输入文本涉及软件架构中的连接器选择，重点讨论了单元互联、语法互联和语义互联这几个软件互联模型。单元互联主要关注系统单元之间的依赖关系，语法互联描述了编程语言中语法元素之间的关系，而语义互联则表达了系统组件的设计者意图和实际使用方式。

在选择连接器方面，文本提到了多种类型的连接器，包括过程调用、数据访问、流和分发器等。此外，还介绍了连接器维度的相互关系，包括要求、禁止、限制和警告等。

- 软件互联模型
  - 单元互联
    - 系统单元关系定义
    - 示例：编译上下文、重新编译策略、系统建模
    - 特点：粗粒度、静态、依赖关系
  - 语法互联
    - 编程语言中的语法元素关系
    - 示例：自动化软件变更管理、静态分析、智能重新编译、系统建模
    - 特点：细粒度、静态&动态、不完整规范
  - 语义互联
    - 系统组件设计者意图和实际使用方式
    - 示例：语义互联的例子
    - 特点：基于语法互联、静态&动态、完整规范、架构层面的必要性
- 连接器选择
  - 多种类型的连接器
    - 过程调用
    - 数据访问
    - 流
    - 分发器
  - 连接器维度相互关系
    - 要求
    - 禁止
    - 限制
    - 警告
- 复合连接器
  - 已知的复合连接器
    - 网格连接器
    - 点对点连接器
    - 客户端-服务器连接器
    - 基于事件的连接器

```

**Choosing Connectors**
**选择连接器**
```

How do we enable components A and B to interact?
我们如何使组件 A 和 B 进行交互？

Attach adapter to A or B
将适配器连接到 A 或 B

Change A’s form to B’s form
将 A 的形式更改为 B 的形式

Make B multilingual
使 B 支持多语言

Publish abstraction of A’s form
发布 A 的形式的抽象

Provide B with import/export converter
为 B 提供导入/导出转换器

Introduce intermediate form
引入中间形式

Negotiate to find common form for A and B
协商找到 A 和 B 的共同形式

Transform on the fly
实时转换

Maintain multiple versions of A
维护 A 的多个版本

Separate B’s "essence" from its packaging
将 B 的“本质”与其封装分离

What is the right answer?
什么是正确的答案？

How Does One Select a Connector?
如何选择连接器？

Determine a system’s interconnection and interaction needs
确定系统的互联和交互需求

Determine roles to be fulfilled by the system’s connectors

确定系统连接器要完成的角色

For each connector:
- Determine its appropriate type(s)
- Determine its dimensions of interest
- Select appropriate values for each dimension
对于每个连接器：
- 确定其适当的类型
- 确定其感兴趣的维度
- 为每个维度选择适当的值

For multi-type, i.e., composite connectors:
- Determine the atomic connector compatibilities
- Perform a trade-off analysis among multiple possible solutions
对于多类型，即复合连接器：
- 确定原子连接器的兼容性
- 在多个可能的解决方案之间进行权衡分析

Simple Example
简单示例

System components will execute in two processes on the same host
系统组件将在同一主机上的两个进程中执行

Mostly intra-process
主要是进程内

Occasionally inter-process
偶尔是进程间

The interaction among the components is synchronous
组件之间的交互是同步的

The components are primarily computation-intensive
组件主要是计算密集型

There are some data storage needs, but those are secondary
有一些数据存储需求，但这些是次要的

Select procedure call connectors for intra-process interaction
选择过程调用连接器进行进程内交互

Combine procedure call connectors with distributor connectors for inter-process interaction
将过程调用连接器与分发器连接器结合以进行进程间交互

RPC
远程过程调用

Select the values for the different connector dimensions
为不同的连接器维度选择值

What are the appropriate values?
什么是适当的值？

What values are imposed by your favorite programming language(s)?
你喜欢的编程语言强加了哪些值？

Procedure Call Connectors Revisited
重新审视过程调用连接器

Distributor Connectors Revisited
重新审视分发器连接器

Two Connector Types in Tandem
两种连接器类型串联

Select the appropriate values for PC and RPC!
选择 PC 和 RPC 的适当值！

Software Interconnection Models
软件互联模型

Interconnection models (IM) as defined by Perry
由 Perry 定义的互联模型

Unit interconnection
单元互联

Syntactic interconnection
语法互联

Semantic interconnection
语义互联

All three are present in each system
所有三者都存在于每个系统中

Are all equally appropriate at the architectural level?

在架构层面上它们是否都同样合适？

Unit Interconnection
单元互联

Defines relations between the system’s units
定义系统单元之间的关系

Units are components (modules or files)
单元是组件（模块或文件）

Basic unit relationship is dependency
基本单元关系是依赖

Unit-IM = ({units},{“depends on”})
单元-IM =（{单元},{“依赖于”}）

Examples
示例

Determining the context of compilation
确定编译的上下文

e.g., C preprocessor
例如，C 预处理器

IM = ({files},{“include”})
IM =（{文件},{“包括”}）

Determining recompilation strategies
确定重新编译策略

e.g., Make facility
例如，Make 工具

IM = ({compile_units},{“depends on”,“has changed”})
IM =（{编译单元},{“依赖于”,“已更改”}）

System modeling
系统建模

e.g., RCS, DVS, SVS, SCCS
例如，RCS、DVS、SVS、SCCS

IM = ({systems, files},{“is composed of”})
IM =（{系统，文件},{“由...组成”}）

Unit Interconnection Characteristics
单元互联特点

Coarse-grain interconnections
粗粒度的互联

At the level of entire components
在整个组件的层次上

Interconnections are static
互联是静态的

Does not describe component interactions
不描述组件间的交互，专注于依赖关系

Syntactic Interconnection
语法互联

Describes relations among syntactic elements of programming languages
描述编程语言中的语法元素之间的关系

