软件复用与软件构件技术*

杨芙清 梅 宏 李克勤
(北京大学计算机科学技术系, 北京 100871)

【提要】 软件复用是在软件开发中避免重复劳动的解决方案. 通过软件复用, 可以提高软件开发的效率和质量. 近十几年来, 面向对象技术出现并逐步成为主流技术, 为软件复用提供了基本的技术支持. 软件复用研究重新成为热点, 被视为解决软件危机, 提高软件生产效率和质量的现实可行的途径. 它通常可分为产品复用和过程复用两条途径. 基于构件的复用是产品复用的主要形式, 也是当前复用研究的焦点. 同时, 在分布对象研究领域, 软件构件技术也是一个重要内容. 当前软件构件技术被视为实现成功复用的关键因素之一. 软件复用技术的广泛应用将促进软件产业的变革, 使其重组分工, 软件构件生产必将成为独立的行业而存在. 这种变革对我国软件产业的发展将是一个很好的机遇. 本文将对软件复用技术的发展作一全面综述, 介绍软件复用的基本概念及关键技术, 同时介绍国内外在软件复用方面的成功的研究和实践活动, 包括我国政府支持的国家重点科技攻关项目青鸟工程, 并对我国如何加强相关技术研究和应用, 推动软件产业发展提出一些思路.

关键词: 软件工程, 软件复用, 软件构件技术, 计算机辅助软件工程 (CASE)

Software Reuse and Software Component Technology

Yang Fuqing, Mei Hong, Li Keqin
(Dept. of Computer Science & Technology, Peking University, Beijing 100871)

Abstract: Software reuse offers a solution to eliminate repeated work and improve efficiency and quality in the software development. In the recent ten years, object-oriented technology has appeared and become a mainstream technology, thereby providing fundamental technology support for software reuse. Software reuse regains more attention in software engineering research and is considered a pratical and feasible approach to solving the software crisis. Software reuse is generally classified into two catalogues: product reuse and process reuse. Reuse based on software components is the important form of product reuse and is the major area of software reuse research. At the same time, software component technology plays an important role in distributed object research. Therefore, software component technology is regarded as a key factor of successful software reuse. The development and application of software reuse technology will facilitate the revolution of software development and reorganize software industry. As a result, the development of software components will become an independent and inseparable industry. The revolution offers a good chance for Chinese software development. This paper is a summarization on the development of software reuse technology. It presents fundamental concepts and key techniques of software reuse. After introducing several successful research and practice in software reuse, including Jade Bird Project, a Chinese national key project supported by the government, it proposes some ideas on how to reinforce research and application of related techniques and facilitate the development of software industry in China.

Key words: Software Engineering; Software Reuse; Software Component Technology; CASE

一、引 言

1. 为什么要复用

通常情况下, 应用软件系统的开发过程包含以下几个阶段: 需求分析、设计、编码、测试、维护等. 当每个应用系统的开发都是从头开始时, 在系统开发过程中就必然存在大量的重复劳动, 如: 用户需求获取的重复、需求分析和设计的重复、编码的重复、测试的重复和文档工作的重复等.

探讨应用系统的本质, 可以发现其中通常包含三类成分: ①通用基本构件: 是特定于计算机系统的构成成分, 如基本的数据结构、用户界面元素等, 它们可以存在于各种应用系统中; ②领域共性构件: 是应用系统所属领域的共性构成成分, 它们存在于该领域的各个应用系统中; ③应用专用构件: 是每个应用系统的特有构成成分.

应用系统开发中的重复劳动主要在于前两类构成成分的重复开发.

* 1998年6月收到, 1998年10月修改定稿. 国家"九五"科技攻关项目及863高技术计划资助课题

软件复用是在软件开发中避免重复劳动的解决方案, 其出发点是应用系统的开发不再采用一切"从零开始"的模式, 而是以已有的工作为基础, 充分利用过去应用系统开发中积累的知识和经验, 如: 需求分析结果、设计方案、源代码、测试计划及测试案例等, 从而将开发的重点集中于应用的特有构成成分.

通过软件复用, 在应用系统开发中可以充分地利用已有的开发成果, 消除了包括分析、设计、编码、测试等在内的许多重复劳动, 从而提高了软件开发的效率, 同时, 通过复用高质量的已有开发成果, 避免了重新开发可能引入的错误, 从而提高了软件的质量.

2 复用的基本概念

软件复用是指重复使用"为了复用目的而设计的软件"的过程[22]. 相应地, 可复用软件是指为了复用目的而设计的软件. 与软件复用的概念相关, 重复使用软件的行为还可能是重复使用"并非为了复用目的而设计的软件"的过程, 或在一个应用系统的不同版本间重复使用代码的过程, 这两类行为都不属于严格意义上的软件复用.

以下的类比有助于进一步说明软件复用的概念. 在软件演化的过程中, 重复使用的行为可能发生在三个维上:

(1)时间维: 使用以前的软件版本作为新版本的基础, 加入新功能, 适应新需求, 即软件维护.

(2)平台维: 以某平台上的软件为基础, 修改其和运行平台相关的部分, 使其运行于新平台, 即软件移植.

(3)应用维: 将某软件(或其中构件)用于其他应用系统中, 新系统具有不同功能和用途, 即真正的软件复用.

这三种行为中都重复使用了现有的软件, 但是, 真正的复用是为了支持软件在应用维的演化, 使用"为复用而开发的软件(构件)"来更快、更好地开发新的应用系统.

复用概念的第一次引入是在 1968 年 NATO 软件工程会议上, McIlroy 的论文"大量生产的软件构件"中. 在此以前, 子程序的概念也体现了复用的思想. 但其目的是为了节省当时昂贵的机器内存资源, 并不是为了节省开发软件所需的人力资源. 然而子程序的概念可以用于节省人力资源的目的, 从而出现了通用子程序库, 供程序员在编程时使用. 例如, 数学程序库就是非常成功的子程序复用的例子.

在其后的发展过程中, 有许多复用技术的研究成果和成功的复用实践活动. 但是, 复用技术在整体上对软件产业的影响却并不尽如人意. 这是由于技术方面和非技术方面的种种因素造成的, 其中技术上的不成熟是一个主要原因. 近十几年来, 面向对象技术出现并逐步成为主流技术, 为软件复用提供了基本的技术支持. 软件复用研究重新成为热点, 被视为解决软件危机, 提高软件生产效率和质量的现实可行的途径[12].

