软件架构基础、理论和实践

1. **连接器（Connectors）：**
   - 模型化组件之间的相互作用
   - 包括规定这些相互作用的规则
2. **软件连接器的作用：**
   - 支持组件间的通信和协调
   - 提供转换服务
   - 管理系统中的数据流和控制流
3. **软件连接器的类型：**
   - 过程调用连接器
   - 事件连接器
   - 数据访问连接器
   - 链接连接器
   - 流连接器
   - 仲裁者连接器
   - 适配器连接器
   - 分发器连接器
4. **过程调用连接器：**
   - 控制流传递
   - 数据传递通过参数和返回值
   - 常用于远程过程调用（RPC）
5. **事件连接器：**
   - 响应事件触发的控制流
   - 提供通信服务
   - 用于异步通信
6. **数据访问连接器：**
   - 允许组件访问数据存储
   - 可能涉及数据格式转换
7. **链接连接器：**
   - 将系统组件连接在一起
   - 提供通信和协调通道
8. **流连接器：**
   - 处理组件之间的大量数据传输
   - 与其他连接器组合以执行复合任务
9. **仲裁者连接器：**
   - 解决组件之间的冲突
   - 控制流的重定向
10. **适配器连接器：**
    - 支持不同组件的互操作
    - 匹配通信策略和交互协议
    - 优化在给定执行环境中的组件交互
11. **分发器连接器：**
    - 识别和路由组件之间的交流和协调信息

- 用于分布式系统
12. **软件架构的灵活性和演变：**
    - 连接器的灵活性支持系统演变
    - 允许添加、删除、替换、重新连接和迁移组件
13. **连接器的难题和挑战：**
    - 刚性连接器
    - 连接器在实现中的“分散”
14. **性能与灵活性的权衡：**
    - 关键问题是性能与灵活性的平衡

Software Connectors
软件连接器
Software Architecture
软件架构
Copyright © Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy. All rights reserved.
版权所有 Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy。保留所有权利。
Foundations, Theory, and Practice
软件架构
2
Intro
简介
● Making architectural design decisions that pertain to integrating those components effectively and ensuring the components' proper interaction in a system is just as important.
制定与有效集成这些组件并确保系统中组件正确交互相关的架构设计决策同样重要。
● Assembling functional components and supporting interactions between components.
组装功能组件并支持组件之间的交互。
● Perform transfer of control and data among components.
在组件之间执行控制和数据的传输。
● Provide services, such as persistence, invocation, messaging, and transactions, that are largely independent of the interacting components' functionalities.
提供服务，如持久性、调用、消息传递和事务，这些服务在很大程度上独立于相互作用的组件功能。
Foundations, Theory, and Practice
软件架构
3
Intro
简介
● These services are usually considered to be "facilities components" in widely used middleware standards such as COREA, DCOM, and RMI.
这些服务通常被认为是广泛使用的中间件标准（如 COREA、DCOM 和 RMI）中的“设施组件”。
● Recognizing these facilities as connectors helps to clarify an architecture and keep the components' focus on application- and domain-specific concerns.

将这些设施视为连接器有助于澄清架构，并使组件集中在应用程序和领域特定的关注点上。

- Treating these services as connectors rather than components can also foster their reuse across applications and domains.

将这些服务视为连接器而不是组件还可以促进它们在应用程序和领域之间的重复使用。

- Connectors allow architects and engineers to compose heterogeneous functionality, developed at different times, in different locations, by different organizations.

连接器允许架构师和工程师在不同的时间、地点和组织中组合异构功能。

- Connectors can be thought of quite appropriately as the guards at the gate of separation of concerns.

可以合理地将连接器看作是关注点分离之门的守卫。Foundations, Theory, and Practice

软件架构

4

What is a Software Connector?

软件连接器是什么？

- Architectural element that models

Interactions among components

Rules that govern those interactions

建模组件之间的交互和管理这些交互的规则的架构元素

- Simple interactions

Procedure calls

Shared variable access

简单的交互

过程调用

共享变量访问

- Complex & semantically rich interactions

Client-server protocols

Database access protocols [JDBC]

Asynchronous event multicast

复杂且语义丰富的交互

客户端-服务器协议

数据库访问协议[JDBC]

异步事件多播

- Each connector provides

Interaction duct(s)

Transfer of control and/or data

每个连接器提供

交互导管

控制和/或数据的传输 Foundations, Theory, and Practice

软件架构

5

What is a Software Connector?

