Lecture 3 Software Architecture Model

- Evolution of S/W Architecture
- S/W Architecture Concept
- Key Attributes

软件架构模型

Two Types of Modeling Methodology:

Practical Approach vs. Academic Approach

- **Practical Approach (PA)**，uses a set of standard modeling symbols to present the architectural model and the real architectural design
- **Academic Approach (AA)**，uses the Architecture Description Language (ADL) to conduct a formal research and logic analysis to software architecture

软件架构模型

PA work: diagrams, in-component connection language, component-based system description and UML

- Diagram presentation， use rectangular symbols to represent the processes, module and subsystems, and directed lines to describe their relationships --- Line-Block Diagram
- In-component connection language，a programming language-like symbol set to connect modules programmed in different languages --- too dependent to the language and limited in architectural description

软件架构模型

- Component-based system description normally is used in specific application domains, not appropriate for general architectural designs
- UML is a widely accepted architectural modeling approach, which can be used to map Kruchten's "4+1" model to the UML diagrams, in which class diagram for logic view, activity diagram for process view, deployment diagram for physical view, component diagram for development view

软件架构模型

AA work: prompt to use Architectural Description Language (ADL) to analyze the structure and behavior of s/w arch

- ADL is a formal approach to s/w architecture, which provides a better abstraction and logic presentation to the system. ADL includes the following keys:
- Port: defines the attributes, services, messages and operations of components, through which the external world can communicate to.

软件架构模型

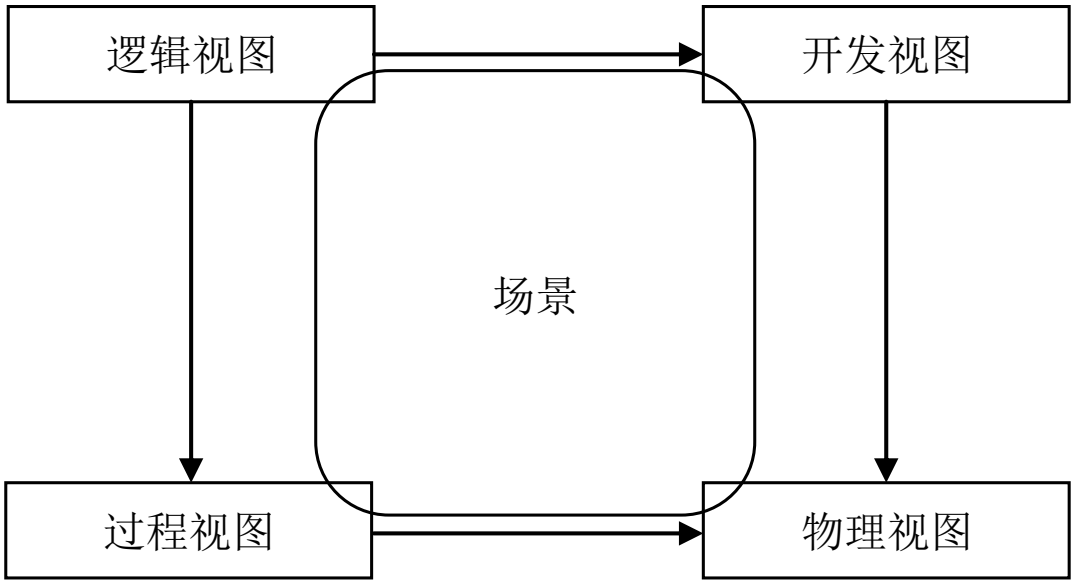
- Type： abstraction of reused components. Similar to multiple instances of one class, there may be multiple uses (realization of design) of one component type.
- Semantics： advanced description to component's behaviors， which can be used for analysis to the architecture, measurement on constraints and integrity of components in one system
- Constraints： the conditions applied to the system or its components, to which a break or failure will cause the system failure

软件架构模型

A Practical Approach – “4+1” View Model

最终用户：功能需求

开发人员：软件管理



系统集成人员：并发性、分布性和容错性等

系统工程人员：系统拓扑结构、安装和通信等

软件架构模型

“4+1” Views

End user

Programmers & software managers

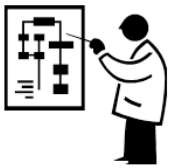
Logical View

Development View

Scenarios

Process View

Physical View



Integrator

System Engineer

软件架构模型

- **Logic View**， also Concept View， describes the functional requirements on the system, i.e. the functions and services that will be provided to end users upon delivery
- **Development View**， also Component View， mainly describes the layout and relationship among architectural units --- a static view on structure
- **Process View**， description on system dynamic behavior, i.e. the reliability, throughput, concurrency, distribution and fault tolerance.

