第 16 章 - 软件重用

- 阅读1
- <编号>
- 第16章软件重用

涵盖的主题

- 再利用的前景（好处和问题）
- 应用框架
- 软件产品线
- COTS 产品再利用
- 第16章软件重用
- <编号>

软件重用

- 在大多数工程学科中，系统是通过组合已在其他系统中使用的现有组件来设计的。
- 软件工程更侧重于原始开发，但现在人们认识到，要以更快的速度和更低的成本实现更好的软件，我们需要一个基于系统软件重用的设计过程。
- 在过去的 10 年中，基于重用的开发发生了重大转变。
- 尽管重用作为一种开发策略在 40 多年前就被提出（McIlroy，1968），但直到 2000 年，“重用开发”才成为新业务系统的规范。
- <编号>

基于重用的软件工程

- (1) 应用系统复用
  - 整个应用程序系统可以通过将其合并到其他系统中（COTS 重用）或通过开发应用程序系列来重用。
- (2) 组件复用
  - 可以重用从子系统到单个对象的应用程序组件。包含在第 17 章中。
- (3) 对象和函数重用
  - 可以重用实现单个明确定义的对象或功能的软件组件。
- <编号>

软件重用的好处

益处	解释
提高可靠性	已在工作系统中试用和测试过的重用软件应该比新软件更可靠。它的设计和实现错误应该已经被发现并修复了。
降低流程风险	现有软件的成本是已知的，而开发成本始终是一个判断问题。这是项目管理的一个重要因素，因为它减少了项目成本估算的误差幅度。当重用相对较大的软件组件（例如子系统）时尤其如此。
有效使用专家	应用专家可以开发可重用的软件来封装他们的知识，而不是一遍又一遍地做同样的工作。

- <编号>
- 第16章软件重用

软件重用的好处

益处	解释
符合标准	一些标准，例如用户界面标准，可以实现为一组可重用的组件。例如，如果用户界面中的菜单是使用可重用组件实现的，则所有应用程序向用户呈现相同的菜单格式。标准用户界面的使用提高了可靠性，因为用户在看到熟悉的界面时犯的错误更少。
加速发展	尽早将系统推向市场通常比整体开发成本更重要。重用软件可以加快系统生产，因为开发和验证时间都可以减少。

- <编号>
- 第16章软件重用

重用问题

问题	解释
维护成本增加	如果重用软件系统或组件的源代码不可用，那么维护成本可能会更高，因为系统的重用元素可能越来越不兼容系统更改。
缺乏工具支持	一些软件工具不支持重用开发。将这些工具与组件库系统集成可能很困难或不可能。这些工具所假设的软件过程可能没有考虑重用。对于支持嵌入式系统工程的工具尤其如此，而对于面向对象的开发工具则不然。
非发明综合征	一些软件工程师更喜欢重写组件，因为他们相信他们可以改进它们。这部分与信任有关，部分与编写原始软件被视为比重用其他人的软件更具挑战性的事实有关。

- <编号>
- 第16章软件重用

重用问题

问题	解释
创建、维护和使用组件库	填充可重用的组件库并确保软件开发人员可以使用该库的成本可能很高。必须调整开发过程以确保使用该库。
查找、理解和调整可重用组件	软件组件必须在库中发现、理解，有时还要适应新环境。在将组件搜索作为其正常开发过程的一部分之前，工程师必须有足够的信心在库中找到组件。

- <编号>
- 第16章软件重用
- 必须调整软件开发过程以考虑重用；
- 当软件重用被计划为组织范围重用计划的一部分时，它是最有效的。
    - 日本再利用规划的重要性：Matsumoto，1984 年，日本的“工厂”方法：Cusamano，1989 年；
    - 公司：惠普（HP）；程序：Griss 和 Wosser，1995

16.1 重用前景

- 尽管重用通常被简单地认为是系统组件的重用，但可以使用许多不同的重用方法。
- 可以在从简单的功能到完整的应用系统的一系列级别上进行重用。
- 重用前景涵盖了可能的重用技术范围。
- 重用技术利用了同一应用程序域中的系统相似并具有重用潜力的事实。
- <编号>

