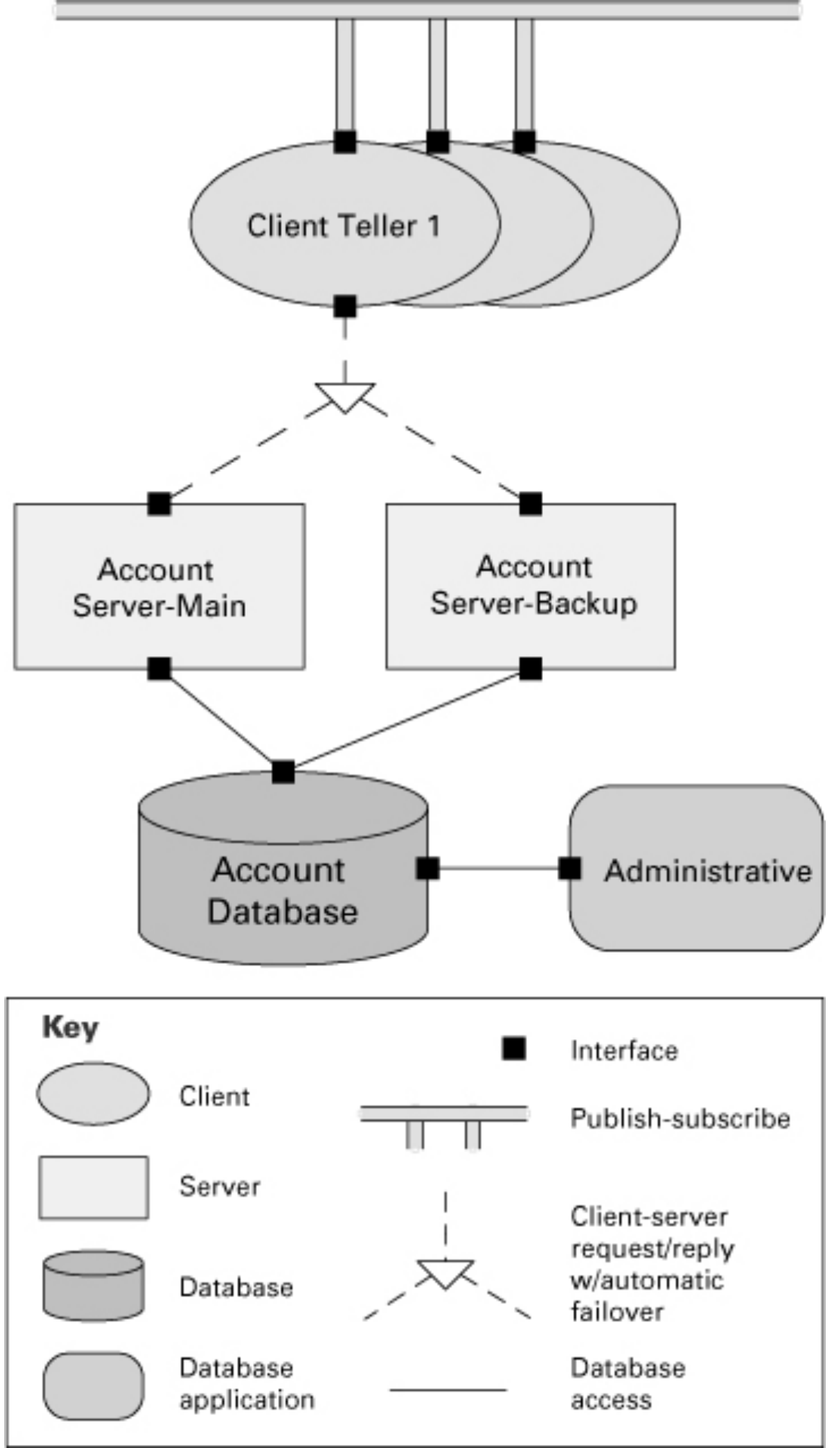