软件架构模型

- **Physical View**， describes how to map s/w units to h/w nodes and shows physical environment, h/w configuration and distribution, under considerations on system performance, scale and fault tolerance
- **Scenario**， an abstraction of user requirement and system function, by which the designer can analyze and design s/w architecture through investigating the constraints under each scenario

In summary, Logic View defines the targets for system development, Development View and Process View provide the technical solution, Physical View presents the system topology, while Scenario addresses the boundary for above tasks.

软件架构模型

UML Model consists of multiple diagrams, describing the system from various angels

- Use Case Diagram: describes the system functions in the view of users and defines business requirements
- Class Diagram: describes the static structure of classes and their relationships
- Package Diagram: describes the layered structure of system by capturing its logic organization
- Activity Diagram: describes the system activities that are required to provide the functions and the conditions on these activities

软件架构模型

- Sequence Diagram: describes the dynamic interaction relationship among the objects by showing message passing order and inter-class relationship
- State Diagram: describes the states and state shift conditions for a group of objects
- Component Diagram: describes the physical existence of s/w codes and their functional interchange connections
- Deployment Diagram: describes the projection and mapping between s/w units and h/w platforms

软件架构模型

Three Approaches of Using UML in Arch Modeling:

- Using UML as one type of ADL to model the architectural without any adjustment
- Using the extension of UML to add constraints to the UML model to meet the requirements of architectural modeling
- Extend the UML model by adding more arch modeling elements to the meta model of UML, so it can be used as a full-feature ADL

Mapping of UML elements to architectural model attributes

- UML’s class, component, node, package and subsystem → Component
- UML’s relationship and certain type of class → Connector
- UML’s port → Port
- UML’s regulation → Constraint
- UML’s component diagram, package diagram and deployment diagram can be used to present Deployment

软件架构模型

- UML’s Use Case Diagram → Logic View
- UML’s class diagram, component diagram → Development View
- UML’s state diagram, sequence diagram and activity diagram → Process View
- UML’s deployment diagram describes the distribution of function units and the storage topology for data → Physical View

Advantages of UML in Modeling Arch

- UML is a widely used OO designing language and a de facto standard, and it has a well designed extendibility
- By using formal definitions, UML becomes a semi-formal modeling language and is independent to programming language or development process
- UML provides multiple views for software analysis, design and development
- UML provides a rich set of modeling elements and symbols that can be used as a unified modeling language for software development

软件架构模型

Drawbacks of "4+1" View Model as ADL:

- "4+1" View Model is incapable of showing various layers in abstraction of architecture and describing architectural patterns
- Data, as a major unit of s/w system, are not fully modeled in the "4+1" View Model
- It doesn't provide sufficient description on architectural relationship and constraints
- In developing s/w systems, it lacks of enough guidelines in designing and evaluating the systems

架构建模语言（ADL）

4.1 软件体系结构描述方法

- 表4-1 当前常见的一些体系结构描述语言

ADL	研发组织	负责人
ACME	Carnegie Mellon大学	David Garlan
Wright	Carnegie Mellon大学	David Garlan
C2	Southern California大学	Medvidovic
Unicon	Carnegie Mellon大学	Mary Shaw
Darwin	英国Imperial College	Jeff Kramer & Jeff Magee
AESOP	Carnegie Mellon大学	David Garlan
Rapide	Stanford大学	David Luckham
Control&MetaH	Honeywell公司技术中心	Steve Vestal
Weaves	美国Aerospace公司	Gorlick
SADL	SRI	Mark Moriconi
UML	Rational 软件公司	
Gestalt	Siemens 公司研究院	Bob Schwanke
Demeter	Northeastern 大学	Karl Lieberherr
FR	Ohio 州立大学	B.Chandrasekaran

