Lecture 2 Software Architecture Model

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

一、Evolution History

- Early days: a series of execution codes that use computing logics and algorithms from the program start to the end
- Then people found, to the large-scale s/w systems, the architectural design is more important than data structure and algorithm
- Experienced programmers tend to use certain types of architectural designs, yet no models or design standards that can follow

软件架构模型

- S/W architectural design originated from software engineering and borrowed a lot concepts from computer systems， and become a separated research field
- Topics include architectural description, architectural style, architectural design evaluation and formal approach to s/w architectural design
- Targets: Reuse of architectural design, design optimization, and s/w performance assessment

软件架构模型

1、No-architecture Stage（1946～1969）

- Programming languages: machine code and assembly language, emerge of advanced languages such as FORTRAN and COBOL
- Programs were small and simple, without considerations on s/w architecture
- Hardware system was simple, too, and s/w systems are mainly centralized
- S/W was bonded to h/w platform without importability

2、Infant Stage (1970～1980)

- Majority were advanced programming languages
- Software Engineering was initiated and researched, and development models, methodology and SDKs were developed
- The preliminary design and detailed design were conducted to guide building a s/w system, based on the schemes of control flow and data flow

3、Junior Stage(1980～1990)

- Appearance of large-scale s/w systems and the object-oriented technology, which allowed the designer, developer and user to more precisely present their design work, which greatly improved the quality of s/w products
- In 1994, Booch、Rumbaugh、Jacobson invented the Unified Modeling Language (UML)
- S/W architectural design draw attentions

软件架构模型

4、Advanced Stage (since 1990)

- S/W systems become more and more complex and larger scale
- Algorithm + Data Structure converted to Component + Architectural Assembly
- Advanced abstraction: decouple Arch. Model from Arch. Design (Kruchten's "4+1" Model)
- No unified definition, no widely accepted standards yet

软件架构模型

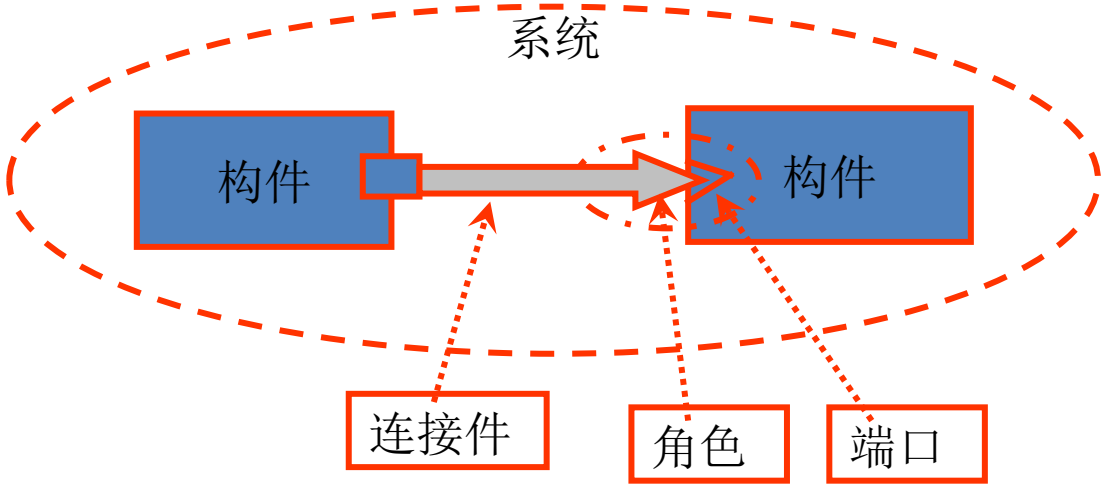
S/W Arch Model – the formal description of software architecture

- S/W Architecture: can be rebuilt and pre-made software framework
- **Component:** can be reused and pre-made s/w units, work as the basic computing units or data storage blocks for software systems
- **Connector:** also a reused and pre-made s/w unit that can provide connection between components
- **Constraint**: the relationship between the components and connectors

软件架构模型

Architectural Model Core consists of

- Components
- Connectors
- Constraints



软件架构模型

- Component: as an encapsulated s/w instance, with the interface interacted with external systems.

