软件构件仓库 SCoRe 的体系结构

叶 伟 钱乐秋

(复旦大学计算机科学系 上海 200433)

摘 要 软件构件重用已成为软件工程的新研究方向，构件标准、构件描述语言 CDL、构件仓库是其主要研究课题，而构件仓库更是其基础。本文提出了一个软件构件仓库 SCoRe 的体系结构及其相关工具；研究了构件仓库对软件重用的支持。SCoRe 用 I^2R 模型表示构件，将构件存储在构件库中，并利用构件库访问工具 CRTs 对其进行存取、剪裁/组合，得到适合应用的新构件(或其实例)。利用构件操作工具 CTs 在构件的内部表示与其源代码表示和 CDL 文本描述之间进行相互转换。

关键词 软件构件，重用，构件仓库，构件描述语言 CDL，CASE 集成环境。

THE ARCHITECTURE OF THE SCoRe ——A SOFTWARE COMPONENT REPOSITORY

YE WEI QIAN LEQIU

(*Department of Computer Science, Fudan University, Shanghai* 200433)

Abstract Software component reuse has been one of the new directions in software engineering, whose main subjects involves component standardization, Component Description Language (CDL), and component repository as the base of the all.

In this paper, we propose the architecture of the SCoRe——A Software Component Repository, and its two related tool sets. Software reuse supported by SCoRe is also discussed.

In SCoRe, a component is presented by I^2R model, and stored in repository, tailored and composed to a new component (or its instance) suitable for the application using Component Repository Tools (CRTs), translated among various forms: inner representation, source codes, CDL text description through Component Tools (CTs).

Keywords Software component, reuse, component repository, CDL (Component Description Language), CASE.

本文 1996 年 11 月 14 日收到。本文得到国家九五攻关项目“青鸟 III 集成 CASE 环境”的资助。叶伟，硕士生，钱乐秋，教授，现均从事软件工程的研究。

一、引　言

OO 应用开发一般分为四个阶段：需求分析、系统设计(规格说明)、详细设计和实现。毫无疑问，在开发的各个阶段，尤其是后期阶段(详细设计、编码)中，使用重用技术可以减少大量工作。当前，软件构件重用是重用技术的一种主要的方式，它包括两个层次：

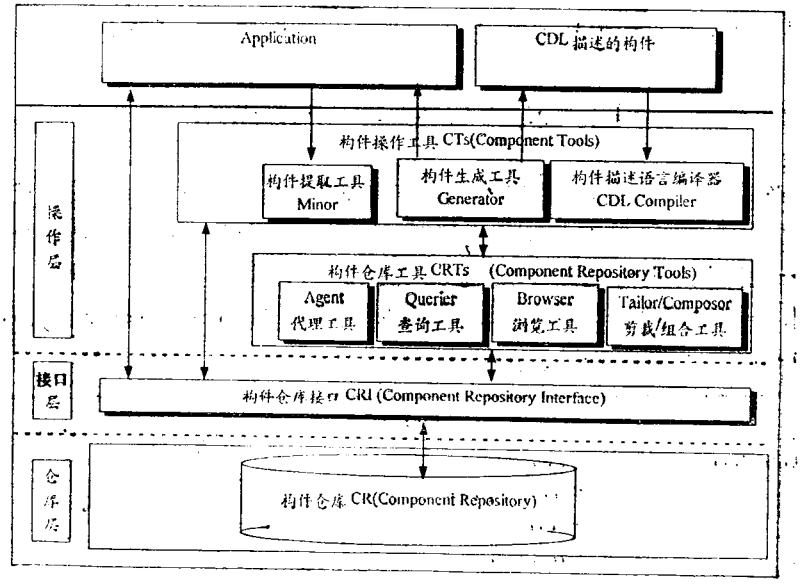
- 复制、继承已有构件类，产生新的构件实例；
- 修改、剪裁、组合已有构件类，生成新的构件类。

构件重用已成为软件工程的新研究方向。为了有效地重用构件，需要收集各种零星分布于各种应用、类库中的可重用构件，并为其定义统一的接口，集中存放，高效管理，方便使用。构件仓库由此应运而生。