分析传统产业的发展, 其基本模式均是符合标准的零部件(构件)生产以及基于标准构件的产品生产(组装), 其中, 构件是核心和基础, "复用"是必需的手段. 实践表明, 这种模式是产业工程化、工业化的必由之路. 标准零部件生产业的独立存在和发展是产业形成规模经济的前提. 机械、建筑等传统行业以及年轻的计算机硬件产业的成功发展均是基于这种模式并充分证明了这种模式的可行性和正确性. 这种模式是软件产业发展的良好借鉴, 软件产业要发展并形成规模经济, 标准构件的生产和构件的复用是关键因素. 这正是软件复用受到高度重视的根本原因.

软件复用可以从多个角度进行考察[18]. 依据复用的对象, 可以将软件复用分为产品复用和过程复用. 产品复用指复用已有的软件构件, 通过构件集成(组装)得到新系统. 过程复用指复用已有的软件开发过程, 使用可复用的应用生成器来自动或半自动地生成所需系统. 过程复用依赖于软件自动化技术的发展, 目前只适用于一些特殊的应用领域. 产品复用是目前现实的、主流的途径.

依据对可复用信息进行复用的方式, 可以将软件复用区分为黑盒(Black-box)复用和白盒(White-box)复用. 黑盒复用指对已有构件不需作任何修改, 直接进行复用. 这是理想的复用方式. 白盒复用指已有构件并不能完全符合用户需求, 需要根据用户需求进行适应性修改后才可使用. 而在大多数应用的组装过程中, 构件的适应性修改是必需的.

3. 如何实现复用

软件复用有三个基本问题, 一是必须有可以复用的对象; 二是所复用的对象必须是有用的, 三是复用者需要知道如何去使用被复用的对象. 软件复用包括两个相关过程: 可复用软件(构件)的开发(Development for Reuse)和基于可复用软件(构件)的应用系统构造(集成和组装)(Development with Reuse). 解决好这几个方面的问题才能实现真正成功的软件复用.

与以上几个方面的问题相联系, 实现软件复用的关键因素(技术和非技术因素)主要包括: 软件构件技术(Software Component Technology)、领域工程(Domain Engineering)、软件构架(Software Architecture)、软件再工程(Software Reengineering)、开放系统(Open System)、软件过程(Software Process)、CASE 技术等以及各种非技术因素.

二、实现软件复用的关键因素

实现软件复用的各种技术因素和非技术因素是互相联系的. 如图 1 所示, 它们结合在一起, 共同影响软件复用的实现.

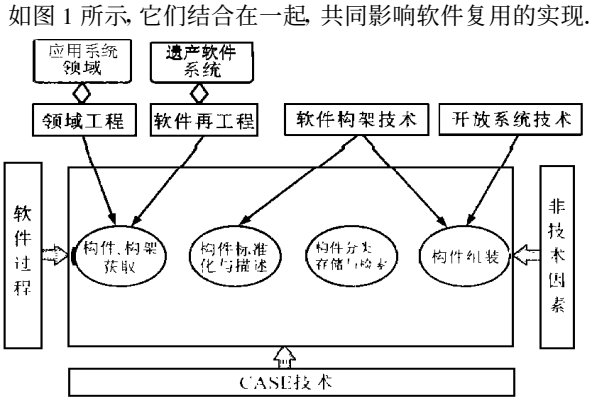


图 1　实现软件复用的关键因素

1. 软件构件技术

构件(Component)是指应用系统中可以明确辨识的构成

成分.而可复用构件(Reusable Component)是指具有相对独立的功能和可复用价值的构件.

可复用构件应具备以下属性[24]:①有用性(Usefulness):构件必须提供有用的功能;②可用性(Usability):构件必须易于理解和使用;③质量(Quality):构件及其变形必须能正确工作;④适应性(Adaptability):构件应该易于通过参数化等方式在不同语境中进行配置;⑤可移植性(Portability):构件应能在不同的硬件运行平台和软件环境中工作.

随着对软件复用理解的深入,构件的概念已不再局限于源代码构件,而是延伸到需求、系统和软件的需求规则约、系统和软件的构架、文档、测试计划、测试案例和数据以及其他对开发活动有用的信息.这些信息都可以称为可复用软件构件.

软件构件技术是支持软件复用的核心技术,是近几年来迅速发展并受到高度重视的一个学科分支.其主要研究内容包括:

(1)构件获取:有目的的构件生产和从已有系统中挖掘提取构件;

(2)构件模型:研究构件的本质特征及构件间的关系;

(3)构件描述语言:以构件模型为基础,解决构件的精确描述、理解及组装问题;

(4)构件分类与检索:研究构件分类策略、组织模式及检索策略,建立构件库系统,支持构件的有效管理;

(5)构件复合组装:在构件模型的基础上研究构件组装机制,包括源代码级的组装和基于构件对象互操作性的运行级组装;

(6)标准化:构件模型的标准化和构件库系统的标准化.

2 软件构架

软件构架是对系统整体结构设计的刻划,包括全局组织与控制结构,构件间通讯、同步和数据访问的协议,设计元素间的功能分配,物理分布,设计元素集成,伸缩性和性能,设计选择等[9].

研究软件构架对于进行高效的软件工程具有非常重要的意义:通过对软件构架的研究,有利于发现不同系统在较高级别上的共同特性;获得正确的构架对于进行正确的系统设计非常关键;对各种软件构架的深入了解,使得软件工程师可以根据一些原则在不同的软件构架之间作出选择;从构架的层次上表示系统,有利于系统较高级别性质的描述和分析.特别重要的是,在基于复用的软件开发中,为复用而开发的软件构架可以作为一种大粒度的、抽象级别较高的软件构件进行复用,而且软件构架还为构件的组装提供了基础和上下文,对于成功的复用具有非常重要的意义.

软件构架研究如何快速、可靠地从可复用构件构造系统的方式,着重于软件系统自身的整体结构和构件间的互联.其中主要包括:软件构架原理和风格,软件构架的描述和规约,特定领域软件构架,构件向软件构架的集成机制等.

