第17章基于组件的软件工程

- 第1讲
- <编号>
- 第十七章 软件重用

涵盖的主题

- 17.1 组件和组件模型
- 17.2 CBSE 流程
- 17.3 组件组成
- <编号>
- 第十七章 软件重用

基于组件的开发

- 基于组件的软件工程 (CBSE) 是一种软件开发方法，它依赖于称为“软件组件”的实体的重用。（CBSE 是定义、实施和集成或组合松散耦合的独立组件到系统中的过程）
- 它源于 1990 年代后期面向对象开发未能支持有效重用。单个对象类过于详细和具体。
- 组件比对象类更抽象（由它们的接口定义），可以被认为是独立的服务提供者。它们可以作为独立实体存在。
- <编号>
- 第十七章 软件重用

CBSE 要点

- CBSE的要点是：
- 由其接口指定的独立组件。
- 便于组件集成的组件标准。
- 为组件互操作性提供支持的中间件。
- 一个面向重用的开发过程。
- <编号>
- 第十七章 软件重用

CBSE 和设计原则

- 除了重用的好处之外，CBSE 还基于完善的软件工程设计原则：
    - 组件是独立的，所以不会相互干扰；
    - 组件实现是隐藏的；
    - 通信是通过定义良好的接口进行的；
    - 组件基础设施/平台提供了一系列可用于应用系统的标准服务。这些服务是共享的，可以降低开发成本。

- <编号>
- 第十七章 软件重用

组件标准

- 需要建立标准，以便组件可以相互通信和互操作。
- 不幸的是，建立了几个相互竞争的组件标准：
  - Sun 的企业 Java Bean (EJB)
  - 微软的 COM 和 .NET
  - CORBA的CCM(OMG)(CCM-CORBA Component Model, CORBA-Common Object Request Broker Architecture, OMG-Object Management Group)
- 在实践中，这些多重标准阻碍了 CBSE 的采用。使用不同方法开发的组件不可能协同工作。
- <编号>
- 第十七章 软件重用

CBSE问题

- 组件可信度——一个没有可用源代码的组件如何被信任?
- 组件认证——谁来认证组件的质量?
- 紧急属性预测 - 如何预测组件组合的紧急属性?
- 需求权衡——我们如何在一个组件和另一个组件的特性之间进行权衡分析?
- <编号>
- 第十七章 软件重用

17.1 组件和组件模型

- 组件提供服务而不考虑组件在何处执行或其编程语言
  - 组件是一个独立的可执行实体，可以由一个或多个可执行对象组成；
  - 组件接口发布，所有交互都通过发布的接口；
- <编号>
- 第十七章 软件重用

组件定义

- 人们对组件提出了不同的定义：
- 委员会和海因曼：
  - *软件组件是符合组件模型的软件元素，可以根据组合标准独立部署和组合，无需修改。*
- 西珀斯基：
  - *软件组件是一个组合单元，仅具有合同指定的接口和显式上下文相关性。软件组件可以独立部署，并由*

第三方组成。

- <编号>
- 第十七章 软件重用

组件特性

元件特性	描述
标准化	组件标准化意味着在 CBSE 过程中使用的组件必须符合标准组件模型。该模型可以定义组件接口、组件元数据、文档、组合和部署。
独立的	一个组件应该是独立的——应该可以组合和部署它而不必使用其他特定的组件。在组件需要外部提供服务的情况下，这些应该在“需要”接口规范中明确规定。
可组合	对于可组合的组件，所有外部交互都必须通过公开定义的接口进行。此外，它必须提供对自身信息的外部访问，例如其方法和属性。

- <编号>
- 第十七章 软件重用

组件特性

元件特性	描述
可部署	为了可部署，组件必须是自包含的。它必须能够在提供组件模型实现的组件平台上作为独立实体运行。这通常意味着组件是二进制的，在部署之前不必编译。如果将组件实现为服务，则不必由组件的用户部署。相反，它由服务提供商部署。
记录在案	组件必须完整记录，以便潜在用户可以决定组件是否满足他们的需求。应该指定所有组件接口的语法和语义。

- <编号>
- 第十七章 软件重用

组件作为服务提供者

- 该组件是一个独立的、可执行的实体。在与其他组件一起使用之前，不必对其进行编译。
- 组件提供的服务通过接口提供，所有组件交互都通过该接口进行。
- 组件接口以参数化操作的形式表示，其内部状态从不公开。
- <编号>
- 第十七章 软件重用