再利用景观

- <编号>
- 第16章软件重用
- 在接下来的三张幻灯片中简要介绍了这些重用方法。

支持软件重用的方法

方法	描述
架构模式	支持常见应用系统类型的标准软件架构被用作应用程序的基础。在第 6、13 和 20 章中进行了描述。
设计模式	跨应用程序发生的通用抽象被表示为显示抽象和具体对象和交互的设计模式。在第 7 章中描述。
基于组件的开发	系统是通过集成符合组件模型标准的组件（对象集合）来开发的。在第 17 章中描述。
应用框架	抽象类和具体类的集合被改编和扩展以创建应用程序系统。
遗留系统包装	遗留系统（见第 9 章）通过定义一组接口并通过这些接口提供对这些遗留系统的访问来“包装”。

- <编号>
- 第16章软件重用

支持软件重用的方法

方法	描述
面向服务的系统	系统是通过链接共享服务来开发的，这些服务可能是外部提供的。在第 19 章中描述。
软件产品线	应用程序类型围绕通用架构进行了概括，以便它可以适用于不同的客户。
COTS 产品再利用	系统是通过配置和集成现有应用系统来开发的。
企业资源规划系统	为组织配置封装通用业务功能和规则的大型系统。
可配置的垂直应用	通用系统的设计使其可以根据特定系统客户的需求进行配置。

- <编号>
- 第16章软件重用

支持软件重用的方法

方法	描述
程序库	实现常用抽象的类和函数库可供重用。
模型驱动工程	软件表示为领域模型和实现独立的模型，代码是从这些模型中生成的。在第 5 章中描述。
程序生成器	生成器系统嵌入了一种应用程序的知识，用于从用户提供的系统模型生成该领域中的系统。
面向方面的软件开发	在编译程序时，共享组件会在不同的地方编织到应用程序中。在第 21 章中描述。

- <编号>
- 第16章软件重用

重用规划因素

- 六个关键因素：
    - (1) 软件开发计划。
    - (2) 预期的软件生命周期。
    - (3) 开发团队的背景、技能和经验。
    - (4) 软件的关键性及其非功能性需求。
    - (5)应用领域。
    - (6) 软件的执行平台。例如，.NET 特定于 Microsoft 平台。
- 是否实现重用通常是一个管理问题而不是技术问题。管理人员可能不愿意妥协他们的要求以允许使用可重用的组件。
- 尽管新软件开发的风险可能更高，但一些管理人员可能更喜欢已知风险而非未知风险。
- <编号>

- 第16章软件重用

16.2 应用框架

- 框架是可以重用的中等大小的实体。它们介于系统和组件重用之间。
- 框架是一种子系统设计，由抽象类和具体类的集合以及它们之间的接口组成。
- 该子系统是通过添加组件来填充设计的各个部分并通过实例化框架中的抽象类来实现的。
- <编号>
- 第16章软件重用

框架类

- (1) 系统基础架构
  - 支持通信、用户界面和编译器等系统基础设施的开发。
- (2) 中间件集成框架
  - 支持组件通信和信息交换的标准和类（例如，MS.:.NET、Java: EJB）。
- (3) 企业应用框架
  - 支持特定类型应用程序的开发，例如电信或金融系统。
- <编号>
- 第16章软件重用

网络应用程序框架

- 支持构建动态网站作为Web应用的前端。
- WAF 现在可用于所有常用的 Web 编程语言，例如 Java、Python、Ruby 等。
- 交互模型（WAF 的架构）基于模型-视图-控制器复合模式。
- 第16章软件重用
- <编号>

模型视图控制器

- MVC 模式最初是在 1980 年代提出的，作为一种 GUI 设计方法，它允许一个对象的多个表示以及与这些表示中的每一个的交互样式。
- MVC 框架支持以不同方式呈现数据，并允许与这些呈现中的每一个进行交互。
- MVC 框架涉及到许多模式的实例化（如第 7 章所述），包括：观察者模式、策略模式、组合模式等。
- <编号>

模型-视图-控制器模式

- <编号>

- 第16章软件重用

WAF 功能

- 大多数 Web 应用程序框架都支持以下功能：
- *(1) 安全*。WAF 可能包含帮助实现用户身份验证（登录）和访问的类。
- *(2) 动态网页*。提供了类来帮助您定义网页模板并使用系统数据库中的特定数据动态填充这些模板。
- *(3) 数据库支持*。该框架可以提供为不同数据库提供抽象接口的类。
- *(4) 会话管理*。创建和管理会话（用户与系统的许多交互）的类通常是 WAF 的一部分。
- *(5) 用户交互*。大多数 Web 框架现在都提供 AJAX 支持（Holdener, 2008），这允许创建更多交互式网页。
- 第16章软件重用
- <编号>