3 领域工程

领域工程是为一组相似或相近系统的应用工程建立基本能力和必备基础的过程,它覆盖了建立可复用软件构件的所有活动[22].领域是指一组具有相似或相近软件需求的应用系统所覆盖的功能区域.领域工程包括三个主要的阶段.

(1)领域分析:这个阶段的主要目标是获得领域模型(Domain Model).领域模型描述领域中系统之间的共同的需求[17].这个阶段的主要活动包括确定领域边界,识别信息源,分析领域中系统的需求,确定哪些需求是被领域中的系统广泛共享的,哪些是可变的,从而建立领域模型.

(2)领域设计:这个阶段的目标是获得领域构架(Domain-Specific Software Architecture,缩写为 DSSA).DSSA 描述在领域模型中表示的需求的解决方案,它不是单个系统的表示,而是能够适应领域中多个系统的需求的一个高层次的设计[17].建立了领域模型之后,就可以派生出满足这些被建模的领域需求的 DSSA.由于领域模型中的领域需求具有一定的变化性,DSSA 也要相应地具有变化性.

(3)领域实现:这个阶段的主要行为是定义将需求翻译到由可复用构件创建的系统的机制.根据所采用的复用策略和领域的成熟和稳定程度,这种机制可能是一组与领域模型和 DSSA 相联系的可复用构件,也可能是应用系统的生成器.

这些活动的产品(可复用的软件构件)包括:领域模型、领域构架、领域特定的语言、代码生成器和代码构件等.

在领域工程的实施过程中,可能涉及的人员包括[5]:

(1)最终用户:使用某领域中具体系统的人员;

(2)领域专家:提供关于领域中系统信息的人员,他应该熟悉该领域中系统的软件设计和实现、硬件限制、未来的用户需求及技术走向;

(3)领域分析员:收集领域信息、完成领域分析并提炼出领域产品(可复用软件构件)的人员,他应该具有完备的关于复用的知识,并对分析的领域有一定程度的了解.

(4)领域分析产品(构件、构架)的使用者:包括最终用户、应用系统的需求分析员和软件设计者.

领域工程的主要产品和人员如图 2 所示.

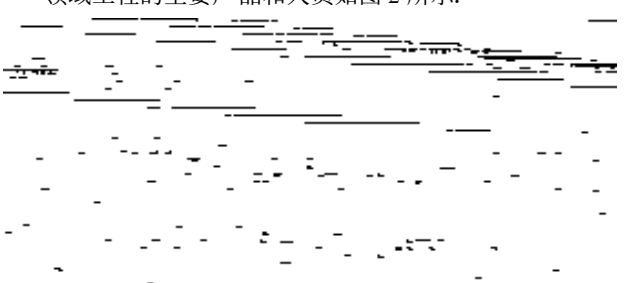


图 2 领域工程

4 软件再工程

软件复用中的一些问题与现有系统密切相关,如:现有软件系统如何适应当前技术的发展及需求的变化,采用更易于理解的、适应变化的、可复用的系统软件构架并提炼出可复用的软件构件?现存大量的遗产软件系统(Legacy Software)由于技术的发展,正逐渐退出使用,如何对这些系统进行挖掘、整理,得到有用的软件构件?已有的软件构件随着时间的流逝会逐渐变得不可使用,如何对它们进行维护,以延长其生命

期, 充分利用这些可复用构件? 等等. 软件再工程(Software Reengineering)正是解决这些问题的主要技术手段.

软件再工程是一个工程过程, 它将逆向工程、重构和正向工程组合起来, 将现存系统重新构造为新的形式[7]. 再工程的基础是系统理解, 包括对运行系统、源代码、设计、分析、文档等的全面理解. 但在很多情况下, 由于各类文档的丢失, 只能对源代码进行理解, 即程序理解.

再工程的主要行为如图 3 所示.

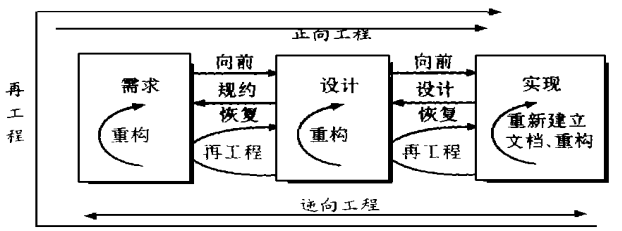


图 3　软件再工程

5. 开放系统技术

开放系统技术的基本原则是在系统的开发中使用接口标准, 同时使用符合接口标准的实现. 这些为系统开发中的设计决策, 特别是对于系统的演化, 提供了一个稳定的基础, 同时, 也为系统(子系统)间的互操作提供了保证. 开放系统技术具有在保持(甚至是提高)系统效率的前提下降低开发成本、缩短开发周期的可能. 对于稳定的接口标准的依赖, 使得开发系统更容易适应技术的进步[8]. 当前, 以解决异构环境中的互操作为目标的分布对象技术是开放系统技术中新的主流技术.

开放系统技术为软件复用提供了良好的支持. 特别是分布对象技术使得符合接口标准的构件可以方便地以"即插即用"的方式组装到系统中, 实现黑盒复用. 这样, 在符合接口标准的前提下, 构件就可以独立地进行开发, 从而形成独立的构件制造业.

6. 软件过程

软件过程又称软件生存周期过程, 是软件生存周期内为达到一定目标而必须实施的一系列相关过程的集合. 一个良好定义的软件过程对软件开发的质量和效率有着重要影响. 当前, 软件过程研究以及企业的软件过程改善已成为软件工程界的热点, 并已出现了一些实用的过程模型标准, 如 CMM、ISO 9001/ TickIT 等.

然而, 基于构件复用的软件开发过程和传统的一切从头开始的软件开发过程有着实质性的不同, 探讨适应于软件复用的软件过程自然就成为一个迫切的问题.

7. CASE 技术

随着软件工程思想的日益深入人心, 以计算机辅助开发软件为目标的 CASE(Computer Aided Software Engineering)技术越来越为众多的软件开发人员所接受, CASE 工具和 CASE 环境得到越来越广泛的应用. CASE 技术对软件工程的很多方面, 例如分析、设计、代码生成、测试、版本控制和配置管理、再工程、软件过程、项目管理等等, 都可以提供有力的自动或半自动支持. CASE 技术的应用, 可以帮助软件开发人员控制软件开发中的复杂性, 有利于提高软件开发的效率和质量.