  Component interface consists of a set of ports, serving as the interacting points with others

软件架构模型

- Connector: as one key block of software architecture, it also has the interface.

  Connector interface consists of a set of Roles that defines the parties involved in the interaction of the connection.

软件架构模型

Formal presentation

- S/W Architecture ::= Core Model | Arch Style

  Core Model ::= (Component, Connector, Constraints)

  Component ::= {port1, port2, …, portN}

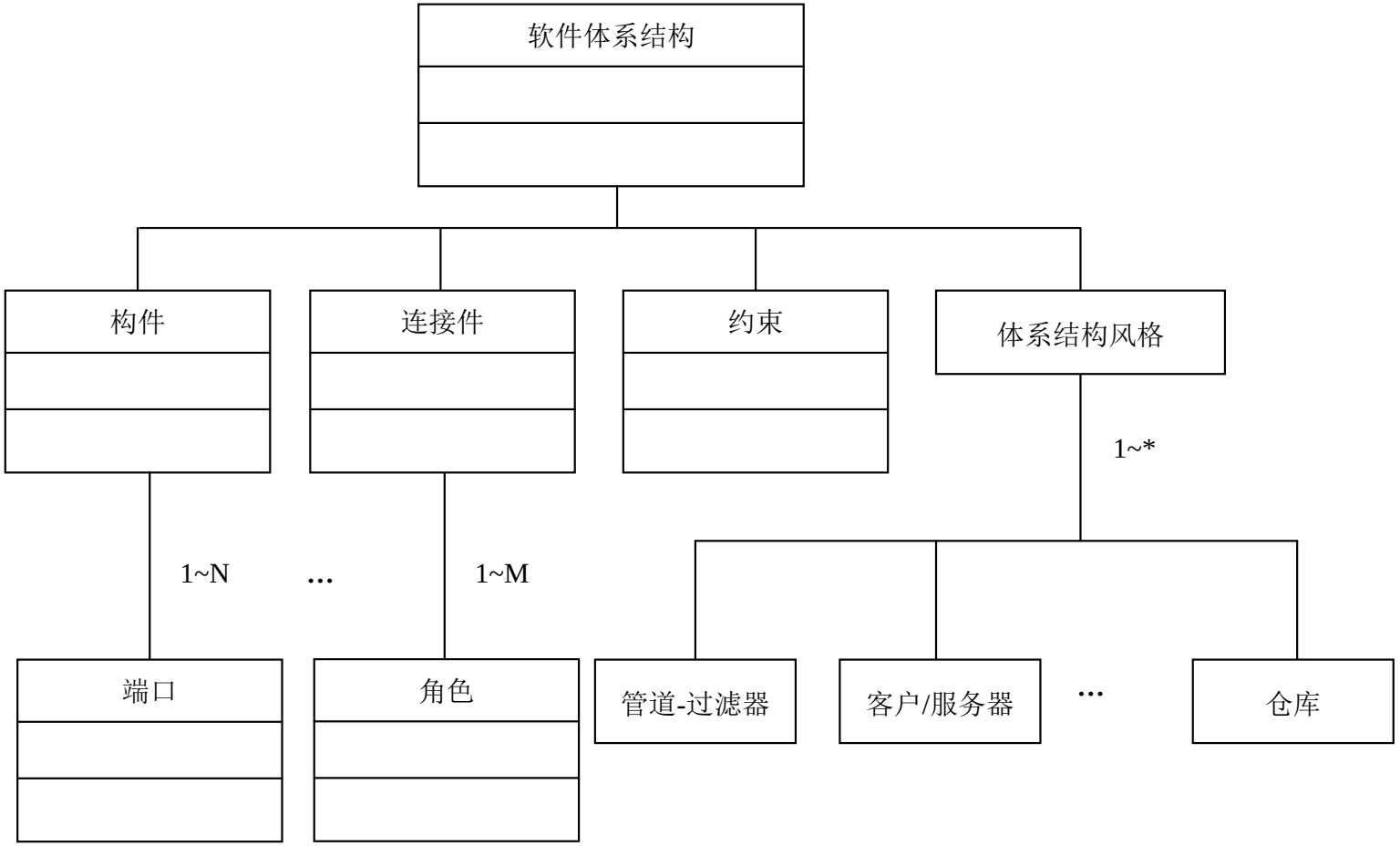
  Connector ::= {role1, role2, …, roleM}