软件连接器是什么？

- Traditional software engineering approaches embrace a rather

narrow view of connectors that is unlikely to be successfully

applicable across all development situations.
传统的软件工程方法对连接器有一个相当狭窄的观点，不太可能在所有开发情况下都成功应用。

● This problem is further exacerbated by the increased
emphasis on development using large, off-the-shelf
components originating from multiple sources.
通过增加对使用大型现成组件进行开发的强调，这个问题进一步加剧，这些组件来自多个源头。

● As components become more complex and heterogeneous,
the interactions among them become more critical
determinants of system properties.
随着组件变得更加复杂和异构，它们之间的交互成为系统属性更为关键的决定因素。

Foundations, Theory, and Practice
软件架构
6

What is a Software Connector?
软件连接器是什么？

● Very important dimension of a software architect's job is to:
非常重要的软件架构师工作维度之一是：

Understand the component interaction needs.
了解组件交互需求。

Carefully identify all of the relevant attributes of that
interaction.
仔细识别该交互的所有相关属性。

Select the candidate connectors.
选择候选连接器。

Assess any trade-offs associated with each candidate.
评估与每个候选连接器相关的任何权衡。

Analyze the key properties of the resulting component,
connector assemblies.
分析生成的组件、连接器集合的关键属性。Foundations, Theory, and Practice
软件架构
7

Where are Connectors in Software
Systems?
在软件系统中，连接器在哪里？

Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。

Foundations, Theory, and Practice
软件架构
8

Implemented vs. Conceptual

Connectors
实现 vs. 概念连接器
- Connectors in software system implementations

Frequently no dedicated code
Frequently no identity
Typically do not correspond to compilation units
Distributed implementation
在软件系统实现中的连接器
通常没有专用代码
通常没有身份
通常不对应于编译单元
分布式实现

- Across multiple modules
- Across interaction mechanismsFoundations, Theory, and Practice

软件架构
9
Implemented vs. Conceptual
Connectors (cont’d)
实现 vs. 概念连接器（续）
- Connectors in software architectures

First-class entities
Have identity
Describe all system interaction
Entitled to their own specifications & abstractions
在软件体系结构中的连接器
一流实体
具有身份
描述所有系统交互
有权拥有自己的规范和抽象 Foundations, Theory, and Practice
软件架构
10
Reasons for Treating Connectors
Independently
将连接器独立处理的原因
- Connector Component

Components provide application-specific functionality
Connectors provide application-independent
interaction mechanisms
组件提供特定于应用程序的功能
连接器提供独立于应用程序的交互机制
- Interaction abstraction and/or parameterization
- Specification of complex interactions

Binary vs. N-ary

Asymmetric vs. Symmetric

Interaction protocols

交互抽象和/或参数化

- Specification of complex interactions

Binary vs. N-ary

Asymmetric vs. Symmetric

Interaction protocols

复杂交互的规范

二进制 vs. N-进制

非对称 vs. 对称

交互协议 Foundations, Theory, and Practice

软件架构

11

Treating Connectors Independently

(cont’d)

独立处理连接器（续）

- Localization of interaction definition
- Extra-component system (interaction) information
- Component independence
- Component interaction flexibility

交互定义的局部化

额外组件系统（交互）信息

组件独立性

组件交互的灵活性 Foundations, Theory, and Practice

软件架构

12

Benefits of First-Class Connectors

一流连接器的优势

- Separate computation from interaction
- Minimize component interdependencies
- Support software evolution

At component-, connector-, & system-level

- Potential for supporting dynamism
- Facilitate heterogeneity
- Become points of distribution
- Aid system analysis & testing

将计算与交互分开

最小化组件之间的相互依赖

支持软件演进

在组件、连接器和系统级别

An Example of Explicit Connectors

明确连接器的示例

Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。

An Example of Explicit Connectors
(cont’d)
明确连接器的示例（续）
Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。

Software Connector Roles
软件连接器的角色

- Locus of interaction among set of components
组件集之间的交互位置
- Protocol specification (sometimes implicit) that defines its properties
协议规范（有时是隐式的），定义其属性
Types of interfaces it is able to mediate
它能够调解的接口类型
Assurances about interaction properties
对交互属性的保证
Rules about interaction ordering
关于交互顺序的规则
Interaction commitments (e.g., performance)
交互承诺（例如性能）
- Roles
角色
Communication 通信
Coordination 协调
Conversion 转换
Facilitation 便利