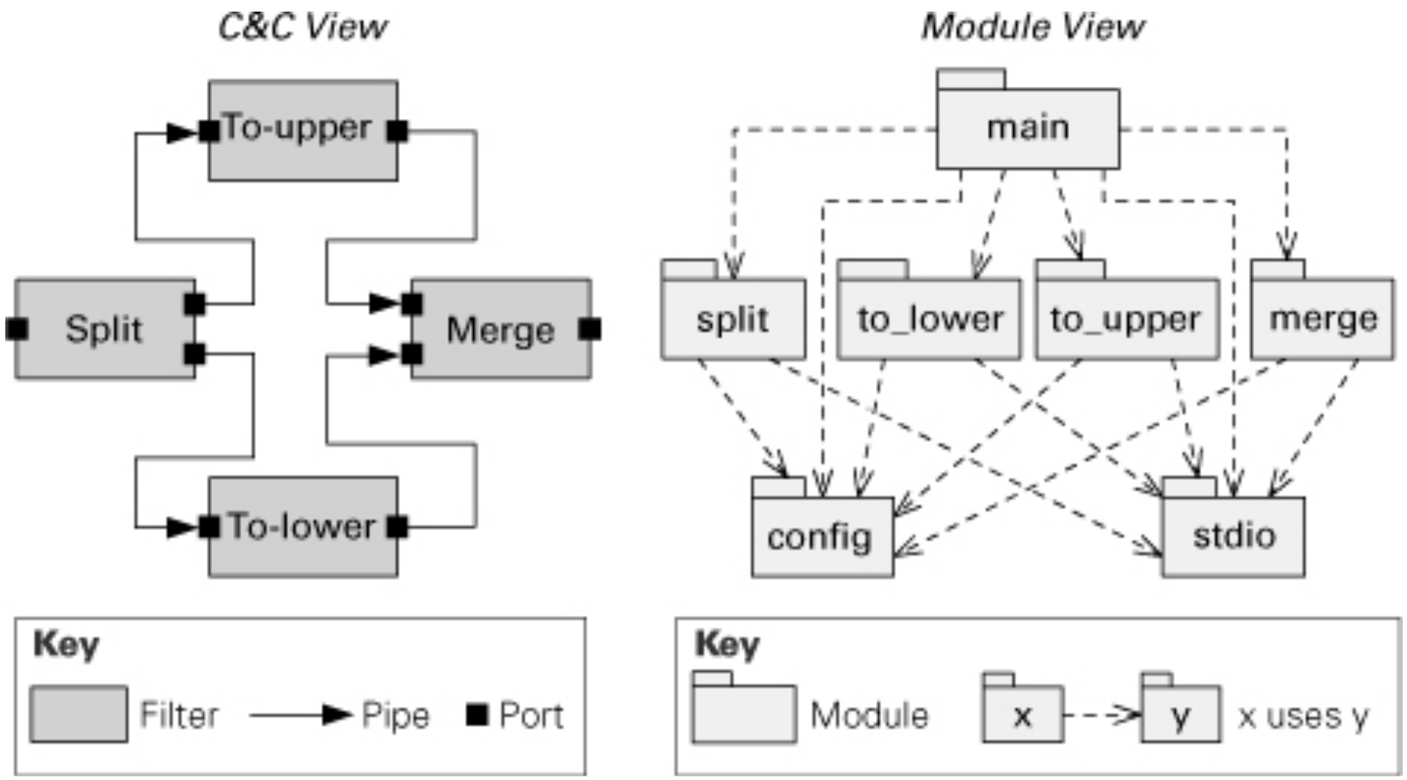
Example