软件复用同样需要 CASE 技术的支持. CASE 技术中与软件复用相关的主要研究内容包括: 在面向复用的软件开发中, 可复用构件的抽取、描述、分类和存储; 在基于复用的软件开发中, 可复用构件的检索、提取和组装; 可复用构件的度量等等.

8. 非技术因素

除了上述的技术因素以外, 软件复用还涉及众多的非技术因素, 如: 机构组织如何适应复用的需求; 管理方法如何适应复用的需求; 开发人员知识的更新; 创造性和工程化的关系; 开发人员的心理障碍; 知识产权问题; 保守商业秘密的问题; 复用前期投入的经济考虑; 标准化问题等等. 这些因素超出了本文的范围, 这里不再详细讨论.

三、复用的研究与实践活动

1. 领域工程

研究实践表明, 软件复用在特定领域内更容易获得成功. 因此, 领域工程受到高度重视, 已有许多研究成果. 这里将介绍三个有代表性的工作.

卡内基·梅隆大学的软件工程研究所(CMU/ SEI)提出了面向特征的领域分析方法(Feature-Oriented Domain Analysis Method, 缩写为 FODA 方法)[8]. 它支持对某领域中系统共性和个性的发现、分析和文档记录. FODA 的过程分为三个阶段: 上下文分析(Context Analysis)、领域建模(Domain Modeling)、构架建模(Architecture Modeling).

(1)上下文分析: 上下文分析的目标是定义领域的范围. 在这个阶段要分析领域与外部元素间的关系, 例如不同的操作环境, 不同的数据需求等, 还要对可变性进行评价. 上下文分析的结果是上下文模型.

(2)领域建模. 领域的范围确定后, 领域建模阶段提供了一些步骤来分析领域中的应用的共同性和差异, 并产生领域模型. 领域建模阶段主要包含三种行为:

·特征分析(Feature Analysis). 在特征分析阶段, 要获得客户或最终用户对一类系统的一般能力的理解, 即特征. 特征描述了领域应用的上下文、需要的操作和属性、以及表示的变化.

·信息分析(Information Analysis). 在信息分析中, 要定义和分析为实现领域中应用所需的领域知识和数据需求. 信息分析的目标是用领域实体及其相互之间的关系表示领域知识, 并使它们在操作分析和构架建模中可以用来派生对象和数据定义.

·操作分析(Operational Analysis). 在操作分析中, 要识别领域中应用的行为特性, 例如数据流和控制流的共同性和差异、有限状态自动机模型等.

(3)构架建模: 这个阶段为领域中的应用提供软件解决方案. 在这个阶段中开发出构架模型, 即领域中应用的高层设计. 这个阶段的焦点是识别并发进程和面向领域的共同模块. 这个阶段中定义进程, 并将定义在领域模型中的特征、功能和数据对象分配到进程和模块中.

FODA 方法已成功地应用于美国空军运动控制等领域.

在美国国防部高级研究项目署(ARPA)资助下, Will Tracz 提出了领域构架方法(Domain-Specific Software Architecture), 缩写为(DSSA 方法)[22]. 在最高的级别上, 该方法有五个阶段. 每个阶段可以进一步划分为一些步骤或子阶段. 每个阶段包括一组需要回答的问题, 一组需要的输入、一组将产生的输出和验证标准. 该方法的领域工程过程是并发的(concurrent)、递归的(recursive)和反复的(iterative). 或者可以说, 它是螺旋型的(spiral). 完成该过程可能需要对每个阶段经历几遍, 每次增加更多的细节.

该领域工程过程的五个阶段是:

(1)定义领域范围: 重点是确定领域中包含哪些元素以及领域工程过程到何时结束. 这个阶段的一个主要输出是领域中的应用需要满足的一系列用户的需求;

(2)定义领域特定的元素: 目标是制订领域字典和领域术语的同义词词典. 在领域工程过程的前一个阶段产生的高层次块图中增加更多的细节, 特别是识别领域中应用间的共同性和差异性;

(3)定义领域特定的设计和实现需求约束: 目标是描述解空间中的特性. 不仅要识别约束, 并且要记录约束对设计和实现决定造成的后果, 还要记录对处理这些问题时产生的所有问题的讨论;

(4)定义领域模型和构架: 目标是产生 DSSA, 并说明构成它的模块或构件的语法和语义;

(5)产生、搜集可复用产品: 目标是为 DSSA 增加构件使得它可以被用来产生问题域中的新应用.

STARS 领域分析过程(Domain Analysis Process)分四个阶段, 每个阶段用结构化分析与设计技术描述, 使用 SADT 图[22]:

图 4 瑞典 CelsiusTech System 公司的产品线

图 5 美国空军电子系统中心(ESC)的产品线

图 6 ALOAF 参考模型

3 构件及构件库的标准化

美国军方与政府资助的项目中, 已建立了若干构件库系统, 如 CARDS、ASSET、DSRS 等. 由 DARPA 发起, 由美国军方、SEI 和 MITRE 支持的 STARS 项目在此基础上考虑了开放体系结构的构件库之间共享资源和无缝互操作的问题, 并于 1992 年提交了 ALOAF(Asset Library Open Architecture Framework, 开放体系结构的构件库框架)Version1.2 版本. 这一报告体现了 STARS 对可复用构件库系统的认识, 给出了一个构件库框架的参考模型, 并就此实现了 ALOAF 规约作为该参考模型的实例, 由此证明以公共元模型为基础, 在构件库之间交换信息和创建易于移植的复用工具是可能的和必要的. ALOAF 参考模型如图 6 所示[21].

(1)准备领域信息: 定义领域, 按自顶向下方式完成高级功能分析;

(2)分类领域实体: 按自底向上方式识别和描述对象及操作, 并构造领域字典;

(3)导出领域模型: 结合上述信息, 构造基于可复用构件的类属功能模型;

(4)扩展模型并分类: 应用并确认模型.