SCoRe 系统的目标是建立一个软件构件仓库及配套的多个工具，作为集成 CASE 环境的一个支撑基础，以便在 OO 应用的所有开发阶段中实施软件重用。

SCoRe 的核心是一个构件仓库，其中存储了大量的已完成的构件，这些构件或来自各种现成类库，或自开发过的各个应用系统中提取、剪裁、组合而成。构件以一定的组织结构存放在库中，重用者可利用构件选取工具(SRT/Selector)，从构件库中选取合适的可重用构件，对其进行剪裁、组合后，用于构造新的应用。

二、SCoRe 体系结构



上图是 SCoRe 的体系结构，它由三个层次组成，由下到上分别为：

- 仓库层——构件仓库 CR，可重用软件构件即存于其中；
- 接口层——构件库接口 CRI，它提供对 CR 的访问接口，所有对构件库的访问操作均需通过 CRI 完成，以保持统一性与一致性；
- 操作层——由两类工具组成：构件库工具 CRT 和构件操作工具 CT。CRT 提供一

组工具，用于帮助重用者访问构件库，包括选择工具CRT/Selector(由查询工具CRT/Querier和浏览工具CRT/Browser组成)，剪裁/组合工具CRT/Tailor&Composor，代理工具CRT/Agent。CT提供一组工具，用于帮助重用者操纵单个构件，完成构件的内部表示与其它表示(如C++代码、CDL描述)之间的转换，包括构件提取工具(从C++代码中抽取出构件，并转换成内部表示)，构件生成工具(将内部表示转换成C++代码)和CT/CDLCompiler(CDL编译器将CDL文本描述的构件编译后，转换成内部表示)。构件创建者可利用CT从应用程序中提取出构件，或从CDL文本描述得到构件，然后通过CRT的代理工具存入CR。CT既可以通过CRT访问CRI，也可以直接调用CRI的API，从而访问CR。

为了保持构件库的实现无关性，外部只能通过CRI访问CR。CRI提供了统一的接口，使得应用系统和操作层工具无需关心CR的实现方式就可以通过它存取构件。

三、构件仓库CR

构件仓库CR是SCoRe的核心部分，允许重用者用构件描述语言CDL描述软件构件，并存储在构件库中作长期使用；允许重用者通过构件库接口CRI统一管理/操纵构件库。同时，它也提供了一组建立在CRI基础上的构件仓库操作工具CRTs，帮助重用者方便地访问构件仓库。

构件是能完成一定功能的独立软件对象。通常，单个类就是一个最基本的构件。但单个类能完成的功能是有限的，有时为了构成一个功能更强大、目标更明确的构件，需由多个相关的类(或其实例)及相关信息组成，这就形成了更大的构件或框架。

四、构件模型——I^2R模型

在面向对象应用中，构件通常模型化为带有(数据)结构和行为属性的功能体。然而，我们需要一个更为丰富的表示，允许重用者据其属性和与外部信息的关系选择构件。据此，我们使用了I^2R模型作为规格说明的模型。

I^2R模型由一个可重用软件构件的三个不同方面组成：Interface(接口)、Implementation(实现)、Relationship(关系)。如下表所示：

Component		
Interface	Name	
	Specifications	
	Properties	
	Operations	
Implementation	DeclareFile	
	CodeFiles	
	SourceCode	
Relationship	InterfaceRelationship	Bases, Parts
	OperationRelationship	Roles, PartOf
	ImplementRelationship	Uses

● Interface 是一个构件的抽象描述。重用者可根据构件的接口理解其功能。一个构件的接口包含其名称、类型、(功能)规格说明、数据结构(属性)、操作等。

● Implementation 是一个构件功能的实现，也即构件实际内容的细节。除非重用者想改变一个构件的功能，否则并不需要知道构件的实现。一个构件的实现既可以直接包含源代码，也可以只描述其声明和实现的文件名。

● Relationship 是可重用构件与其所依赖的其它构件之间的关系。例如，构件 A 使用了另一构件 B 中声明的某个资源、某个类型或某个过程，我们就说构件 A 依赖于构件 B，或者说构件 A 的依赖集包括 B。I^2R 模型中的关系还可进一步定义。某个构件的 Interface 与其依赖的其它构件的 Interface 间的关系称为接口性关系，如基类(Bases)、组成部件(Parts)等；某个构件 A 操作的数据结构被其它构件所定义、包含，则 A 与这些构件间的关系称为操作性关系，如角色(Roles)、部件(PartOf)；某个构件的实现与其依赖的其它构件的 Interface 间的关系称为实现性关系，如使用(Uses)。

