Name: Ta Quang Hiep

ID: T090363

Class: K15T2

**Reading Question 5**

1. List all styles of C&C Views. Each team member will be assigned one different style and establish style guide for that style.
2. DFD (Data Flow Diagram) see similar-looking C&C views of architecture. What is difference?
3. Peer-to-Peer, client-server, and other call-and-return styles all involve interactions between producers and consumers of data or services. If an architect is not careful when using one of these styles, he or she will produce a C&C view that simply shows a request flowing in one direction and a response flowing in the other. What means are at the architect's disposal to distinguish among these three styles?

**Call-Return Styles**

**Client-Server Style**

1. **Overview**

As with all call-return styles, client-server style components interact by requesting services of other components. Requesters are termed clients, and service providers are termed servers, which provide a set of services through one or more of their ports. Some components may act as both clients and servers. There may be one central server or multiple distributed ones.

1. **Elements, Relations, and Properties**

**Element**

* **Client**: is a component that invokes services of a server component.
* **Server**: is a component that provides services to client components. Include information about the nature of the server ports (how many clients can connect)
* **Request/reply connector**: is used by a client to invoke services on a server.

**Relations:** The attachment relation associates client service-request ports with the request role of the connector and server service-reply ports with the reply role of the connector.

1. **Constraints:**

* Clients are connected to servers through request/reply connectors.
* Server components can be clients to other servers.
* Specializations may impose restrictions:
* Numbers of attachments to a given port
* Allowed relations among servers
* Components may be arranged in tiers

1. **What It’s For**

* Promoting modifiability and reuse by factoring out common services
* Improving scalability and availability in case server replication is in place
* Analyzing dependability, security, and throughput

1. **Notations**
2. **Relation to Other Styles**

* The client-server style decouples producers of services and data from consumers of those services and data
* Other styles, such as peer-to-peer do not have the asymmetric relationship between clients and servers

1. **Examples of the Client-Server Style**

The World Wide Web may be the best known example of Client-Server Style.Client-server system allows clients (Web browsers) to access information from servers distributed across the Internet.

**Peer-to-Peer Style**

1. **Overview**

* Components directly interact as peers by exchanging services
* Any component can, in principle, interact with any other component by requesting its services.
* Each peer component provides and consumes similar services, and sometimes all peers are instances of the same component type.

1. **Elements, Relations, and Properties**

**Element:**

* **Peer component**
* **Call-return connector:** 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.

**Properties**: Emphasis on protocols of interaction and performance-oriented properties. Attachments may change at runtime.

1. **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

1. **What It’s For**

* Providing enhanced availability
* Providing enhanced scalability
* Enabling highly distributed systems, such as file sharing, instant messaging, and desktop grid computing

1. **Notations**
2. **Relation to Other Styles**

The absence of hierarchy means that peer-to-peer systems have a more general topology than client-server systems.

1. **Examples of the Peer-to-Peer Style**

A bidirectional file transfers like torrent file-sharing networks where single computers (peers) sharing file together.

**Service-Oriented Architecture Style**

1. **Overview**

* Service-oriented architectures consist of a collection of distributed components that provide and/or consume services.
* Service provider components and service consumer components can use different implementation languages and platforms.
* Service providers and service consumers are usually deployed independently, and often belong to different systems or even different organizations.

1. **Elements, Relations, and Properties**

**Element:**

* **Service providers:** 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:** invoke services directly or through an intermediary.
* **ESB (enterprise service bus)**: is an intermediary element that can route and transform messages between service providers and consumers.
* **Registry of services:** may be used by providers to register their services and by consumers to query and discover services at runtime.
* **Orchestration server:** coordinates the interactions between service consumers and providers based on scripts that define business workflows.
* **SOAP connector:** 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:** relies on the basic request/reply operations of the HTTP protocol.
* **Messaging connector:** uses a messaging system to offer point-to-point or publish-subscribe asynchronous message exchanges.

**Relations:** Attachment of the different kinds of ports available to the respective connectors

1. **Constraints:**

* Service consumers are connected to service providers, but intermediary components (such as ESB, registry, or BPEL server) may be used.
* ESBs lead to a hub-and-spoke topology.
* Service providers may also be service consumers.
* Specific SOA patterns impose additional constraints.

1. **What It’s For**

* Allowing interoperability of distributed components running on different platforms or across the Internet
* Integrating legacy systems
* Allowing dynamic reconfiguration

1. **Notations**
2. **Relation to Other Styles**
3. **Examples of the Service-Oriented Architecture Style**

**Event-Based Styles**

1. **Overview**

Components may sub scribe to a set of events. It is the job of the publish-subscribe runtime infrastructure to make sure that each published event is delivered to all subscribers of that event. Thus the main form of connector in this style is a kind of event bus. Components place events on the bus by announcing them; the connector then delivers those events to the components that have registered an interest in those events.

1. **Elements, Relations, and Computational Model**

**Element**

Any C&C components 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.

1. **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

1. **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

1. **Notations**
2. **Relation to Other Styles**

The publish-subscribe style is similar to a blackboard repository style, because in both styles components are automatically triggered by changes to some component. However, in a black-board system, the database is the only component that generates such events; in a publish-subscribe system, any component may generate events.

1. **Examples of the Client-Server Style**

Eclipse UI event manager acts as an event bus for user-interface events (such as button clicks). Subscription information—that is, what UI events are relevant to the system and what components handle them—is defined at load time when the event manager reads the SEI.ArchE.UI plug-in config XML file. From then on a UI event generated by the user working on a view or editor is dispatched via implicit invocation to the action handler objects that subscribe to that event.