2. 产品线系统

产品线系统(Product Line System)是 CMU/SEI 提出的产品开发的组织方式. 产品线集中体现了软件复用思想. 经验表明, 单靠技术方法并不能保证成功的产品线生产能力, 经济、组织、管理和过程在建立和维护产品线中起到了关键作用. 一个产品线是共享一组共同设计及标准的产品族, 从市场角度看是在某市场片断中的一组相似的产品. 建立产品线是根据生产的经济学, 使产品族可复用构件能达到最大限度的复用目的. 产品线方法可以通过各种可复用软件构件, 如需求、需求规约、构架、代码构件、文档、测试策略和计划、测试案例和数据、开发人员的知识和技能、过程、方法及工具等, 支持最大限度的软件复用. 产品线也是基于在相同产品价格条件下提高竞争力的商业考虑.

产品线系统已有成功的应用实例. 图 4 和图 5 分别是瑞典 CelsiusTech System 公司和美国空军电子系统中心(ESC)的产品线系统[3,6].

这两个产品线系统的共同特点是构架组、构件组和集成组的分离. 构架组负责产品线系统构架的定义和演化. 构件组负责根据产品线系统构架, 生产和管理可复用构件. 集成组则根据具体客户的需求, 利用产品线系统构架和可复用构件进行具体的系统集成.

ALOAF 的短期目标是尽快实现不同复用库间的构件共享, 长期目标是实现异质构件库间的无缝互操作, 包括构件及其描述的共享及丰富的构件库工具集的易移植性. 其基本观点是软件系统的开发将演化成一种过程驱动的、特定于领域的和基于复用的技术支持范型. STARS 认为构件库是软件工程环境(SEE)的子系统之一, 其提供的功能应与 SEE 框架的参考模型一致. 构件库系统利用 SEE 提供的服务, 并向上层提供服务. ALOAF 更关注构件库框架而非更广泛的 SEE 框架. ALOAF 包括构件库参考模型及其服务规约和数据描述规约. ALOAF 在构件库和复用服务方面对 SEE 参考模型进行补充. ALOAF 与 SEE 的关系如图 7 所示.

可复用库互操作组织(Reuse library Interoperability

Group, 缩写为 RIG)是一个志愿的基于一致意见的组织, 由来自政府、学校和私人企业的成员组成. RIG 认为, 当前几乎没有有效的方法在众多的政府和企业界软件复用库间共享软件构件, 其结果是低效和不必要的冗余. 可互操作性, 即复用库间软件构件的共享, 可能是在短期内提高效率、增加复用库的价值和影响, 并在长期中避免不兼容协议的爆炸性增长的关键. 因此, RIG 的目标是致力于开发可互操作性的解决方案. RIG 的成果包括 BIDM 和 UDM[19~20]. UDM 是基于 ALOAF 的三层数据模型的统一数据模型. BIDM 是 UDM 的一个子集, 定义了为了实现互操作, 复用库交换软件构件时所需的信息的最小集. BIDM 已被 IEEE 采纳为标准.

北大西洋公约组织(NATO)针对 NATO、NATO 参与国和承包商制定了一组关于软件复用的标准, 其中包括"可复用构件开发标准"、"可复用软件构件库管理标准"、"软件复用过程标准"[13~15]. 制订这些标准的目标是供 NATO 及其参与国的项目管理部门使用它们来建立复用计划需求和向承包商提供指导. 承包商则将它们用于特定项目的开发实践.

4 构件组装技术

CORBA 是公共对象请求代理体系结构(Common Object Request Broker Architecture)的缩写, 是对象管理组织(OMG)开发的一套分布式对象技术标准, 涉及接口、注册、数据库、通信和出错处理等方面的问题[16]. 和对象管理体系结构(OMA)定义的其他对象服务相结合, CORBA 成为支持分布式系统中对象技术的中间件设施. CORBA 的对象请求代理(ORB)作为转发消息的中间件, 实现了对象间的无缝集成和互操作. 因此, CORBA 可作为面向对象的软件构件在运行级上组装的技术基础, 从而实现构件的黑盒复用. 当前已有许多符合 CORBA 的 ORB 产品, 如 ORBIX、NEO、VisiBroker、PowerBroker、SmallTalkBroker、SOM/DSOM、DAIS、NonStop 等.

OLE 是微软公司开发的支持对象连接嵌入的机制, 其初旨是解决复合文档问题[10]. OLE 为构件对象的互操作提供了基础支持, 因此, 也为构件的黑盒复用提供了技术基础. OLE 是主要的不符合 CORBA 标准的对象连接中间件. DCOM 是微软公司开发的分布式构件对象模型, 支持分布式系统中的面向对象技术[11].

JAVA 是近几年随着 WEB 风行全球而发展起来的一种新语言. 它是一种纯 OO 的语言, 外观象 C++, 内核类似 Smalltalk. JAVA 和 WEB 的结合带来了移动的对象、可执行的内容等关键概念. JAVA 具有体系结构中立的特性, 从而使得 JAVA 程序可不需修改甚至重编译而运行于不同平台之上. JAVA 的新版还加入了远程方法调用(RMI)的特性, 在效果上, RMI 提供了类似CORBA 的 ORB 的功能. JAVA 的这些特性使其成为软件构件技术的良好支持工具. 用 JAVA 书写的构件将具有平台独立性和良好的互操作性.

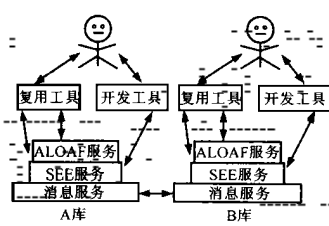


图 7　ALOAF 与 SEE 的关系

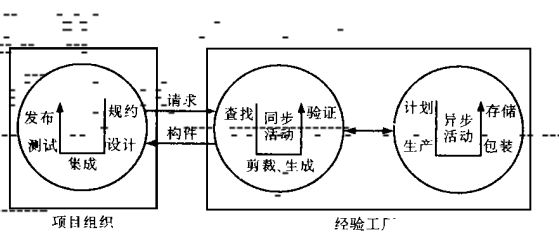


图 8　一种基于复用的软件生产方式

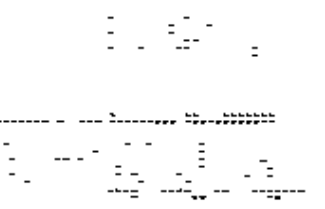


图 9　青鸟软件生产线

这些技术的流行为构件组装提供了很好的技术支持, 同时它们也为构件提供了实现标准. 软件复用和分布对象技术的结合使得即插即用的构件黑盒组装成为可能.