实际存放在构件库中的构件除了上述三部分之外，还包含了一些用于维护的信息，如下所示：

Component	
*ComponentID ComponentName	
Info	ComponentType Version Creator CreateDate LastModifiedDate AccessTimes
Interface	
Implementation	
Relationship	

其中，ComponentID 是构件的唯一标识，用于区分不同的构件。Info 域中有多个与构件维护相关的字段，包括版本信息、创建信息、修改信息、访问次数等，可以利用这些信息来进行维护，不断完善构件库。

五、仓库存储系统

仓库存储系统可以采用多种方式：面向对象数据库、关系数据库和文件。

5.1 面向对象数据库

由于我们的构件具有面向对象的特性，用面向对象数据库来存储是再好不过了，既有数据库的数据容量大、易于维护，又具有面向对象的各种优越性，易于扩充、易于数据格式动态变动。可惜，OODBMS 尚无流行的现成产品可用。但是不久的将来，它必将成为构件仓库的主要存储系统。

5.2 关系数据库

关系数据库用于存储大量数据时是非常有效的，而且其理论基础严密，技术上也相当成熟。然而，构件仓库本身存储的是构件，其数据变动性较大，要求易于扩充。采用 RDBMS 需要作仔细设计。

5.3 文件系统

文件系统虽然不利于存储大量数据，特别是当构件库日益庞大后并不适用。但它极为灵活，完全可以体现面向对象的特性，尤其是作为研究用的小型构件库，可以迅速建立起原型系统。事实上，Borland 公司最新旗舰软件 Delphi 2.0 的对象仓库(ObjRepository)正是用文件系统实现的，虽然比较简单，却对软件重用起到了很好的实用效果。

六、构件库接口 CRI——操作 API 集

构件库接口 CRI 是一个独立部分，它是一个构件库的操作 API 集。CRI 可以同时支持不同的构件库存储体系，正由于 CRI 层的存在，构件库操作工具 CRTs 和应用程序可以透明地访问构件库。CRI 主要提供以下几类 API 操作：

- 查询/选择：重用者可以通过多种方式检索构件库，例如：构件名称、构件属性值、关键词等特征值，或者 IsA、PartOf、Uses 等构件间关系；
- 加入/删除：将一个构件加入到构件库中，或将构件从构件库中删除；
- 替换：用构件的新版本替换构件库中已存在的版本；
- 更新/修改：更改构件库中指定构件的某些特征值或构件间关系；
- 库合并/分解：将两个构件库合并，或将某个构件库按给出的标准(比如版本号、作者)分解成多个小库。

七、构件库工具 CRTs

CRTs 是建立在 CRI 基础上的一组构件库操作工具，主要目标是辅助重用者：

- 从构件库 CR 中选取适当的构件；
- 对已有构件进行组合和剪裁，形成新的构件。

构件库的普通操作，如加入/获取/删除/修改构件，构件版本更新等，可由代理工具 CRT/Agent 完成。

7.1 选取工具 CRT/Selector

由于应用的需求是各不相同的，因而从构件库中找到一个完全适合的构件是比较困难的。重用者使用一个选择窗口，搜索构件仓库，通过该窗口可浏览仓库的内容，或从中查询与给出条件相匹配的构件。CRT/Selector 浏览或查询仓库是智能渐增式的，随着给出的条件逐步详细、精确，在知识库的辅助下，不断缩小搜索范围，最后找到所需的构件。CRT/Selector 由两个子工具组成：浏览工具 CRT/Browser 和查询工具 CRT/Querier。

7.1.1 浏览工具 CRT/Browser 浏览工具 CRT/Browser 提供一个仓库的超文本类型的图形接口，给出一个初始构件的名称，它就能通过以下链接显示与选中构件相关的其它构件：

结构性链接：表述了面向对象的典型关系(IsA, PartOf 及 MessageExchange 等)；

功能性链接：来自构件中的功能描述和行为属性。

这些链接允许浏览者逐步细化对构件的描述，并将它们映射到更详细、更明确的构件，或将它们链接到设计构件中。

当重用者选择了一种链接类型时，CRT 就显示相应的子网，历史窗口提供所遍历过的构件的导航线路。若重用者希望跳到以前查看(遍历)过的构件，就可在历史列表中选择。CRT 可标记浏览过的构件，或将之存入一个构件池，以备再次查询和浏览。