- Several **clients** interact with two **servers** (one back up)
- servers access a shared **repository**
- clients communicate with each other using a **publish-subscribe** connector
- **administrative** component maintains shared repository



Relationship to other Views: **Module Views**

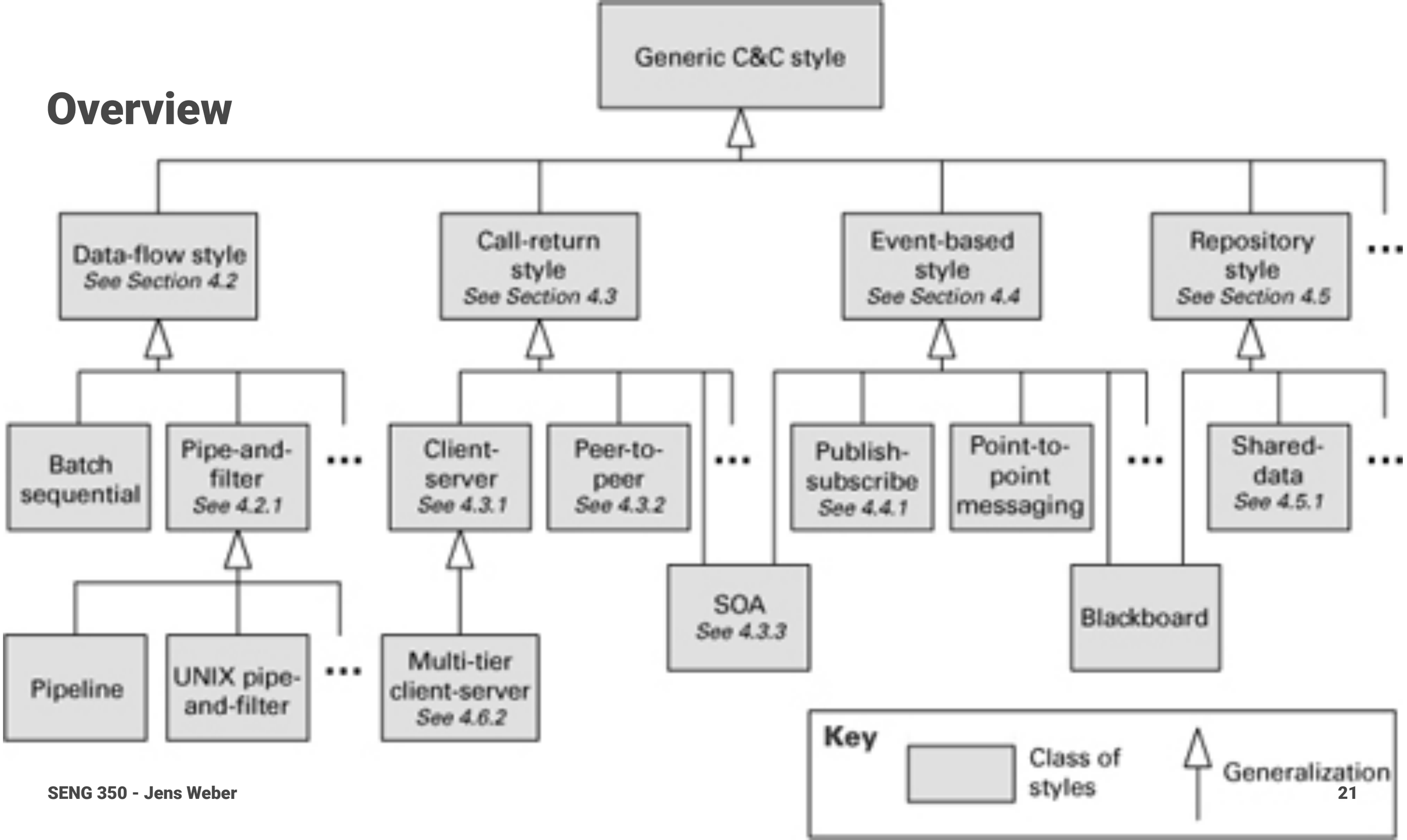
Since C&C views depict runtime elements and module views depict implementation units, these views may be very different.



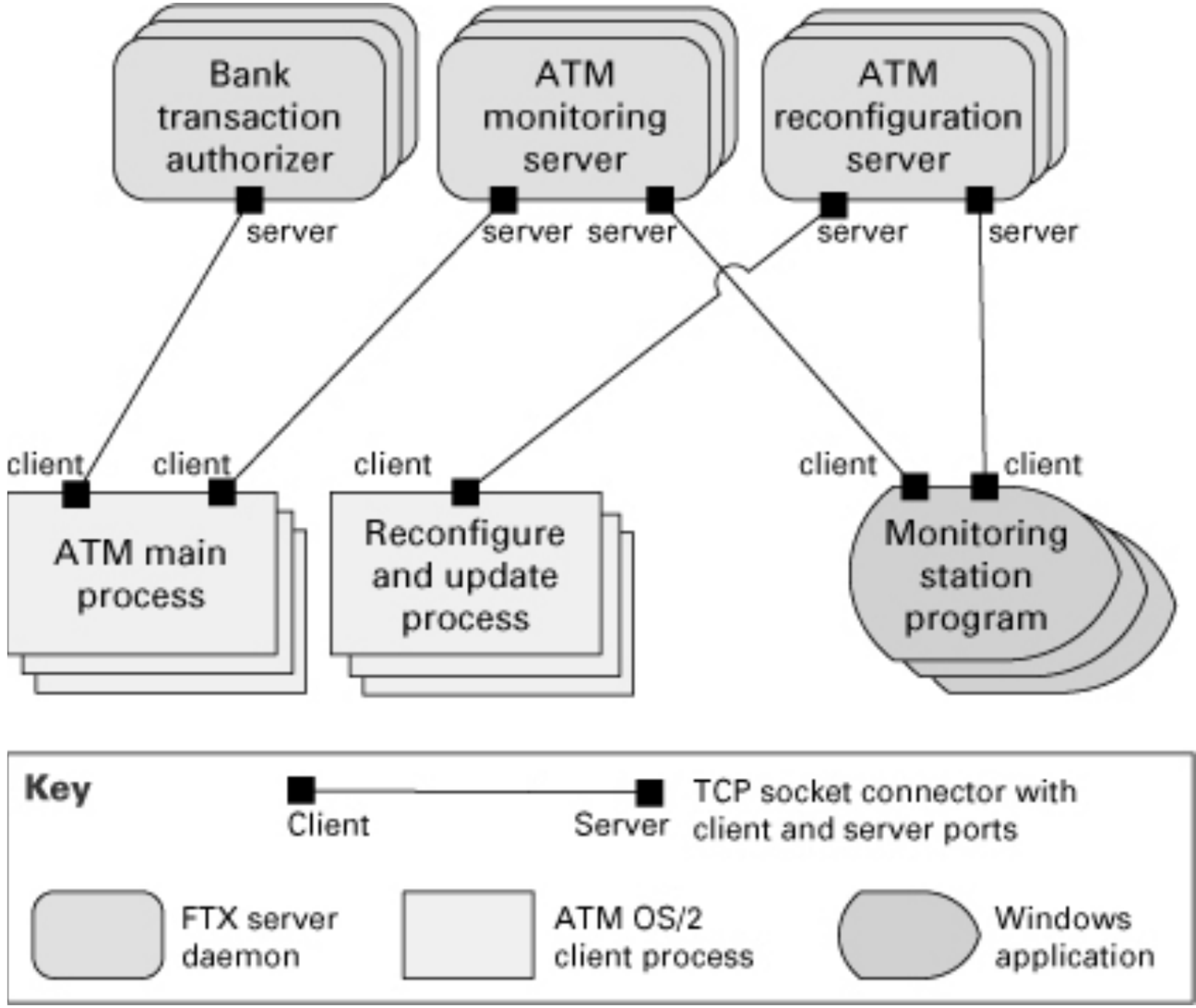
In such cases, a table may provide a mapping between runtime units and implementation units