组件接口

- 组件有两个相关的接口：
- “提供”界面
  - 定义组件向其他组件提供的服务。
  - 这个接口本质上是组件 API。它定义了组件用户可以调用的方法。
- “需要”界面
  - 指定如果组件要正确运行，系统中的其他组件必须提供哪些服务。如果这些不可用，则组件将无法工作。
  - 这不会损害组件的独立性或可部署性，因为“需要”接口没有定义应如何提供这些服务。
- <编号>
- 第十七章 软件重用

组件接口

- 在 UML 组件图中，'provides' 接口是一个圆圈，'requires' 接口是一个半圆，位于组件图标的行尾；
- 注意 UML 符号。球（提供接口）和插座（需要接口）可以组合在一起。
- <编号>
- 第十七章 软件重用

示例：数据收集器组件的模型

- <编号>
- 第十七章 软件重用
- 此图显示了一个组件模型，该组件设计用于从传感器阵列收集信息。它自主运行以在一段时间内收集数据，并根据请求将收集的数据提供给调用组件。
- “provides”接口包括addSensor()、removeSensor()等方法；
- 这些方法具有指定传感器标识符、位置等的相关参数。
- “requires”接口用于将组件连接到传感器。

17.1.1 组件模型

- 组件模型是组件实现、文档和部署标准的定义。
- 组件模型示例
    - EJB 模型（企业 Java Bean）
    - COM+ 模型（.NET 模型）
    - Corba 组件模型（CCM）（Corba - 公共对象请求代理架构）
- 组件模型指定应如何定义接口以及应包含在接口定义中的元素。
- <编号>
- 第十七章 软件重用

组件模型的基本元素

- <编号>
- 第十七章 软件重用
- 该图总结了模型元素。
- 它显示了组件模型的元素定义了组件接口、在程序中使用组件所需的信息以及应该如何部署组件。下一张幻灯片显示了更多详细信息。

组件模型的元素

- 接口
    - 组件是通过指定它们的接口来定义的。组件模型指定应如何定义接口以及应包含在接口定义中的元素，例如操作名称、参数和异常。
- 用法
    - 为了远程分发和访问组件，它们需要具有唯一的名称或与之关联的句柄。这必须是全球唯一的。
- 部署
    - 组件模型包括如何将组件打包以部署为独立的可执行实体的规范。
- <编号>
- 第十七章 软件重用

中间件支持

- 组件模型是为执行组件提供支持的中间件的基础。
- 组件模型实现提供：
    - 允许根据模型编写的组件进行通信的平台服务；
    - 支持服务是由不同组件使用的独立于应用程序的服务。
    - 下一张幻灯片中的更多详细信息。
- 为了使用模型提供的服务，组件被部署在容器中。这是一组用于访问服务实现的接口。
- <编号>

- 第十七章 软件重用

组件模型中定义的中间件服务

- <编号>
- 第十七章 软件重用
    - 平台服务，使组件能够在分布式环境中进行通信和互操作。这些是所有基于组件的系统中必须提供的基本服务；
    - 支持服务，这是许多不同组件可能需要的公共服务。例如，许多组件需要身份验证以确保组件服务的用户得到授权。提供一组标准的中间件服务供所有组件使用是有意义的。这降低了组件开发的成本并且可以避免潜在的组件不兼容。

17.2 CBSE 流程

- CBSE 流程是支持基于组件的软件工程的软件流程。
    - 它们考虑了重用的可能性以及开发和使用可重用组件所涉及的不同流程活动。
- 重用开发
    - 此过程涉及开发将在其他应用程序中重用的组件或服务。它通常涉及对现有组件进行泛化。（面向重用）
- 重用开发
    - 此过程是使用现有组件和服务开发新应用程序的过程。（基于重用）
- <编号>
- 第十七章 软件重用

CBSE 流程

- <编号>
- 第十七章 软件重用
    - 带有重用和用于重用的 CBSE 的基本过程具有与组件获取、组件管理和组件认证相关的支持过程。下一张幻灯片中的更多详细信息。