5 基于复用的软件开发过程

Caldieri 和 Basili 提出了一种工厂化的软件生产方式[4], 如图 8 所示.

在这种模型中, 构件生产组和系统开发组间严格按照生产者——消费者关系进行任务分工, 经验工厂负责生产、提供构件, 项目组不再编程, 而是通过从经验工厂中请求所需的构件集成组装而得到最终所需的系统. 经验工厂的活动分为同步活动和异步活动. 同步活动指配合项目组的活动, 接收构件查找请求或定制请求, 为项目组服务. 异步活动指有目的的构件生产或对同步活动中的构件进行再工程以提高构件的可复用性.

6 复用成熟度模型(RMM)

受 SEI 的能力成熟度模型(Capability Maturity Model, 缩写为 CMM)的启发, 已出现了几个复用成熟度模型(Reuse Maturity Model, 缩写为 RMM), 作为对企业内复用水平层次的度量.

在 IBM 的 RMM 中, 将企业的软件复用水平分为五级. 随着复用成熟度级别的提高, 复用的范围、复用使用的工具和复用的对象都有变化: ① 初始级(Initial): 不协调的复用努力. 复用是个人的行为, 没有库的支持, 主要的复用对象是子程序和宏; ② 监控级(Monitored): 管理上知道复用, 但不作为重点. 复用是小组的行为, 有非正式的、无监控的数据库, 复用的对象包括模块和包; ③ 协调级(Coordinated): 鼓励复用, 但没有投资. 复用的范围包括整个部门, 有配置管理和构件文档的数据库, 复用的对象包括子系统、模式和框架; ④ 计划级(Planned): 存在组织上的复用支持. 在项目级别支持复用, 有复用库, 复用的对象包括应用生成器; ⑤固有级(Ingrained): 规范化的复用支持. 复用成为整个企业范围的行为, 有一组领域相关的复用库, 复用的对象包括 DSSA.

Loral Federal System 公司的 RMM 也分为五级: ①初始级: 偶尔的开发过程复用; ②基本级: 在项目级上定义的开发过程复用; ③系统化级: 标准的开发过程复用; ④面向领域级: 大规模的子系统复用; ⑤软件制造级: 可配置的生成器及

DSSA.

HP 的 RMM 将复用成熟度与复用率联系起来，也分为五级：①无复用：－20％至 20％的复用率；②挖掘整理：15％至 50％的复用率；③计划复用：30％至 40％的复用率；④系统化复用：50％至 70％的复用率；⑤面向领域的复用：80％至 90％的复用率.

四、青鸟软件生产线

青鸟工程是国家重点支持的科技攻关课题，已有十余年的发展历程.“七五”、“八五”期间，青鸟工程面向我国软件产业基础建设的需求，以实用的软件工程技术为依托，研究开发具有自主版权的软件工程环境，为软件产业提供基础设施——软件工具、平台和环境，建立工业化生产的基本手段，促进我国软件开发由手工作坊式转向用计算机辅助开发，以提高软件开发效率，改善软件产品质量. 大型软件开发环境青鸟系统便是这一阶段攻关工作的成果[1].

“九五”期间，青鸟工程的任务是在前期攻关工作的基础上，为形成我国软件产业规模提供技术支持. 重点是研究软件的工业化生产技术，开发软件工业化生产系统——青鸟软件生产线系统，即基于构件——构架模式的软件开发技术及系统，为软件开发提供整体解决方案，推行软件工业化生产模式，促进软件产业规模的形成.

作为研究成果之一，青鸟工程开发了基于异构平台、具有多信息源接口的应用系统集成(组装)环境青鸟Ⅲ型(JB3)系统. 青鸟Ⅲ型系统研制的目标是针对软件工业化生产的需求，完善并初步实现青鸟软件生产线的思想，制定软件工业化生产标准和规范，研究基于“构件——构架”模式的软件工业化生产技术，研制支持面向对象技术，支持软件复用的，基于异构平台、具有多信息源接口的应用系统集成(组装)环境. 其最终目标是要构造如图 9 所示的软件生产线系统[2].

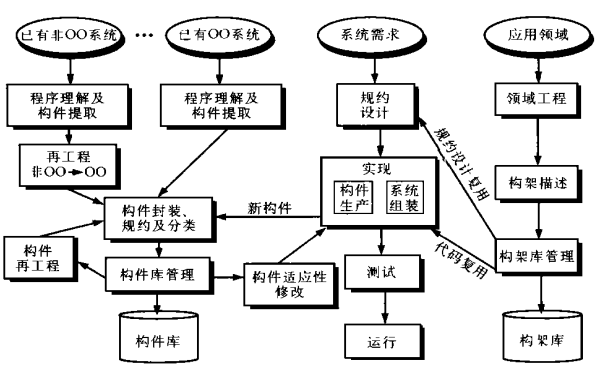


图 10 青鸟软件开发过程

在青鸟软件生产线中，软件的生产过程划分为三类不同的生产车间，即应用构架生产车间、构件生产车间和基于构件、构架复用的应用集成(组装)车间，从而形成软件产业内部的合理分工，实现软件的工业化生产. 软件开发人员被划分成三类：构件生产者、构件库管理者和构件复用者. 这三种角色所需完成的任务是不同的，构件生产者负责构件的生产、描述；构件库管理者负责构件分类以及构件库的管理工作；而构件复用者负责进行基于构件的软件开发，包括构件查询、构件理解、适应性修改、构件组装以及系统演化.

其技术路线是以现有的青鸟软件开发环境为基础，研究构件－构架模式的软件生产技术，制定符合国情、国际兼容的青鸟构件标准规范；支持专业化的构件生产，即有目的的构件开发和采用再工程技术从已有系统中获取构件；采用领域工程技术，支持专业化的构架开发；强有力的构件库、构架库管理；支持软件过程设计和控制；支持基于构件、构架复用的应用系统集成(组装). 青鸟Ⅲ型系统支持的软件开发过程如图 10 所示.