C&C View	Module View
System as a whole	main
Split	split, config, stdio
To-lower	to_lower, config, stdio
To-upper	to_upper, config, stdio
Merge	merge, config, stdio
Each pipe	stdio

Overview


Generic C&C style
Data-flow style
See Section 4.2
Call-return style
See Section 4.3
Event-based style
See Section 4.4
Repository style
See Section 4.5
Batch sequential
Pipe-and-filter
See 4.2.1
Client-server
See 4.3.1
Peer-to-peer
See 4.3.2
SOA
See 4.3.3
Publish-subscribe
See 4.4.1
Point-to-point messaging
Blackboard
Shared-data
See 4.5.1
Pipeline
UNIX pipe-and-filter
Multi-tier client-server
See 4.6.2
Key
Class of styles
Generalization

Example: ATM banking system

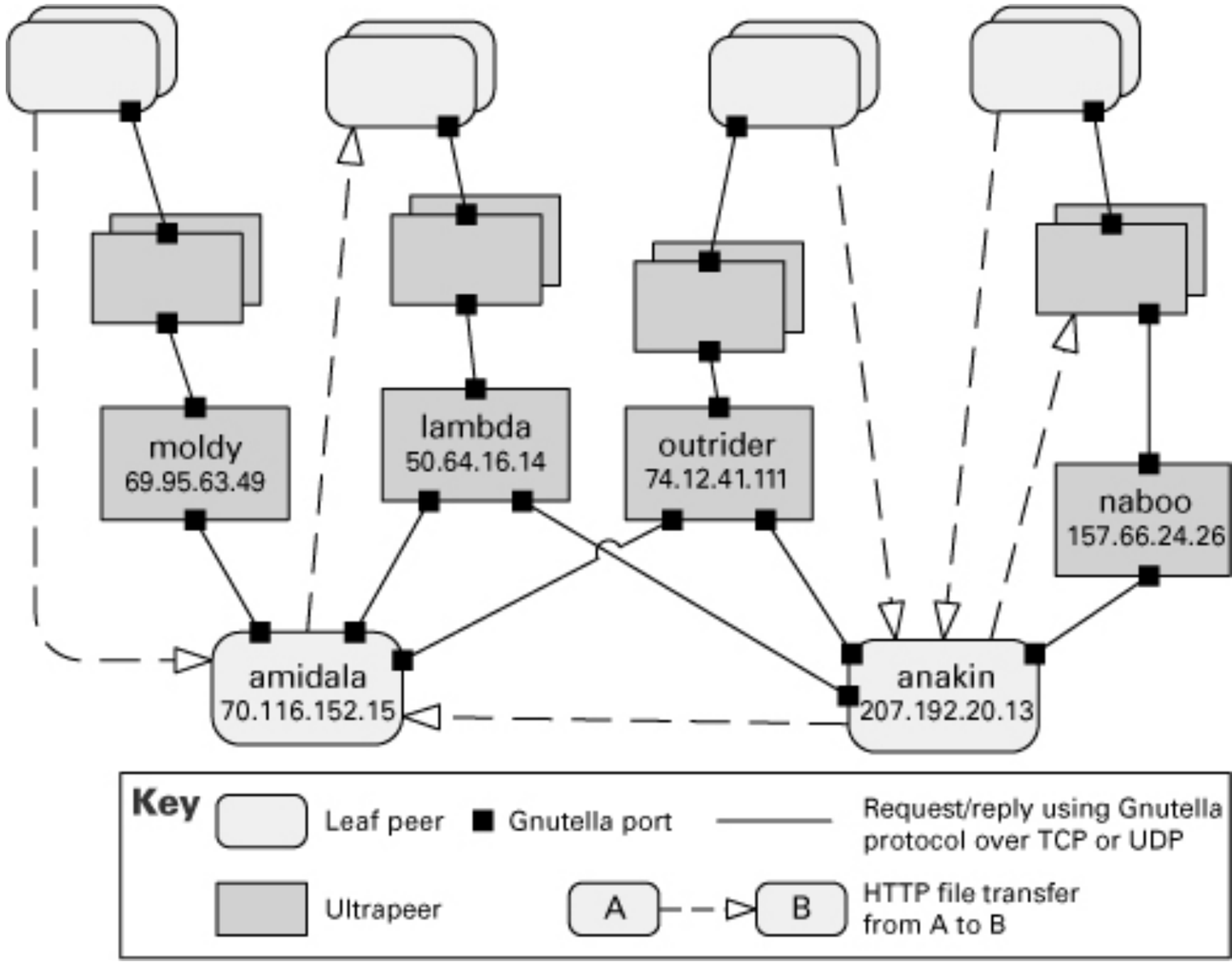


Peer-to-Peer Style

Elements	• *Peer* component • *Call-return connector*, which is used to connect to the peer network, search for other peers, and invoke services from other peers
Relations	The *attachment* relation associates peers with call-return connectors.
Computational Model	*Computation* is achieved by cooperating peers that request services of one another.
Properties	Same as other C&C views, with an emphasis on protocols of interaction and performance-oriented properties. Attachments may change at runtime.
Constraints	• Restrictions may be placed on the number of allowable attachments to any given port, or role. • Special peer components can provide routing, indexing, and peer search capability. • Specializations may impose visibility restrictions on which components can know about other components.
What It's For	• Providing enhanced availability • Providing enhanced scalability • Enabling highly distributed systems, such as file sharing, instant messaging, and desktop grid computing

Example: Gnutella

- Gnutella peer requests information from all of its connected peers, which respond with any information of interest.
- The connected peers also pass the request to their peers successively
- All the peers that have positive results for the search request reply directly to the requester
- The requester then establishes a connection directly with the peers that have the desired file and initiates the data transfer using HTTP (outside the Gnutella network).