Connectors as Communicators
连接器作为通信者

- Main role associated with connectors
与连接器相关的主要角色

- Supports

支持

Different communication mechanisms

不同的通信机制

- e.g. procedure call, RPC, shared data access, message passing

例如：过程调用、RPC、共享数据访问、消息传递

Constraints on communication structure/direction

对通信结构/方向的限制

- e.g. pipes

例如：管道

Constraints on quality of service

对服务质量的限制

- e.g. persistence

例如：持久性

- Separates communication from computation (SOC)

将通信与计算分离（SOC）

- May influence non-functional system characteristics

可能影响非功能性系统特性

例如性能、可伸缩性、安全性 Foundations, Theory, and Practice

软件架构

17

Connectors as Coordinators

连接器作为协调者

- Determine computation control

确定计算控制

- Control delivery of data

控制数据传递

- Separates control from computation

将控制与计算分离

- Orthogonal to communication, conversion, and facilitation

正交于通信、转换和便利

Elements of control are in communication, conversion
and facilitation

控制的元素在通信、转换和便利中

support transfer of control among components.

支持在组件之间传递控制。

passing the thread of execution to each other.

将执行线程相互传递。

Function calls and method invocations are examples of
coordination connectors. Foundations, Theory, and Practice

软件架构

18

Connectors as Converters

连接器作为转换器

- Enable interaction of independently developed, mismatched components

使独立开发的、不匹配的组件进行交互

- Mismatches based on interaction

Type

Number

Frequency

Order

基于交互的不匹配

类型

数量

频率

顺序

- Examples of converters

转换器的示例

Adaptors (adapts input to other interface, a bridge)

适配器（将输入适应到其他接口，桥接）

Wrappers (simplify API)Foundations, Theory, and Practice

软件架构

19

Connectors as Facilitators

连接器作为便利设施

- Enable interaction of components intended to interoperate

使打算互操作的组件进行交互

Mediate and streamline interaction

调解和简化交互

- Govern

access to shared information

管理对共享信息的访问

- Ensure proper performance profiles (non-functional req.)

确保适当的性能概要（非功能需求）

例如负载平衡、调度服务、并发控制

- Provide synchronization mechanisms

提供同步机制

Critical sections

关键部分

Monitors

监视器

further facilitating and optimizing components’ interactionsFoundations, Theory, and Practice

软件架构

20

Connector Types

连接器类型

- Procedure call

过程调用
- Data access

数据访问
- Event

事件
- Stream

流
- Linkage

链接
- Distributor

分发器
- Arbitrator

仲裁者
- AdaptorFoundations, Theory, and Practice

软件架构
21
A Framework for Classifying Connectors
对连接器分类的框架
Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。
• Each connector is identified by its primary service category and further refined based on the choices made to realize these services.
每个连接器都通过其主要服务类别进行标识，并根据实现这些服务所做的选择进一步细化。
• A particular connector instance (that is, species) can take a number of values from different types.
特定的连接器实例（即物种）可以从不同类型中取得多个值。
• does not result in a strict hierarchy, but rather in a directed acyclic graph. Foundations, Theory, and Practice

Procedure Call Connectors
过程调用连接器
Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。
Foundations, Theory, and Practice
软件架构
23
Procedure Call Connectors
过程调用连接器
Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.

软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。

Event Connectors
事件连接器
Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。
● Similar to procedure call connectors in that they affect the flow of control among components. The flow is precipitated by an event.
类似于过程调用连接器，因为它们影响组件之间的控制流。控制流由事件引发。
● Once the event connector learns about the occurrence of an event, it generates messages (event notifications) for all interested parties and yields control to the components for processing those messages.
一旦事件连接器得知事件的发生，它会为所有感兴趣的方生成消息（事件通知），并将控制权交给组件来处理这些消息。
● The contents of an event can be structured to contain more information about the event, such as the time and place of occurrence, and other Application-specific data.
事件的内容可以被构造成包含有关事件的更多信息，例如发生的时间和地点，以及其他特定于应用程序的数据。
● Event connectors therefore also provide communication service.
因此，事件连接器还提供通信服务。