青鸟Ⅲ型系统的体系结构如图 11 所示：

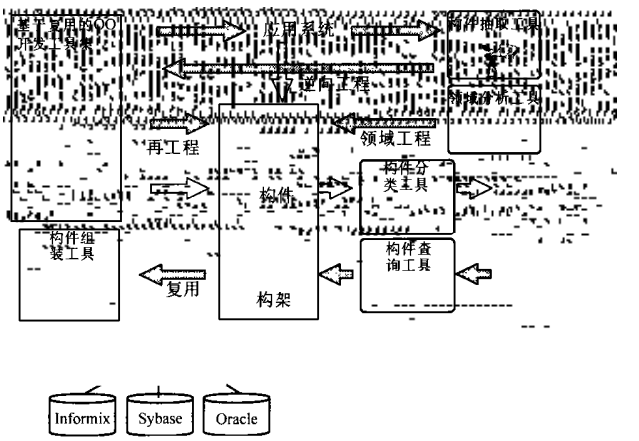


图 11 青鸟Ⅲ型系统的体系结构

在对国际相关标准规范进行比较研究的同时，青鸟工程对国际上学术界关于软件复用和软件构件技术的一些主要成果进行了分析研究，结合前期攻关和 863 课题研究的实践和基础，提出了一套青鸟构件标准规范. 这套标准规范保持了和国际标准规范的接轨和兼容，同时也融进了一些国际上重要研究成果和自己的研究成果，具有自身特色和优点. 这些规范包括：青鸟可复用构件制作指南、青鸟领域工程方法指南、青鸟构件模型、青鸟构件描述语言、青鸟构件库概念模型、基于复用的软件开发过程.

青鸟构件－构架模式的软件开发技术已在北大青鸟商用系统公司开发的若干商场管理系统中进行试用. 青鸟公司先后开发了桂林百货大楼、北京前门商场等 POS 系统，积累了丰富的领域信息和经验，通过领域工程，提炼出 POS 系统的类属构架及 120 多个构件. 现正使用这些构架和构件进行新的 POS 系统的集成组装工作.

五、结束语

自软件复用被提出以来，人们进行了许多复用的实践活动. 归纳起来，复用项目的成功主要发生于以下几种情形：①在较小的特定领域；②在理解充分的领域；③当领域知识变动缓慢时；④当存在构件互联标准时；⑤当市场规模形成时(大量的项目可以分担费用)；⑥当技术规模形成时(有大量可用的、可获利的构件).

而复用项目失败的原因主要包括：①缺乏对复用的管理支持；②没有对开发可复用软件及复用已有软件的激励措施；

③没有强调复用问题的规程或过程; ④没有足够的可复用资源; ⑤没有良好的分类模式, 使得构件查找比较困难; ⑥没有良好的构件库支持和控制复用; ⑦构件库中的构件没有良好的接口; ⑧已有的部件不是为了复用而开发的.

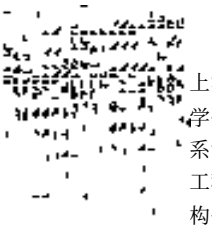
经过软件复用的研究和实践方面的努力, 在构件开发方面已经取得一定的成果. 当前已存在一些政府、军方或企业自己拥有的构件库, 在某些领域, 如科学计算, 已有商用的构件存在. 同时, 存在大量独立于应用领域的计算机特定的软件构件, 如: 程序设计语言的类库、函数库、VBX、OCX、用户界面构件等. 但对大多数特定领域来说, 可复用构件仍十分短缺, 从而形成了一个巨大的应用软件构件市场.

软件复用技术将促进软件产业的变革, 使软件产业真正走上工程化、工业化的发展轨道. 软件复用将造成软件产业的合理分工, 专业化的构件生产将成为独立的产业而存在, 软件系统的开发将由软件系统集成商通过购买商用构件, 集成组装而成. 软件复用所带来的产业变革将会带来更多的商业契机, 形成新的增长点.

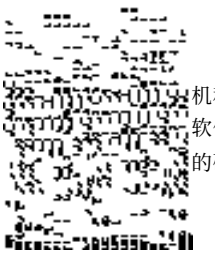
在当前的情况下, 我国的软件产业发展一定要结合国情、抓住机遇. 软件构件技术的应用, 正在促进软件产业改革和重组分工, 这对我国软件产业的发展是一个良好的机遇. 我国正在大力加强国家信息化工作, 具有广大的信息市场. 由于具体的国情, 大多数信息系统的开发工作是由国内公司承担, 因此, 也就培养了一大批领域专家, 为推行软件构件技术、发展软件构件生产业奠定了良好的基础. 同时, 在构件技术方面, 多年的攻关研究已使我们具有良好的技术积累. 我们应在国家支持下, 在行业部门的领导下, 以政府或行业行为的方式推广软件构件技术, 促进软件构件企业的发展. 我国已失去了较多的信息产品市场, 目前, 正面临振兴的机遇, 如何抓住机遇, 也是严峻的挑战.

在软件产业的发展策略上, 应由政府或行业主管部门组织构件标准规范的制定和发布; 选择若干领域进行软件构件技术的推广和软件构件企业建设的试点工作, 推行基于构件—构架模式的软件生产线的工程化、工业化软件生产技术; 加快软件复用和软件构件技术的普及、培训工作, 培养一批高质量的领域分析员队伍; 制定合理措施及政策, 激励软件构件技术的采用和推广; 建立软件风险投资机制和软件生产基金, 激励构件专门企业的形成和零散构件的开发; 尽快完成试点工作, 及早进军国际应用构件市场.

杨芙清　中国科学院院士, 教授, 博士生导师, 中国电子学会、计算机学会副理事长, 北京大学计算机科学技术系主任. 1932 年生于江苏无锡, 1958 年北京大学研究生毕业. 主要研究领域为系统软件、软件工程和软件工程环境、软件复用和软件构件技术、软件工业化生产技术等. 已发表论文 70 多篇, 出版专著 5 部.