架构建模语言（ADL）

ADL Type S/W Arch Modeling Language

ACME、Unicon、Wright、Darwin、Aesop、SADL、MetaH、Rapide和C2

ACME

- ✓ Core features: component, connector, system, port, role, description, presentation diagram,
- ✓ Also an arch model conversion language by which one ADL model can be converted to another model

架构建模语言（ADL）

ACME scripts sample

```
System simple_cs={
    Component client={Port sendRequest}
    Component server={Port receiveRequest}
    Connector rpc={Roles{caller, callee}}
    Attachments={
        client.sendRequest to rpc.caller;
        server.receiveRequest to rpc.callee;
    }
}
```

架构建模语言（ADL）

ACME scripts sample

Connector HTTP_Connector : HTTP_Connector_Type = **new** HTTP_Connector_Type **extended with** {.....}

Connector SQL_Connector : SQL_Connector_Type = **new** SQL_Connector_Type **extended with** {.....}

Attachment Browser.Send_Request **to** HTTP_Connector.HTTP_Caller;

Attachment Web_Server.Receive_Request **to** HTTP_Connector.HTTP_Responder;

Attachment DataBase_Server.Receive_Request **to** SQL_Connector.SQL_Responder;

}

架构建模语言（ADL）

```
Connector HTTP_Connector : HTTP_Connector_Type = new
HTTP_Connector_Type extended with {
   Role HTTP_Caller : HTTP_Caller_Role_Type = new HTTP_Caller_Role_Type
extended with {
      rule exactly1Attachment = heuristic size(self.ATTACHEDPORTS) == 1;
      rule attachedOnlyToRequestPort = invariant forall p : Port in
self.ATTACHEDPORTS |
         declaresType(p, RequestPortT);
   }
   Role HTTP_Responder : HTTP_Responder_Role_Type = new
HTTP_Responder_Role_Type extended with {
      rule exactly1Attachment = heuristic size(self.ATTACHEDPORTS) == 1;
      rule attachedOnlyToResponsePort = invariant forall p : Port in
self.ATTACHEDPORTS |
         declaresType(p, ResponsePortT);
   }
……
}
```

架构建模语言（ADL）

Unicon

- ✓ Modeling is built based on two core elements: component and connector
- ✓ In Unicon, Component represents system's computing unit and data storage unit, in which it separates computing from data storage and defines a complete set of semantics and action for each independent unit
- ✓ Connector is defined as a class for component interaction, providing a bridge between two independent components

架构建模语言（ADL）

Unicon’s component definition

<Component>:=Component<identifier>
    <interface>
    <component_implementation>
    end<identifier>

Unicon’s connector definition

<Connector>:=Connector<identifier>
    <protocol>
    <connector_implementation>
    end<identifier>

架构建模语言（ADL）

Wright

- Wright defines the connector as a set of protocols that represent each party and its role in the interaction relationship
- Wright provides an explicit specification on connectors and a definition to complex connection
- Wright provides the instance of component and connector abstraction and the attachment between the port and role, by which the system configuration is presented

架构建模语言（ADL）

In Wright, Architecture Presented in Three Sections:

- Section I: definition of component and connector types by the port-spec and component-spec, and each port-spec regulates a component's logic interaction with its eternal environment
- Section II: sets of instances of component and connector
- Section III: description of assembly of component and connector by association of a port to a role of connector

架构建模语言（ADL）

Wright's Presentation of Client/Server Architecture

```
System SimpleExample
    Component Server=
        port provide[provide protocol]
        spec [Server specification]
    Component Client=
        port request[request protocol]
        spec [Client specification]
    Connector C-S-connector=
        role client[client protocol]
        role server[server protocol]
    Instances
        s:Server
        c:Client
        cs:C-S-connector
    Attachments
        s.provide as cs.server
        c.request as cs.client
end SimpleExample
```