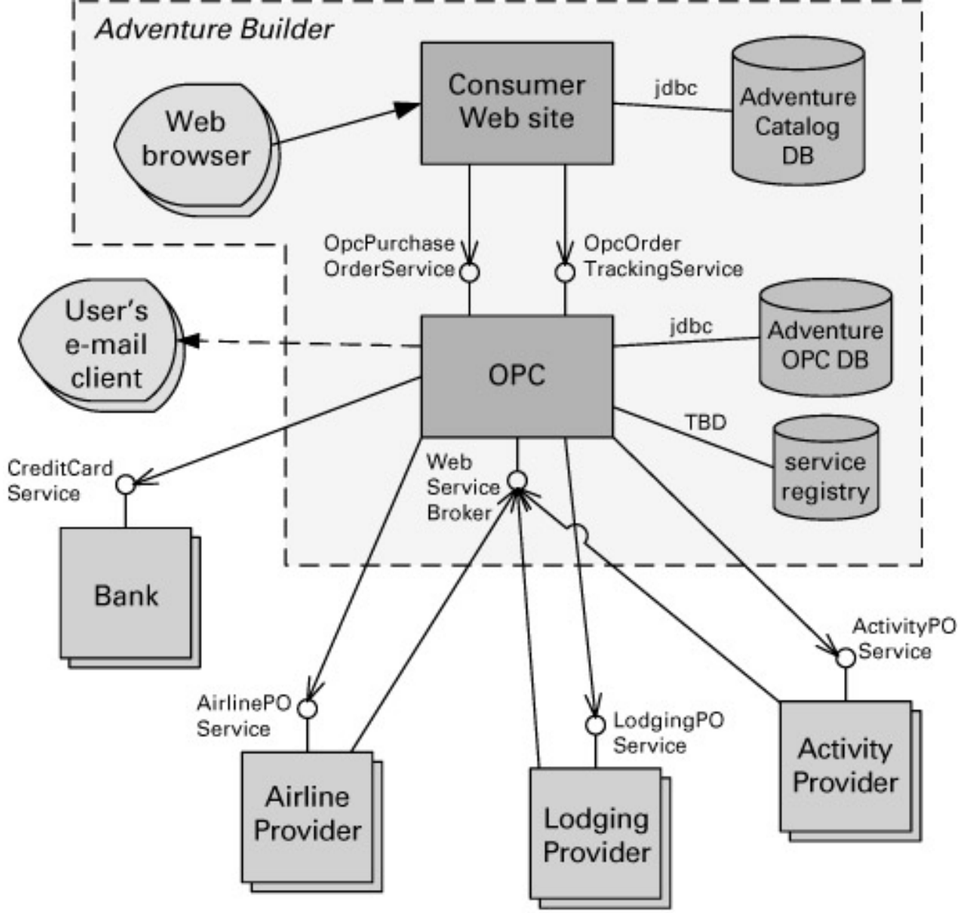
Service-Oriented Architecture Style

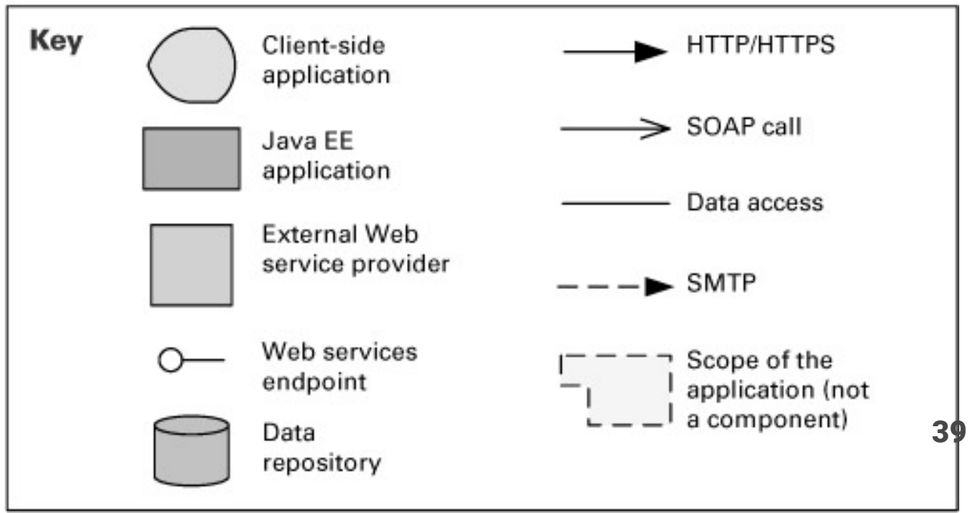
Elements

- *Service providers*, which provide one or more services through published interfaces. Properties will vary with the implementation technology (such as EJB or ASP.NET) but may include performance, authorization constraints, availability, and cost. In some cases these properties are specified in a service-level agreement (SLA).
- *Service consumers*, which invoke services directly or through an intermediary.
- *ESB*, which is an intermediary element that can route and transform messages between service providers and consumers.
- *Registry of services*, which may be used by providers to register their services and by consumers to query and discover services at runtime.
- *Orchestration server*, which coordinates the interactions between service consumers and providers based on scripts that define business workflows.
- *SOAP connector*, which uses the SOAP protocol for synchronous communication between Web services, typically over HTTP. Ports of components that use SOAP are often described in WSDL.
- *REST connector*, which relies on the basic request/reply operations of the HTTP protocol.
- *Messaging connector*, which uses a messaging system to offer point-to-point or publish-subscribe asynchronous message exchanges.

Example: Adventure Builder

- the OPC (Order Processing Center) is the "orchestration server"