支持流程

- 组件获取是获取组件以供重用或开发为可重用组件的过程。
    - 它可能涉及访问本地开发的组件或服务，或从外部源查找这些组件。
- 组件管理涉及管理公司的可重用组件，确保它们被正确编目、存储并可供重用。
- 组件认证是检查组件并证明其符合规格的过程。
- 第十七章 软件重用

- <编号>

关键点

- CBSE 是一种基于重用的方法，用于定义松散耦合的组件并将其实现到系统中。
- 组件是一个软件单元，其功能和依赖关系完全由其接口定义。
- 组件模型定义了一组组件提供者和编写者应该遵循的标准。
- 关键的 CBSE 过程是用于重用的 CBSE 和带有重用的 CBSE。
- <编号>
- 第十七章 软件重用

第17章基于组件的软件工程

- 第二讲
- <编号>
- 第十七章 软件重用

17.2.1 用于重用的 CBSE

- 用于重用的 CBSE 侧重于组件开发。
- 为特定应用程序开发的组件通常必须通用化以使其可重用。
- 如果组件与稳定的域抽象（业务对象）相关联，那么它最有可能是可重用的。
- 例如，在医院中，稳定域抽象与基本目的相关——护士、患者、治疗等。
- <编号>
- 第十七章 软件重用

可重用的组件开发

- 可重用的组件可以通过概括现有组件来特殊构造。
- 组件可重用性
    - 应该反映稳定的领域抽象；
    - 应该隐藏状态表示；
    - 应该尽可能独立；
    - 应该通过组件接口发布异常。
- 可重用性和可用性之间存在权衡
    - 界面越通用，可重用性就越大，但它也越复杂，因此可用性越低。
- <编号>
- 第十七章 软件重用

可重用性的变化

- 您可以对组件进行更改以使其更可重用，包括：
- 删除特定于应用程序的方法。
- 更改名称以使其通用。
- 添加方法以扩大覆盖范围。
- 使异常处理一致。
- 添加组件适配的配置接口。
- 集成所需的组件以减少依赖性。
- <编号>
- 第十七章 软件重用

异常处理

- 组件本身不应处理异常，因为每个应用程序对异常处理都有自己的要求。
    - 相反，组件应该定义可能出现的异常，并且应该将这些作为接口的一部分发布。
- 然而，在实践中，这有两个问题：
    - 发布所有异常会导致接口臃肿，难以理解。这可能会推迟组件的潜在用户。
    - 组件的操作可能依赖于本地异常处理，改变这一点可能会对组件的功能产生严重的影响。
- 第十七章 软件重用
- <编号>

遗留系统组件

- 可以将满足有用业务功能的现有遗留系统重新打包为组件以供重用。
- 这涉及编写实现提供和需要接口的包装器组件，然后访问遗留系统。
- 尽管成本高昂，但这比重写遗留系统要便宜得多。
- <编号>
- 第十七章 软件重用

可重用组件

- 可重用组件的开发成本可能高于特定等价物的成本。这种额外的可重用性增强成本应该是组织成本而不是项目成本。
- 通用组件的空间效率可能较低，并且执行时间可能比其特定的等效组件更长。
- <编号>
- 第十七章 软件重用

组件管理

- 组件管理涉及决定如何对组件进行分类以便可以发现它、使组件在存储库中或作为服务可用、维护有关组件使用的信息并跟踪不同的组件版本。
- 拥有重用计划的公司可能会在组件可供重用之前进行某种形式的组件认证。

    - 认证意味着除开发人员之外的其他人检查组件的质量。
- 第十七章 软件重用
- <编号>

17.2.2 可重用的 CBSE

- 具有重用过程的 CBSE 必须找到并集成可重用的组件。
- 在重用组件时，必须在理想需求和可用组件实际提供的服务之间进行权衡。
- 该过程包括：
    - 制定大纲要求；
    - 搜索组件，然后根据可用功能修改需求。
    - 再次搜索以查找是否有更好的组件满足修订后的要求。
    - 组合组件以创建系统。
    - 下一张幻灯片中的数字。
- <编号>
- 第十七章 软件重用