扩展框架

- 框架是通用的，并被扩展以创建更具体的应用程序或子系统。它们为系统提供了一个骨架架构。
- 扩展框架涉及
    - 在框架中添加从抽象类继承操作的具体类；
    - 定义为响应框架识别的事件而调用的回调方法。
- 框架的问题在于它们的复杂性，这意味着有效地使用它们需要很长时间。
- <编号>

框架中的“控制反转”

- <编号>
- 第16章软件重用
- “回调”是为响应框架识别的事件而调用的方法。
- 框架对象负责系统中的控制，它们调用“钩子方法”。
- 特定于应用程序的功能以适当的方式响应事件。
- *框架通常比专注于特定应用系统系列的软件产品线更通用。*

关键点

- 大多数新的商业软件系统现在都是通过重用以前实施的系统中的知识和代码来开发的。
- 有许多不同的方法可以重用软件。这些范围从重用库中的类和方法到重用完整的应用程序系统。
- 软件重用的优点是成本更低、软件开发更快、风险更低。系统可靠性增加。通过将专业知识集中在可重复使用组件的设计上，可以更有效地使用专家。
- 应用程序框架是具体和抽象对象的集合，旨在通过专门化和添加新对象来重用。它们通常通过设计模式结合良好的设计实践。
- 第16章软件重用
- <编号>

第 16 章 - 软件重用

- 阅读2
- <编号>
- 第16章软件重用

16.3 软件产品线

- 软件产品线或应用程序系列是具有通用功能的应用程序，可以进行调整和配置以在特定环境中使用。
- 软件产品线是一组具有通用架构和共享组件的应用程序，每个应用程序专门用于反映不同的需求。
- 适应可能涉及：
    - 组件和系统配置；
    - 向系统添加新组件；
    - 从现有组件库中选择；
    - 修改组件以满足新的要求。
- <编号>

应用框架和产品线

- 应用程序框架依靠*面向对象的*特性（例如多态）来实现扩展。产品线不需要面向对象（例如移动电话的嵌入式软件）
- 应用程序框架专注于提供技术支持，而不是特定领域的支持。产品线嵌入域和平台信息。
- 产品线通常控制设备的应用。
- 软件产品线由通常由同一组织拥有的一系列应用程序组成。
- 第16章软件重用
- <编号>

产品线专业化

- 平台专业化
    - 针对不同平台开发了不同版本的应用程序。例如，可能存在用于 Windows、Mac OS 和 Linux 平台的应用程序版本。
- 环境专业
    - 创建不同版本的应用程序以处理不同的操作环境，例如不同类型的通信设备。例如，根据车辆通信系统的不同，紧急服务系统可能存在不同的版本。
- 功能专业化
    - 为具有不同要求的客户创建了不同版本的应用程序。例如，图书馆自动化系统可以根据它是在*公共*图书馆、*参考*图书馆还是*大学*图书馆中使用而被修改。
- 工艺专业化

    - 创建不同版本的应用程序以支持不同的业务流程。例如，订购系统可以是集中式的或分布式的。
- <编号>

产品线架构

- 架构必须以这样一种方式构建，以分离不同的子系统并允许修改它们。
- 体系结构还应该将实体及其描述分开，系统中的更高级别通过描述而不是直接访问实体。
- <编号>

资源分配系统的架构

- <编号>
- 第16章软件重用
- 这个四层结构在下一张幻灯片中实例化。

车辆调度员的产品线架构

- <编号>
- 第16章软件重用
- 此图显示了可能包含在车辆调度系统产品线中的模块。

车辆调度

- 一个专门的资源管理系统，其目的是分配资源（车辆）来处理事件。
- 改编（各个层次的组成部分）包括：
    - 在 UI 层，有用于操作员显示和通信的组件；
    - 在 I/O 管理级别，有处理身份验证、报告和路由规划的组件；
    - 在资源管理层面，有车辆定位和调度、管理车辆状态和事件记录等组件；
    - 在数据库层面，除了通常的事务管理支持外，还有设备、车辆和地图的独立数据库。
- <编号>

产品实例开发

- <编号>
- 第16章软件重用
- 扩展软件产品线以创建新应用程序所涉及的步骤，下一张幻灯片中有更多详细信息。

产品实例开发

- (1) 引出利益相关者的要求

    - 使用现有的家庭成员作为原型。
- (2) 选择最合适的家庭成员
    - 找到最符合要求的家庭成员。
- （三）重新协商要求
    - 根据软件功能的需要调整需求。
- (4) 适应现有系统
    - 开发新模块并为家庭成员进行更改。
- (5) 交付新的家庭成员
    - 产品线的新实例交付给客户。您应该记录关键功能以进一步发展会员。
- <编号>