Event-based Styles

Publish-Subscribe Style

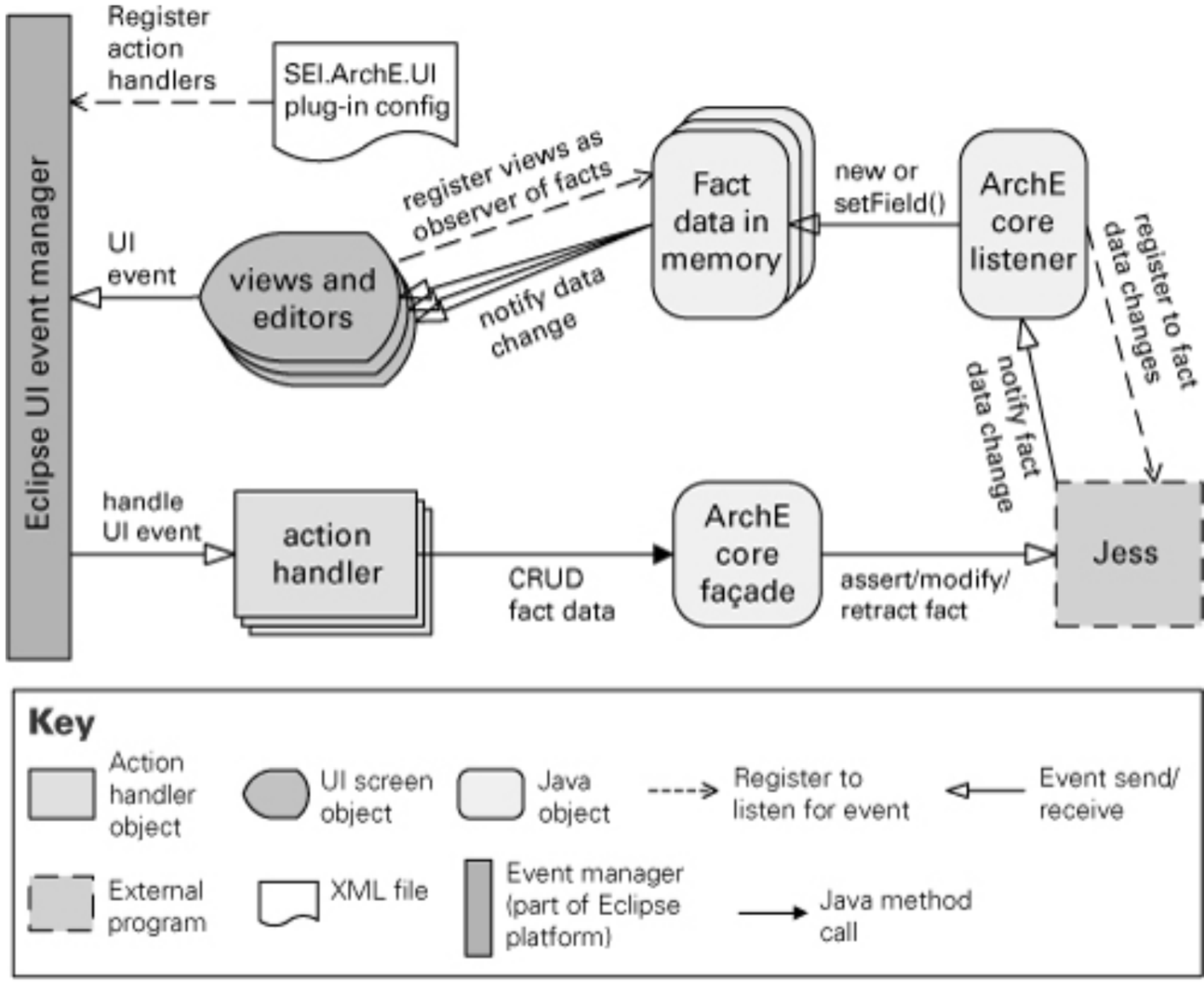
Elements	• Any C&C component with at least one publish or subscribe port. Properties vary, but they should include which events are announced and/or subscribed to, and the conditions under which an announcer is blocked. • *Publish-subscribe connector*, which will have announce and listen roles for components that wish to publish and/or subscribe to events.
Relations	*Attachment* relation associates components with the publish-subscribe connector by prescribing which components announce events and which components have registered to receive events.
Computational Model	Components subscribe to events. When an event is announced by a component, the connector dispatches the event to all subscribers.
Constraints	All components are connected to an event distributor that may be viewed as either a bus—that is, a connector—or a component. Publish ports are attached to announce roles, and subscribe ports are attached to listen roles. Constraints may restrict which components can listen to which events, whether a component can listen to its own events, and how many publish-subscribe connectors can exist within a system. A component may be both a publisher and a subscriber, by having ports of both types.
What It's For	• Sending events to unknown recipients, isolating event producers from event consumers • Providing core functionality for GUI frameworks, mailing lists, bulletin boards, and social networks

Examples systems using PubSub Style

- Graphical user interfaces, where a user's low-level input actions are treated as events that are routed to appropriate input handlers
- Applications based on the model-view-controller (MVC) pattern, where view components are notified when the state of a model object changes
- Extensible programming environments, in which tools are coordinated through events
- Mailing lists, where a set of subscribers can register interest in specific topics
- Social networks, where "friends" are notified when changes occur to a person's Web site