可重用的 CBSE

- <编号>
- 第十七章 软件重用
    - CBSE 的主要活动与重用过程
    - 带有重用的 CBSE 与用于原始软件开发的软件过程之间的本质区别是：
    - 用户需求最初是在大纲而不是细节中开发的，并且鼓励利益相关者在定义他们的需求时尽可能灵活。
    - 根据可用的组件，在流程的早期对需求进行细化和修改。
    - 在设计系统架构之后，还有进一步的组件搜索和设计改进活动。
    - 开发是一个组合过程，其中集成了发现的组件。

组件识别过程

- <编号>
- 第十七章 软件重用
    - 最初，您的重点应该放在搜索和选择上。
    - 您需要说服自己有可用的组件来满足您的要求。
    - 显然，您应该进行一些初始检查以确保组件合适，但可能不需要进行详细测试。
    - 在后期，系统架构设计完成后，你应该花更多的时间在组件验证上。
    - 您需要确信所识别的组件确实适合您的应用程序；如果没有，那么您必须重复搜索和选择过程。
    - CBSE 流程独有的一项活动是识别候选组件或服务以供重用。这涉及到多个子活动，如下图所示：

组件识别问题

- 相信。您需要能够信任组件的供应商。充其量，不受信任的组件可能不会像宣传的那样运行；在最坏的情况下，它可能会破坏您的安全。
- 要求。不同的组件组将满足不同的要求。
- 验证。
    - 组件规范可能不够详细，无法开发全面的测试。
    - 组件可能具有不需要的功能。您如何测试这不会干扰您的应用程序?
    - 下一张幻灯片中的更多详细信息。
- <编号>
- 第十七章 软件重用

组件验证

- 组件验证涉及为组件开发一组测试用例（或者，可能扩展随该组件提供的测试用例）和开发测试工具来运行组件测试。
    - 组件验证的主要问题是组件规范可能不够详细，无法让您开发一套完整的组件测试。
- 除了测试要重用的组件是否满足您的要求之外，您可能还必须检查该组件是否不包含您不需要的任何恶意代码或功能。下一张幻灯片显示了一个示例。
- 第十七章 软件重用
- <编号>

Ariane 启动器失败 – 验证失败?

- 1996年，阿丽亚娜5号火箭第一次试飞，在起飞37秒后发射装置失控，以灾难告终。
- 问题是由于之前版本的发射器（惯性导航系统）中的一个重复使用的组件失败了，因为在开发该组件时所做的假设不适用于 Ariane 5。
- 该组件中失败的功能在Ariane 5中是不需要的，重用软件的验证测试是基于Ariane 5的需求。（因为对失败的功能没有需求，没有开发测试。因此，问题与该软件在发射模拟测试期间从未被发现。）
- <编号>
- 第十七章 软件重用
- 更多描述：问题的原因是一个未处理的异常，当定点数转换为整数时导致数值溢出。这导致运行时系统关闭惯性参考系统，无法保持发射器的稳定性。该故障在 Ariane 4 中从未发生过，因为它的引擎功率较小，并且转换的值不能大到足以使转换溢出。

17.3 组件组成

- 组装组件以创建系统（或另一个组件）的过程。
- 组合涉及将组件相互集成以及与组件基础架构集成。
- 通常，您必须编写“粘合代码”来集成组件。
- <编号>
- 第十七章 软件重用

构图类型

- 顺序组合，其中组合的组件按顺序执行。这涉及组合每个组件的***提供接口***。
- 分层组合，其中一个组件调用另一个组件的服务。一个组件的***提供接口***与另一个组件的***需要接口***组成。
- 添加组合，其中两个组件的接口放在一起以创建一个新组件。提供和需要集成组件的接口是组成组件的接口的组合。
- <编号>
- 第十七章 软件重用

组件组成的类型

- <编号>
- 第十七章 软件重用
- 组件组成的类型
- (a)顺序组合（组合***提供接口***，调用组件 A 提供的服务，然后将 A 返回的结果用于调用组件 B 提供的服务。）
- (b)分层组合（一个组件的***提供接口***与另一个组件的***requires 接口***组成，这两个接口必须**兼容。**如果A 的***requires 接口***和B 的***提供接口不匹配***，则可能需要***额外的代码***。）
- (c) 加法组合（Provides and requires interfaces是组件A和B中对应接口的***组合***。组件通过组合组件的外部接口***单独调用***。A和B不依赖，不相互调用。 ）