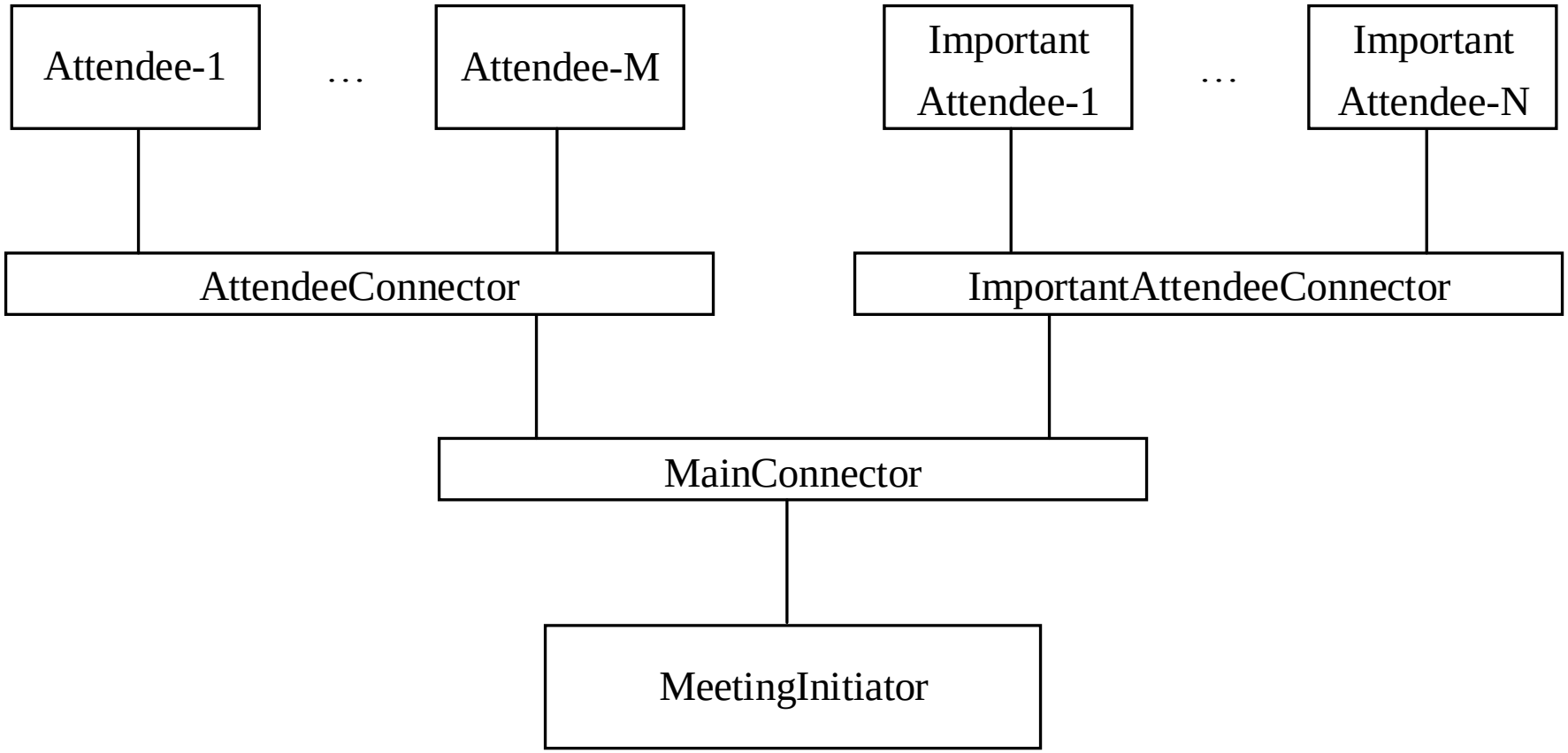
架构建模语言（ADL）

C2

- Connector is responsible for message passing between components, through its top or bottom ports
- Component cannot directly exchange messages between themselves, but only through connectors
- Each port of component can only connect to one connector, yet a connector may connect to any number of componets

架构建模语言（ADL）

An Arch Model of Meeting System in C2



架构建模语言（ADL）

Pros of ADL

Can provide an effective formal modeling tool for software architecture and support performance assessment and optimization of architectural design

Cons of ADL

- ✓ NO unified formal description specification and standard tool for software development
- ✓ Compared to UML, not easy to learn and use for developers
- ✓ Each ADL may fit a particular use, but none is accepted as a full-featured modeling tool

End of Lecture

Thanks!