Example: ArchE tool

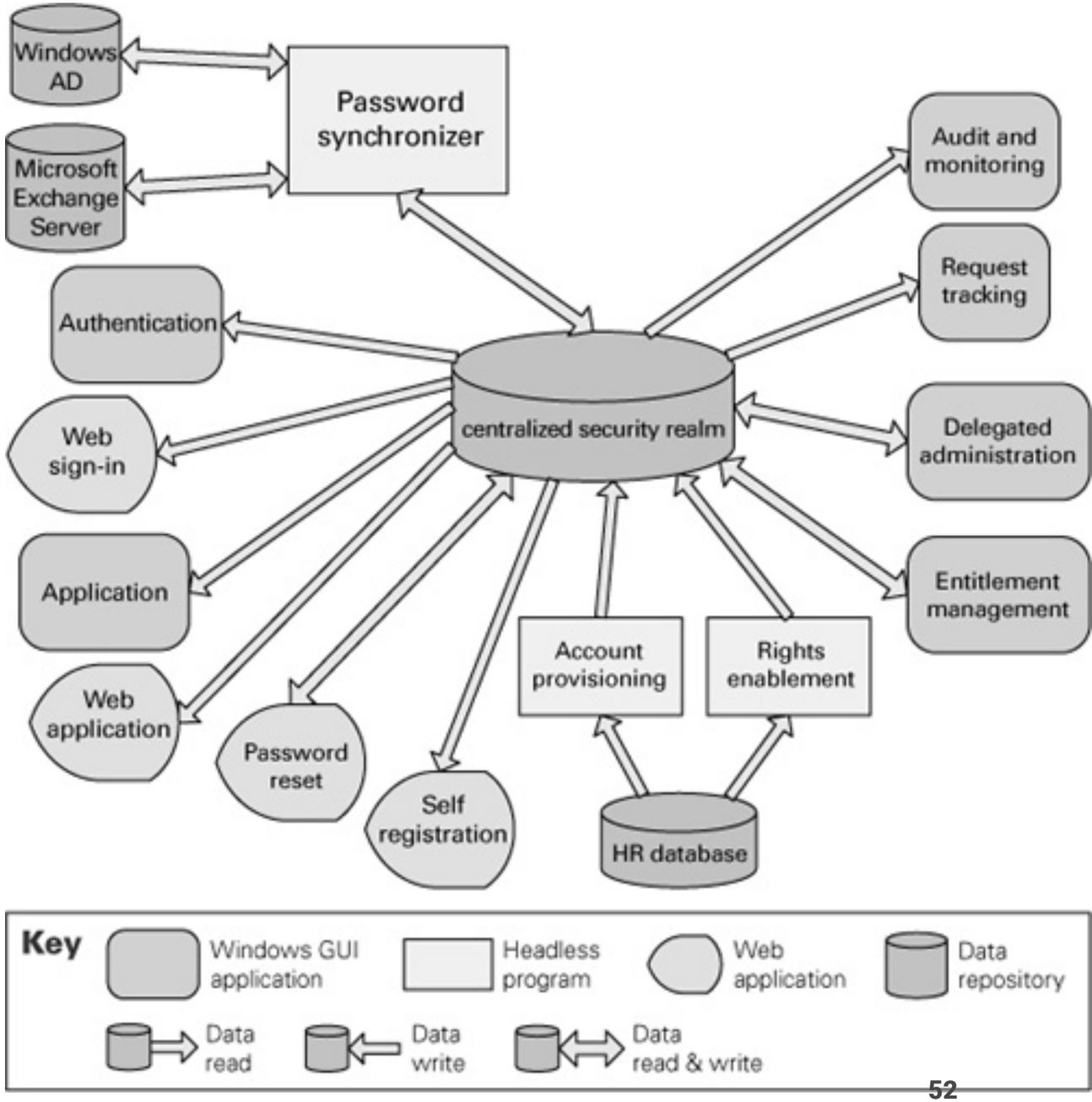
- `Eclipse Event Manager` acts as pub-sub connector
- `Fact data memory` consits of Java objects with "observable" interfaces
- `Jess` rule engine observes UI events and notifies `ArchE core listener` when facts need changing



Shared-Data Style

Elements	• *Repository component*. Properties include types of data stored, data performance-oriented properties, data distribution, number of accessors permitted. • *Data accessor component*. • *Data reading and writing connector*. An important property is whether the connector is transactional or not.
Relations	*Attachment* relation determines which data accessors are connected to which data repositories.
Computational Model	Communication between data accessors is mediated by a shared-data store. Control may be initiated by the data accessors or the data store. Data is made persistent by the data store.
Constraints	Data accessors interact with the data store(s).
What It's For	• Allowing multiple components to access persistent data • Providing enhanced modifiability by decoupling data producers from data consumers

Example: Enterprise access management system



Crosscutting issues for C&C Styles

- Some concerns relate to many C&C styles in a similar way
- Examples:
  - **Concurrency** (which C&C run in the same threads)
  - **Tiers** (groupings of components and connectors)
  - **Dynamic reconfiguration** (which components/connectors are created / configured / removed together)

Documenting these concerns *augments* existing C&C styles with additional detail.