接口不兼容

- 参数不兼容，其中操作名称相同但类型不同。
- 操作不兼容，组合接口中的操作名称不同。
- 操作不完整性，其中一个组件的“*提供”接口*是另一个组件的“*需要”接口*的子集。
- 在所有情况下，您都可以通过编写一个适配器来解决不兼容问题，该适配器可以调和正在重用的两个组件的接口。适配器组件将一个接口转换为另一个接口。
- <编号>
- 第十七章 软件重用

接口不兼容的组件

- <编号>
- 第十七章 软件重用
- 为了说明适配器，请考虑下面显示的两个组件，它们的接口不兼容：
- 这些可能是紧急服务（简化）使用的系统的一部分。
- 当紧急接线员接听电话时，将电话号码输入到**addressFinder**组件以定位地址。
- 然后使用**mapper**组件，操作员打印一张地图，发送给被派往紧急情况的车辆。
- 组件**addressFinder**查找与电话号码匹配的地址。所述**映射器**组件需要一个后置码和显示，或在指定尺度打印围绕该代码的区域的街道地图。

适配器组件

- 通过协调组成的组件的接口来解决组件不兼容的问题。
- 根据组合物的类型需要不同类型的适配器。
- “addressFinder”和“mapper”组件可以通过从地址中剥离邮政编码并将其传递给映射器组件的适配器组成。
- <编号>
- 第十七章 软件重用

通过适配器组合

- 组件“postCodeStripper”是促进“addressFinder”和“mapper”组件顺序组合的适配器。
- “postCodeStripper”从“addressFinder”中获取位置数据并**去除**邮政编码。然后将此邮政编码用作“ **mapper** ”的输入，并以 1:10,000 的比例显示街道地图。以下代码是顺序组合的示例，说明了实现这一点所需的调用顺序：
    - 地址= ***addressFinder .location*** (phonenumber);
    - 邮政编码= ***postCodeStripper*** .getPostCode(address);
    - ***映射器***.displayMap(postCode,10000);
- <编号>
- 第十七章 软件重用

连接数据收集器和传感器的适配器

- <编号>
- 第十七章 软件重用
- 下图中适配器的使用，其中适配器用于链接“**数据收集器**”和“**传感器**”组件。这些可用于实施荒野气象站系统。
- “数据收集器”组件设计有一个通用的“需要”接口，支持传感器数据收集和传感器管理。
- 对于这些操作中的每一个，参数是表示特定传感器命令的文本字符串。例如，要发出“收集”命令，您可以说 sensorData(“collect”)。
- 适配器解析输入字符串，识别命令（例如，collect），然后调用 Sensor.getdata() 来收集传感器值。然后将结果（作为字符串）返回到“数据收集器”组件。这种界面风格意味着数据收集器可以与不同类型的传感器进行交互。
- “传感器”本身具有单独的操作，例如“开始”、“停止”和“获取数据”。
- *为每种类型的传感器实施了一个单独的“适配器”，它将传感器命令从“数据收集器”转换为实际的传感器接口。*

光库组合

- <编号>
- 第十七章 软件重用
- 上面对组件组合的讨论假设您可以从组件文档中判断接口是否兼容。
- 当然，接口定义包括操作名称和参数类型，因此您可以从这里对兼容性进行一些评估。

- 为了说明这个问题，请考虑图中所示的组合：
- *这些组件用于实现从数码相机下载图像并将它们存储在照片库中的系统。*
- *照片库组成*

接口语义

- 您必须依靠组件文档来确定在语法上兼容的接口是否实际上兼容。
- 考虑 PhotoLibrary 组件的接口（接口中的方法）：
- public void ***addItem*** （标识符pid；照片p；CatalogEntry photodesc）；
- 公共照片***检索***（标识符pid）；
- public CatalogEntry ***catEntry***（标识符pid）；
- <编号>
- 第十七章 软件重用

照片库文档

- **假设“addItem”方法的文档是：**
- *“此方法将照片添加到库中，并将照片标识符和目录描述符与照片相关联。”*
- **此描述似乎解释了组件的作用，但请考虑以下问题：**
- *“如果照片标识符已经与图书馆中的照片相关联，会发生什么？”*
-
- *“照片描述符是否与目录条目以及照片相关联，即如果我删除照片，我是否也删除目录信息？”*
-
- *'addItem' 的非正式描述中没有足够的信息来回答这些问题。所以一些基于对象约束语言（OCL）的信息被添加到非正式描述中。请参阅下一张幻灯片。*
- <编号>
- 第十七章 软件重用