Variable definition/use
变量定义/使用

Method definition/invocation
方法定义/调用

IM = ({methods, types, variables, locations},{“is def at”，“is set at”，“is used at”，“is del from”，“is changed to”，“is added to”})
IM =（{方法，类型，变量，位置},{“在...处被定义”，“在...处被设置”，“在...处被使用”，“从...处被删除”，“被更改为...”，“被添加到...”}）

Examples
示例

Automated software change management
自动化软件变更管理

e.g., Interlisp’s masterscope
例如，Interlisp 的 masterscope

Static analysis
静态分析

e.g., Detection of unreachable code by compilers
例如，编译器检测不可达代码

Smart recompilation
智能重新编译

Changes inside the unit ➔ recompilation of only the changes
单元内的更改 ➔ 仅重新编译更改部分

System modeling
系统建模

Finer level of granularity than unit-IM
比单元互联更细粒度

Syntactic Interconnection Characteristics
语法互联特点

Finer-grain interconnections
细粒度的互联

At the level of individual syntactic objects
在个别语法对象的层次上

Interconnections are static & dynamic
互联是静态和动态的

Incomplete interconnection specification
互联规范不完整

Valid syntactic interconnections may not be allowed by semantics
有效的语法互联可能会受到语义的限制

Operation ordering, communication transactions
操作排序，通信事务

e.g., Pop on an empty stack
例如，在空栈上执行弹栈操作

Violation of (intended) operation semantics
违反（预期的）操作语义

e.g., Trying to use calendar add operation to add integers
例如，试图使用日历添加操作来添加整数

Semantic Interconnection
语义互联

Expresses how system components are meant to be used
表达了系统组件的设计者意图

Component designers' intentions
组件设计者的意图

Captures how system components are actually used
捕捉系统组件实际使用的方式

Component users' (i.e., system builders') intention
组件用户（即系统构建者）的意图

Interconnection semantics can be formally specified
互联语义可以被正式指定

Pre- & post-conditions
前置条件和后置条件

Dynamic interaction protocols (e.g., CSP, FSM)
动态交互协议（例如，CSP，FSM）

IM = ({methods, types, variables, ..., predicates},{"is set at", "is used at", "calls", "called by", ..., "satisfies"})
IM =（{方法，类型，变量，...，谓词},{"在...处被设置"，"在...处被使用"，"调用"，"被调用"，...，"满足"}）

Example of Semantic Interconnection
语义互联的示例

connector Pipe =
role Writer = write → Writer ∏ close → ✓
role Reader =
let ExitOnly = close → ✓
in let DoRead = (read → Reader read-eof → ExitOnly)
in DoRead ∏ ExitOnly
glue = let ReadOnly = Reader.read → ReadOnly
Reader.read-eof → Reader.close → ✓

Reader.close → ✓
in let WriteOnly = Writer.write → WriteOnly
Writer.close → ✓
in Writer.write → glue
Reader.read → glue
Writer.close → ReadOnly
Reader.close → WriteOnly

Semantic Interconnection Characteristics
语义互联特点

Builds on syntactic interconnections
基于语法互联

Interconnections are static & dynamic
互联是静态和动态的

Complete interconnection specification
完整的互联规范

Specifies both syntactic & semantic interconnection validity
指定语法和语义互联的有效性

Necessary at the level of architectures
在架构层面上是必要的

Large components
大型组件

Complex interactions
复杂的交互

Heterogeneity
异构性

Component

reuse
组件重用

What about ensuring other properties of interaction?
如何确保互动的其他属性？
Robustness, reliability, security, availability, ...

Composing Basic Connectors
组合基本连接器

In many systems, a connector of multiple types may be required to service (a subset of) the components
在许多系统中，可能需要多类型的连接器来为（一部分）组件提供服务

All connectors cannot be composed
不能将所有连接器组合

Some are naturally interoperable
有些天然可互操作

Some are incompatible
有些不兼容

All are likely to require trade-offs
所有可能需要权衡

The composition can be considered at the level of connector type dimensions and subdimensions
可以在连接器类型的维度和子维度的层面上考虑组合

Connector Dimension Inter-Relationships
连接器维度的相互关系

Requires – Choice of one dimension mandates the choice of another
要求 – 选择一个维度将强制选择另一个维度

Prohibits – Two dimensions can never be composed into a single connector
禁止 – 两个维度永远不能组合成一个连接器

Restricts – Dimensions are not always required to be used together
限制 – 不必总是一起使用维度

Cautions – Combinations may result in unstable or unreliable connectors
警告 – 组合可能导致不稳定或不可靠的连接器

Dimension Inter-Relationships in a Nutshell
维度相互关系概述

Well-Known Composite Connectors
已知的复合连接器

Grid connectors (e.g., Globus)

网格连接器（例如，Globus）
- Procedure call
- Data access
- Stream
- Distributor

Peer-to-peer connectors (e.g., Bittorrent)
点对点连接器（例如，Bittorrent）
- Arbitrator
- Data access
- Stream
- Distributor

Client-server connectors
客户端-服务器连接器

Event-based connectors
基于事件的连接器