  Constraint ::= {(porti, portj), …}

  Arch Style ::= {pipe-filter, client/server, DB, …}

软件架构模型

A Formal Presentation of S/W Arch in Graph



Arch Attribute Details - Component

- Can be as small as a process/thread, and as large as an application, can be a function, an object, a process, a binary instance, a class or a data block
- Independent to each other, with inside details encapsulated and service provided through well defined interface
- Mechanism contains the attributes of ports, type, semantics, constraints, evolution and non-functional properties

软件架构模型

- Principles for Component Composition
  - ✓ Components joining the composition keep their independence， which leads to the composite component having advantages of better re-use and better evolution behavior
  - ✓ Composition shall be realized through using of external units, such as connectors. Composition contains the interaction among components, not just a list of components
  - ✓ Component Decoupling – an opposite operation, will explain later

软件架构模型

Connector’s Formal Definition:

（*ID, Role, Beha, Msgs, Cons, Non-func*）

ID: connector’s identifier

Role: a set of roles

Beha: a set of behavior definition

Msgs: a set of messages associated to connector

Cons: a set of constraint conditions, including initialization, pre-definition and post-constraint

Non-func: non-functional description of connector

Arch Attribute Details - Connector

- Used as the supporting blocks for component interaction and role realization
- Can be in the form of message/signal passing, function/method calling, data transfer/conversion, as well as process sync.
- Examples: pipe in Pipe-Filter, communication protocols in Client/Server, and SQL connection between the application program and database
- Interface of Connector: a set of Roles that serve as the interaction points and define the role and position of participating parties

软件架构模型

Key Design Attributes of Connector： extendibility, interoperation, dynamics and responsiveness

- Extendibility： allows the dynamic changes on the associated components and their interactions
- Interoperation： the connected components may directly or indirectly operate on other components through the connectors
- Dynamics： the connector may apply dynamic actions to the connected components
- Responsiveness: Concurrent and timing response by the connectors

Component vs. Connector

- To components, the connectors serve as a bond to support the interaction among components
- Similar to components, connectors are also building blocks of the software architecture, yet provide different functions or services
- Constraints describe and define the relationship and applying roles between components and connectors. Can be presented in a deployment scheme or a topology

Arch Attribute Details - Constraint

- Described as the rules and conditions
- Provides a boundary for architectural connection, in which connecting requirements for components, ports, communications are specified, and the behaviors are defined
- Bridges the software architecture to the system requirements through well defined and correctly designed connecting components

软件架构模型

Presentation of Architectural Model

Component Diagram, Connection Diagram, and Assignment Diagram

- **Component Diagram** describes the functions of each component and the relationship among them, including the division diagram, utility diagram, layered diagram, and class diagram
- **Assignment Diagram** shows the assignment of each s/w units in the building environment or execution environment, including the deployment diagram, realization diagram, and task assignment diagram

软件架构模型

- **Connection Diagram**, in which the components serve as the major computing blocks and the connectors provide encapsulated communication channels, including process diagram, concurrency diagram and data sharing diagram
- Above model diagrams provide the view of s/w architectural model from various angles, which can be used to measure the quality of architectural design