对象约束语言

- 对象约束语言 (OCL) 旨在定义与 UML 模型相关联的约束。
- 它基于前置和后置条件规范的概念——许多形式方法都很常见。
- OCL 允许您表达必须始终为真的谓词，在方法执行之前必须为真；并且在方法执行后这必须是真的。它们是不变量、前置条件和后置条件。
- 要在操作之前访问变量的值，请在其名称后添加 @pre。因此，以'age'为例：age=age@pre+1
- 此语句意味着操作后 'age' 的值比该操作前的值多 1。
- <编号>
- 第十七章 软件重用

图片库界面的OCL描述

- 以下包括照片库中*addItem*和*delete*方法的规范：

- -- 上下文关键字命名条件适用的组件
- **上下文添加** 项
- -- 前提条件指定在执行 addItem 之前必须为真
- **前**：PhotoLibrary.libSize() > 0
- PhotoLibrary.retrieve(pid) = null
- -- 后置条件指定执行后什么为真
- **后**：libSize () = libSize()@pre + 1
- PhotoLibrary.retrieve(pid) = p
- PhotoLibrary.catEntry(pid) = photodesc
- **上下文** 删除
- **pre** : PhotoLibrary.retrieve(pid) <> null ;
- **帖子**：PhotoLibrary.retrieve(pid) = null
- PhotoLibrary.catEntry(pid) = PhotoLibrary.catEntry(pid)@pre
- PhotoLibrary.libSize() = libSize()@pre-1

- <编号>

- 第十七章 软件重用

- *下一张幻灯片显示了有关条件的更多详细信息。*

照片库条件

- 按照规定，与照片库组件关联的 OCL 声明（“addItem”状态的条件）：
- 先决条件：
    - 图书馆中不得有与要输入的照片具有相同标识符的照片；
    - 库必须存在 - 假设创建库向其中添加了单个项目；
    - 后置条件：
    - 每个新条目将库的大小增加 1；
    - 如果您使用相同的标识符检索，那么您将取回您添加的照片；
    - 如果您使用该标识符查找目录，则会取回您创建的目录条目。

- <编号>

- 第十七章 软件重用

成分权衡

- 在组合组件时，您可能会发现功能性和非功能性需求之间的冲突，以及快速交付和系统演进的需求之间的冲突。
- 您需要做出以下决定：
    - 什么样的组件组合可以有效地满足功能需求?
    - 什么样的组件组成允许未来的变化?
    - 组合系统的涌现特性是什么?

- <编号>

- 第十七章 软件重用

数据收集和报告生成组件

- <编号>
- 第十七章 软件重用
    - 考虑如图所示的情况，其中可以通过两种替代组合创建系统：
    - 适应性和性能之间存在潜在的冲突：
    - 组合物（a）适应性更强；
    - (b) 可能更快更可靠。
    - 组合（a）的***优点***是报告和数据管理是分开的，因此对于未来的变化有更大的灵活性。可以更换数据管理系统，如果需要当前报告组件无法生成的报告，也可以更换该组件而无需更改数据管理组件。
    - 在组合(b) 中，使用了具有内置报告功能（例如Microsoft Access）的数据库组件。
    - 组合（b）的主要***优点***是组件较少，因此这将是一个更快的实现，因为没有组件通信开销。
    - 此外，适用于数据库的数据完整性规则也适用于报告。这些报告将无法以错误的方式组合数据。*在组合（a）中，没有这样的限制，因此报告中可能会出现错误。*
- *一般来说，一个好的组合原则是关注点分离原则。也就是说，您应该尝试以这样一种方式设计您的系统，即每个组件都有明确定义的角色，并且理想情况下，这些角色不应重叠。*

关键点

- 在 CBSE 过程中，需求工程和系统设计的过程是交错的。
- 组件组合是将组件“连接”在一起以创建系统的过程。
- 在组合可重用组件时，您通常必须编写适配器以协调不同的组件接口。
- 在选择组合时，您必须考虑所需的功能、非功能需求和系统演进。

- <编号>
- 第十七章 软件重用