Event Connectors
事件连接器
Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。
● Event connectors are also different from procedure calls in that virtual connectors are formed between components interested in the same event topics.
事件连接器与过程调用不同之处在于，对于对相同事件主题感兴趣的组件之间形成了虚拟连接器。
● Those connectors may appear and disappear dynamically depending on the components' changing interests.
这些连接器可能会根据组件兴趣的变化而动态出现和消失。
● Event-based distributed systems rely on the notion of time and ordering of actions. Dimensions such as causality, atomicity, and synchronicity play a critical role.
基于事件的分布式系统依赖于时间和操作顺序的概念。因果关系、原子性和同步性等维度起

着关键作用。

● Event connectors are found in distributed applications that require asynchronous communication.

事件连接器存在于需要异步通信的分布式应用程序中。

Foundations, Theory, and Practice

软件架构

26

Event Connectors

事件连接器

Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.

软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。

Eg. the cardinality of a multicasting event will be a single producer and multiple observer components.

例如，多播事件的基数将是单个生产者和多个观察者组件。

Foundations, Theory, and Practice

软件架构

27

Data Access Connectors

数据访问连接器

Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.

软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。

● Data access connectors allow components to access data maintained by a datastore component (communication services).

数据访问连接器允许组件访问由数据存储组件（通信服务）维护的数据。

● Data access often requires preparation of the data store before and cleanup after access has been completed.

数据访问通常要求在访问完成之前准备数据存储，并在访问完成后进行清理。

● In case there is a difference in the format of the required data and the format in which data is stored and provided, data access connectors may perform translation of the information being accessed, that is, conversion.

如果所需数据的格式与存储和提供数据的格式不同，数据访问连接器可能会执行所访问信息的翻译，即转换。

Foundations, Theory, and Practice

软件架构

28

Data Access Connectors

数据访问连接器

Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor

, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with

permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。

● The data can be stored either persistently or temporarily.
数据可以被持久地或暂时地存储。

● Examples of persistent data access include query mechanisms, such as SQL for database access, and accessing information in repositories, such as a software component repository.
持久数据访问的示例包括查询机制，例如用于数据库访问的 SQL，以及访问存储库中的信息，例如软件组件存储库。

● Examples of transient data access include heap and stack memory access and information caching.
短暂数据访问的示例包括堆和栈内存访问以及信息缓存。

Data Access Connectors
数据访问连接器

Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。

Linkage Connectors
链接连接器

Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。

● are used to tie the system components together and hold them in such a state during their operation.
用于将系统组件紧密连接在一起，并在其运行期间保持它们处于这样的状态。

● enable the establishment of ducts-the channels for communication and coordination;
启用管道的建立 - 用于通信和协调的通道；

● then used by higher-order connectors to enforce interaction semantics. In other words, linkage connectors provide facilitation services.
然后由更高级别的连接器使用，以强制执行交互语义。换句话说，链接连接器提供了便利服务。

Linkage Connectors
链接连接器

Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。
The granularity dimension refers to the size of components and level of detail required to establish a linkage.
颗粒度维度指的是建立链接所需的组件大小和详细级别。
Unit interconnection specifies only that one component-which can be a module, an object, or a file-depends on another. Eg. configuration management and system building facilities such as Make.
单元互连仅指定一个组件（可以是模块、对象或文件）依赖于另一个组件。例如，配置管理和系统构建设施，如 Make。
Semantic interconnection ensures that the interaction requirements and constraints are explicitly stated and satisfied.
语义互连确保明确陈述并满足交互要求和约束。

Linkage Connectors
链接连接器
Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。
The cardinality of a linkage connector refers to the number of places in which a system resource-such as a component, procedure, or variable-is defined, used, provided, or required.
链接连接器的基数指的是系统资源（如组件、过程或变量）在其中定义、使用、提供或所需的位置数。
A resource is defined and/or provided in a single location and used or required from multiple locations.
资源在单个位置被定义和/或提供，并且在多个位置被使用或需要。
A linkage connector can establish the binding between components very early, early, or late.
链接连接器可以在组件之间建立绑定，非常早期、早期或晚期。

Stream Connectors
流连接器
Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。
- perform transfers of large amounts of data between autonomous processes, thus provide communication services in a system.