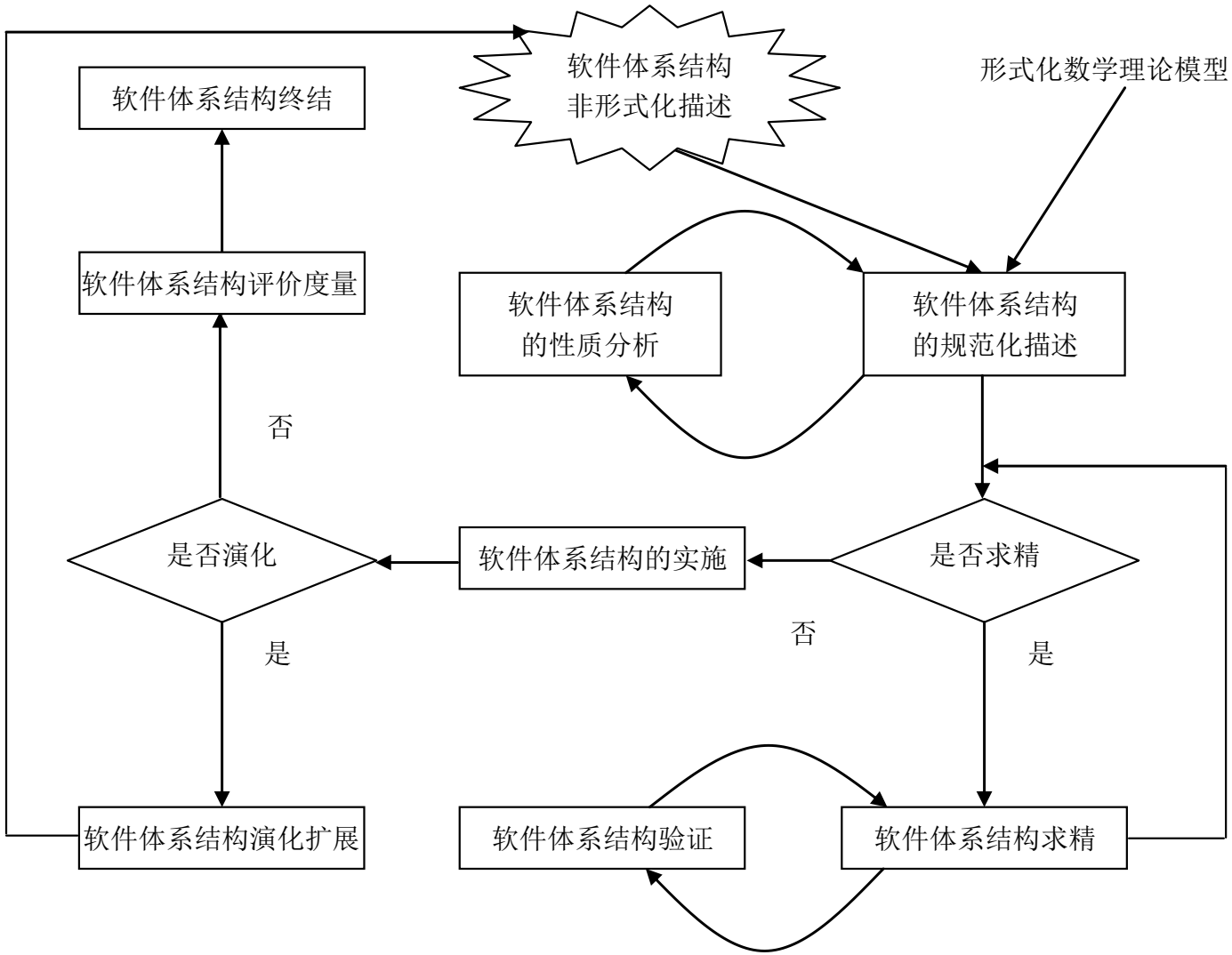
Software Architecture Life Cycle

- Non-formal natural language description of s/w architecture's life cycle of design, building, deployment, and measurement. Used in the early stage of software engineering by designers and engineers
- Formal specification of s/w architectural design and analysis through math model and formal definitions, which is more precise and regulated

软件架构模型

- Design, measure and improve --- a work cycle for building and optimizing software's architecture for large-scale software systems
- The analysis results or conclusions will be used to guide the architectural design, building and deployment of software systems
- The quality and performance measurement based on the architectural model and system outputs will be evaluated and reused in further improvement

软件架构模型


软件体系结构
非形式化描述
形式化数学理论模型
软件体系结构终结
软件体系结构评价度量
软件体系结构
的性质分析
软件体系结构
的规范化描述
否
是否演化
软件体系结构的实施
是否求精
否
是
是
软件体系结构演化扩展
软件体系结构验证
软件体系结构求精

End of Lecture

Thanks!