7.1.2 查询工具 CRT/Querier 查询是基于构件的特征和构件间的关系的，这些关系来自面向对象的角色链接 (IsA, PartOf, Role) 和方法链接(类间通信)。形成一个查询需包括构件属性(名称、角色、属性、消息)及构件间关系作为参数。一个查询有两部分：构件属性部分和基于功能描述的部分。

7.2 组合与剪裁

由于选取的构件不一定完全符合应用的需求，因而还需对其进行修改后方能使用。

组合与剪裁是一个自顶向下和自底向上穿插进行的过程。CRT 在一给定的抽象层次(系统设计层、详细设计层、实现层)上辅助重用者组合与剪裁，对一个构件的细化(自顶向下开发)和抽象(自底向上开发)提供帮助。它也维护不同抽象层次上构件规格说明间的映射，以检查一致性。自顶向下细化可将一个构件分解为子构件，子构件提供同样的功能服务，但在更详细的层次上进行描述。自底向上开发是典型的重用，允许重用者取出一批相关的构件。为提供有关这批构件的功能的更抽象的描述，CRT 可提供一个更抽象的规格说明，描述这批构件作为一个整体的功能。

重用者可根据需要通过 CRT/Selector 从 CR 中选取出多个构件，组合在一起；或对某一构件裁去不需要的部分，构成新的构件。新的构件既可以直接用于应用中，也可以存入构件库中，扩充/完善构件库。

八、构件操作工具 CT

CT 用于帮助重用者操纵单个构件，完成构件的内部表示与其它表示(如 C++ 代码、CDL 描述)之间的转换，包括构件提取工具、构件生成工具和 CT/CDL Compiler。

8.1 构件提取工具 CT/Minor

CT/Minor 从源代码中抽取出构件，并转换成内部表示。源代码应是 C++ 或 Object Pascal 等面向对象语言。构件创建者可以将源代码作为 CT/Minor 的输入，得到初始构件，再使用 CRT 工具对其进行修改后，加入到构件库中。

8.2 构件生成工具 CT/Generator

CT/Generator 恰好与 CT/Minor 相反，用于将构件从内部表示转换成 C++ 或 Object Pascal 源代码，或者 CDL 描述文本。重用者利用 CRT/Selector 从构件库中选取适合当前应用需求的可重用构件，用 CRT/Tailor&Composor 修改之后，利用 CT/Generator 将之转换成相应的源代码，或者转换成 CDL 描述文本。

8.3 CDL 编译器——CT/CDL Compiler

CT/CDL Compiler 完成构件描述语言 (CDL) 书写的构件与内部表示之间的相互转

换。构件创建者可以直接用 CDL 语言书写构件，然后用 CDL Compiler 对之进行编译，产生内部表示后，再加入构件库。

CDL 语言本身尚在开发之中，目前尚无统一标准，我们也在探索之中。

九、总　结

SCoRe 系统尚在研究开发之中，本文对其基本体系结构进行了描述，也对构件的表示模型、构件库的结构组织、构件库的操作工具、构件的提取/生成、CDL 语言描述等作了简略表述。SCoRe 本身还有许多方面需要作进一步研究，如：构件库对框架的支持；CDL 语言设计及 CDL Compiler 的开发；CRI 的扩充；CRTs 和 CTs 的完善、强化与新工具开发等。但是，我们也已感受到如何充分利用构件库进行软件重用，提高软件生产率和软件质量，同样是迫切而重要的，主要的研究课题有：构件库对 OO 开发各阶段（尤其是前面阶段）支持的研究及相关工具开发；构件库支持下 CASE 环境的开发；构件提取。

参 考 文 献

[1] Rucger, "Software Reuse", ACM Computing Surveys, pp. 131～183, June 1992.

[2] Faustle and M. Fugini, "Retrieval of Reusable Components in a Development Information System", Proc. Workshop on Software Reuse, IEEE CS Press, Los Alamitos, Calif., 1993.

[3] Bellinzona, M. Fugini and V. de Mey, "Reuse of Specifications and Designs in a Development Information System", in Information System Development Process, N. Prakash, C. Rolland and B. Pernici,eds., North-Holland, Amsterdam, 1993.

[4] Borland International Corp. «Delphi User's Manual».