产品线配置

- 软件产品线旨在重新配置，这种重新配置可能涉及从系统中添加或删除组件、定义系统组件的参数和约束，以及包括业务流程的知识。
- 这种配置可能发生在开发过程的不同阶段：
- (1) 设计时配置
    - 产品线根据特定客户的要求进行调整和更改。
- (2) 部署时间配置
    - 产品线是通过嵌入客户需求和业务流程的知识来配置的。软件源代码本身没有改变。
- <编号>

部署时配置

- <编号>
- 第16章软件重用
- 部署时配置涉及使用配置工具创建特定的系统配置，该配置记录在配置数据库中或作为一组配置文件。
- 执行系统在执行时查询该数据库，以便其功能可以专门用于其执行上下文。

部署时间配置的级别

- 系统中可以提供多个级别的部署时配置：
- (1) 组件选择，您可以在其中选择系统中提供所需功能的模块。
- (2) 工作流和规则定义，您可以在其中定义工作流（如何处理信息，分阶段）和验证规则，这些规则应应用于用户输入或系统生成的信息。
- (3) 参数定义，您可以在其中指定反映您正在创建的应用程序实例的特定系统参数的值。
- 第16章软件重用

- <编号>

16.4 COTS 产品再利用

- 商用现货（COTS）产品是一种软件系统，可以在不改变系统源代码的情况下适应不同的客户。
- COTS 系统具有通用功能，因此可以在不同环境中使用/重用。
- COTS 产品通过使用内置配置机制进行调整，允许根据特定客户需求定制系统功能。
    - 例如，在医院患者记录系统中，可能会为不同类型的患者定义单独的输入表单和输出报告。
- 第16章软件重用
- <编号>

COTS 重用的好处

- 在过去的 15 年左右的时间里，这种软件重用方法已被大公司广泛采用，因为它比定制软件开发具有显着的优势：
- (1) 与其他类型的重用一样，可以更快速地部署可靠系统。
- (2) 可以看到应用程序提供了什么功能，因此更容易判断它们是否可能合适。
- (3)利用现有软件避免了一些开发风险。然而，这种方法有其自身的风险，我将在下面讨论。
- (4) 企业可以专注于他们的核心活动，而不必将大量资源用于 IT 系统开发。
- (5) 随着操作平台的发展，技术更新可能会被简化，因为这些是 COTS 产品供应商而不是客户的责任。
- 第16章软件重用
- <编号>

COTS重用问题

- 当然，这种软件工程的方法有其自身的问题：
- (1) 通常必须调整需求以反映 COTS 产品的功能和操作模式。这可能会导致对现有业务流程的破坏性更改。
- (2) COTS 产品可能基于几乎不可能改变的假设。因此，客户必须调整其业务以反映这些假设。
- (3) 为企业选择合适的 COTS 系统可能是一个困难的过程，尤其是因为许多 COTS 产品没有很好的文档记录。
- (4) 可能缺乏支持系统开发的本地专业知识。
- (5) COTS 产品供应商控制系统支持和演进。
- 第16章软件重用
- <编号>

COTS 解决方案和 COTS 集成系统

COTS-解决方案系统	COTS 集成系统
提供客户所需功能的单一产品	集成多种异构系统产品，提供定制化功能
基于通用解决方案和标准化流程	可以为客户流程开发灵活的解决方案
开发重点是系统配置	开发重点是系统集成
系统厂商负责维护	系统所有者负责维护
系统供应商为系统提供平台	系统所有者为系统提供平台

- <编号>
- 第16章软件重用
- COTS 产品重用有两种类型，即 COTS 解决方案系统和 COTS 集成系统：

16.4.1 COTS 解决方案系统

- COTS 解决方案系统是通用应用系统，可以设计为支持特定的业务类型、业务活动，或者有时是一个完整的企业。
    - 例如，可以为牙医生产 COTS 解决方案系统，用于处理预约、牙科记录、患者召回等。
- 特定领域的 COTS 解决方案系统，例如支持业务功能（例如文档管理）的系统，提供了一系列潜在用户可能需要的功能。
    - 例如，支持大学学生注册的系统可能假设学生将在一所大学注册一个学位。但是，如果大学合作提供联合学位，那么实际上不可能在系统中体现这一点。
- 第16章软件重用
- <编号>