执行在自治进程之间传输大量数据，从而在系统中提供通信服务。

- Streams are also used in client-server systems with data transfer protocols to deliver results of computation.

流还用于具有数据传输协议的客户端-服务器系统

Streams can be combined with other connector types, such as data access connectors, to provide composite connectors for performing database and file storage access, event connectors, to multiplex the delivery of a large number of events.

- Examples: Unix pipes, TCP/UDP communication sockets, and proprietary client-server protocols.

流连接器可以与其他连接器类型结合，例如数据访问连接器，以提供用于执行数据库和文件存储访问的复合连接器，事件连接器，以复用大量事件传递。

- 例子：Unix 管道，TCP/UDP 通信套接字和专有的客户端-服务器协议。

Arbitrator Connectors

仲裁连接器

- When components are aware of the presence of other components but cannot make assumptions about their needs and state, arbitrators streamline system operation and resolve any conflicts (thereby providing facilitation services), and redirect the flow of control (providing coordination services).

当组件意识到其他组件的存在但不能对它们的需求和状态进行假设时，仲裁者简化系统操作并解决任何冲突（从而提供便利服务），并重新定向控制流（提供协调服务）。

Arbitrator Connectors

仲裁连接器

Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.

软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。

- When components are aware of the presence of other components but cannot make assumptions about their needs and state, arbitrators streamline system operation and resolve any conflicts (thereby providing facilitation services), and redirect the flow of control (providing coordination services).

当组件意识到其他组件的存在但不能对它们的需求和状态进行假设时，仲裁者简化系统操作并解决任何冲突（从而提供便利服务），并重新定向控制流（提供协调服务）。

- e.g.
  use
synchronization
and
concurrency
control
to
guarantee consistency and atomicity of operations.
  negotiate service levels and mediate interactions

requiring guarantees for reliability and atomicity.
provide scheduling and load balancing services.
ensure system trustworthiness by providing crucial
support for dependability in the form of reliability, safety,
and security.

Adaptor Connectors
适配器连接器
Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。

- provide facilities to support interaction between components that have not been designed to interoperate.
- involve matching communication policies and interaction protocols among components, thereby providing conversion services.
- interoperation of components in heterogeneous environments.
- optimize component interactions for a given execution environment: remote call->convert to->local call

提供设施以支持未经设计以进行相互操作的组件之间的交互。
涉及在组件之间匹配通信策略和交互协议，从而提供转换服务。
在异构环境中组件之间的相互操作。
为给定的执行环境优化组件相互作用：远程调用->转换为->本地调用

Distributor Connectors
分发器连接器
Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.
软件架构：基础、理论和实践；Richard N. Taylor、Nenad Medvidovic 和 Eric M. Dashofy；© 2008 John Wiley & Sons, Inc. 版权所有，已获得许可。

- perform the identification of interaction paths and subsequent routing of communication and coordination information among components along these paths (facilitation services).
- provide assistance to other connectors, such as streams or procedure calls.
- Distributed systems use distributor connectors to direct the data flow.
- Domain name service (DNS), routing, switching, and many other network services belong to this type.
- have an important effect on system scalability and survivability.

执行对交互路径的识别，随后在这些路径上在组件之间进行通信和协调信息的路由（提供便利服务）。
为其他连接器提供帮助，例如流或过程调用。
分布式系统使用分发器连接器来指导数据流。
域名服务（DNS）、路由、交换和许多其他网络服务属于此类型。
对系统的可扩展性和生存能力有重要影响。

Discussion

讨论

- Connectors allow modeling of arbitrarily complex interactions
- Connector flexibility aids system evolution

Component addition, removal, replacement,

reconnection, migration

- Support for connector interchange is desired

Aids system evolution

May not affect system functionality

连接器允许对任意复杂的交互进行建模

连接器的灵活性有助于系统演化

组件的添加、删除、替换、重新连接、迁移

期望支持连接器的互换

有助于系统演化

可能不会影响系统功能

Discussion 讨论

- Libraries of OTS connector implementations allow developers to focus on application-specific issues
- Difficulties Rigid connectors Connector “dispersion” in implementations
- Key issue Performance vs. flexibility

开发人员可以专注于应用程序特定问题的 OTS 连接器实现库

困难 刚性连接器 连接器在实现中的"分散"

关键问题 性能与灵活性