企业资源规划系统

- 企业资源计划 (ERP) 系统是一个通用系统，支持常见的业务流程，例如订购和开票、库存管理和制造调度等。
- 这些在大公司中被非常广泛地使用——它们可能代表了最常见的软件重用形式。
- 通用核心通过包含模块和合并业务流程和规则的知识进行调整。
- <编号>

ERP系统的架构

- <编号>
- 第16章软件重用
    - 支持一系列业务功能的 ERP 系统的整体架构模型如下所示：

ERP架构

- 上述ERP架构的主要特点是：
- (1) 多个模块，支持不同的业务功能。
- (2) 一组已定义的业务流程，与每个模块相关联，并与该模块中的活动相关。
- (3) 一个公共数据库，用于维护有关所有相关业务功能的信息。
- (4) 一套适用于数据库中所有数据的业务规则。
- 第16章软件重用
- <编号>

ERP配置

- 从系统中选择所需的功能。
- 建立一个数据模型，定义组织的数据在系统数据库中的结构。
- 定义适用于该数据的业务规则。
- 定义与外部系统的预期交互。
- 设计系统生成的输入表单和输出报告。
- 设计符合系统支持的底层流程模型的新业务流程。
- 设置定义系统在其底层平台上部署方式的参数。
- 第16章软件重用
- <编号>

16.4.2 COTS 集成系统

- COTS 集成系统是包含两个或多个 COTS 产品和/或遗留应用程序系统的应用程序。
- 当没有单一的 COTS 系统满足您的所有需求，或者您希望将新的 COTS 产品与您已经使用的系统集成时，您可以使用这种方法。
- 第16章软件重用
- <编号>

设计选择

- 要使用 COTS 产品开发系统，您必须做出许多设计选择：
- (1) 哪些 COTS 产品提供最合适的功能?
    - 通常，会有几种可用的 COTS 产品，它们可以以不同的方式组合（经验很重要）。
- (2) 数据将如何交换?
    - 不同的产品通常使用独特的数据结构和格式。您必须编写从一种表示转换为另一种表示的适配器。
- (3) 产品的哪些特性会被实际使用?
    - COTS 产品可能包含比您需要的更多的功能，并且功能可能会在不同的产品中重复。
- 第16章软件重用
- <编号>

    - 如果可能，您还应该拒绝访问未使用的功能，因为这会干扰正常的系统操作。阿丽亚娜 5 号火箭，1997）

- 阿丽亚娜 5 号火箭（Nuseibeh，1997 年）的首飞失败是由阿丽亚娜 4 号系统重新使用的惯性导航系统故障造成的。但是，在 Ariane 5 中实际上并不需要失败的功能。

一个 COTS 集成采购系统

- <编号>
- 第16章软件重用

面向服务的 COTS 接口

- 如果使用面向服务的方法，可以简化 COTS 集成。
- 面向服务的方法意味着允许通过标准服务接口访问应用系统的功能，为每个离散的功能单元提供服务。
- 某些应用程序可能会提供服务接口，但有时，该服务接口必须由系统集成商来实现。您必须编写一个包装器来隐藏应用程序并提供外部可见的服务（下一张幻灯片中的图）。
- 第16章软件重用
- <编号>

应用包装

- <编号>
- 第16章软件重用

COTS系统集成问题

- 四个重要的 COTS 系统集成问题：
- (1) 缺乏对功能和性能的控制
    - COTS 系统可能没有看起来那么有效
- (2) COTS系统互操作性问题
    - 不同的 COTS 系统可能会做出不同的假设，这意味着集成很困难
- (3) 无法控制系统演化
    - COTS 供应商不是系统用户控制演进
- (4) 来自 COTS 供应商的支持
    - COTS 供应商可能不会在产品的整个生命周期内提供支持
- <编号>

关键点

- 软件产品线是从公共基础开发的相关应用程序。该通用系统适用于满足功能、目标平台或操作配置的特定要求。
- COTS 产品重用涉及大规模、现成系统的重用。这些提供了许多功能，它们的重用可以从根本上降低成本和开发时间。系统可以通过配置单个通用 COTS 产品或通过集成两个或多个 COTS 产品来开发。
- 企业资源规划系统是大规模 COTS 重用的例子。您可以通过使用有关客户业务流程和规则的信息配置通用系统来创建 ERP 系统的实例。
- 基于 COTS 的重用的潜在问题包括缺乏对功能和性能的控制、缺乏对系统演化的控制、需要外部供应商的支持以及难以确保系统可以互操作。

- 第16章软件重用

- <编号>