梅　宏　教授. 1963 年 5 月出生. 1992 年于上海交通大学计算机科学及工程系获工学博士学位, 1994 年 10 月从北京大学计算机科学技术系博士后出站. 主要研究领域为软件工程和软件工程环境、新型程序设计语言、软件复用和软件构件技术等. 已发表论文 50 多篇.



李克勤　1973 年 3 月出生. 北京大学计算机科学技术系博士研究生. 主要从事软件工程和软件工程环境、软件复用和软件构件技术等方面的研究工作.

参　考　文　献

1 杨芙清, 邵维忠, 梅宏. 面向对象 CASE 环境 JBⅡ型系统的设计和实现. 中国科学, 1995, 5

2 杨芙清. 青鸟过程现状与发展—兼论我国软件产业发展途径. 第六次全国软件工程学术会议论文集, 软件工程进展—技术、方法和实践. 杨芙清, 何新贵主编, 清华大学出版社, 1996, 5

3 Lisa Brownsword, Paul Clements. Technical Report, CMU/SEI-96-TR-016, October 1996

4 Caldiera, V. R. Basili. IEEE Computer, Feb. 1991, 24(2): 61～70

5 Sholom G. Cohen et al. Technical Report, CMU/SEI-91-TR-28, ESD-91-TR-28, June 1992

6 Sholom Cohen et al. Technical Report, CMU/SEI-96-TR-018, September 1996

7 Peter H. Feiler. Reengineering: An engineering problem. Technical Report CMU/SEI-93-SR-5, Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, July 1993

8 John Foreman, Kimberly Brune, Patricia McMillan, Robert Rosenstein. Software Technology Review, CMU/SEI, June 1997

9 David Garlan, Mary Shaw. Advances in Software Engineering and Knowledge Engineering, volume I, World Scientific Publishing, 1993

10 Microsoft Corporation. The component object model specification. Version 0. 9, October 24, 1995[online]. Available WWW〈URL: http://www.mocrosoft.com/oledev/〉(1995)

11 Microsoft Corporation. Distributed component object model protocol COM/1.0. draft, November 1996[online]. Available WWW〈URL: http://www.microsoft.com/oledev〉(1996)

12 Hafedh Mili et al. IEEE Transactions on Software Engineering, June 1995, 21(6): 528～562

13 NATO. NATO standard for developing reusable software components. Vol. 1 of 3 volumes, NATO contact number CO-5957-ADA, 1991

14 NATO. NATO standard for management of a reusable software component library. Vol. 2 of 3 volumes, NATO contact number CO-5957-ADA, 1991

(下转第 51 页)

探测器的量子效率 η 无关.

(2)与光电流不同的是在闭环光场中, 量子噪声分量与经典调制分量之间的反相关是不完全的, 其总的强度起伏可以是亚泊松的, 也可以是超泊松或泊松的, 这取决于反馈增益 G 和光探测器的量子效率 η. 当 $\eta > 50\%$ 时, 不论 G 多大, 闭环光场总的强度起伏总是亚泊松的(也即振幅压缩态光场).

(3)在闭合反馈回路中光电流噪声的压缩与光场强度噪声的压缩包含着不同的内容, 前者不仅包含了被观测到的光场强度噪声的压缩(即后者), 而且还包含了由经典调制分量中与光探测器噪声之间的反相关效应所导致的压缩, 因而光电流噪声的压缩可望突破由探测器量子效率所决定的理论极限.

此外, 由于本实验中 $\eta \geq 85\%$, 故我们所观测到的深度压缩的闭环光场是一个反相关态光场(在量子噪声分量与经典调制分量之间存在着高度的反相关效应, 由于这种反相关效应导致了量子噪声压缩, 故我们称之为非经典的光子反相关效应), 其总的强度起伏是亚泊松的, 也即是一个闭环的振幅压缩态光场.

参 考 文 献

1 S. Mashida and Y. Yamamoto. Observation of sub-Poissonian photoelectron statistics in a negative feedback semiconductor laser. Opt. Commun., 1986, 57(4): 290 ~ 296

2 Y. Yamamoto, N. Imoto. Amplitude squeezing in a semiconducior laser using quantum nondemolition measurement and negative feedback. Phys. Rev., 1986, A33(5): 3243 ~ 3261

3 K. Tsubona and S. Moriwaki. Shot-noise limited low-frequency intensity noise of a Nd: YAG laser. Jan. J. Appl. Phys., 1992, 31: 1241 ~ 1242

4 M. S. Taubman, H. Wiseman, H. A. Bachor. Quantum-limited properties of an intensity noise eater. 1993: 1 ~ 12 (Private Communication)

5 Yin Jianping and Wang Yuzhu. Generation of sub-Poissonian light in an open-loop LEDs system by the quantum correlation. Chin. Phys. Lett., 1993, 10(12): 724 ~ 727

6 J. H. Shapiro, et al. Semiclassical theory of light detection in the present of feedback. Phys. Rev. Lett., 1986, 56(11): 1136 ~ 1139

7 J. H. Shapiro, et al. Theory of light detection in the present of feedback. J. Opt. Soc. Am., 1987, B4(10): 1604 ~ 1619

8 H. M. Wiseman. Quantum theory of continous feedback. Phys. Rev., 1994, A49(3): 2133 ~ 3150

9 P. J. Edwards. Sub-Poisson light from GaAlAs infrared emitting diodes. Int. J. Optoelectr., 1991, 6(42): 23 ~ 28

(上接第 75 页)

15 NATO. NATO standard for software reuse procedures. Vol. 3 of 3 volumes, NATO contact number CO-5957-ADA, 1991

16 Object Management Group home page [online]. Available WWW 〈URL: http //www.omg.org〉(1996)

17 James Petro, et al. Proc. Seventh International Workshop on CASE, July 10 ~ 14, 1995, Toronto, Ontario, Canada, IEEE Computer Society Press, Los Alamitos, California, USA: 60 ~ 69

18 Ruben Prieto-Diaz. Status report: software reusability. IEEE Software, May 1993, 10(3): 61 ~ 66

19 RIG. Basic Interoperability data model (BIDM). RPS-0001, Reuse Library Interoperability Group, April, 1993

20 RIG. Uniform data model for reuse libraries (UDM). RPS-0002, Reuse Library Interoperability Group, Jan., 1994

21 STARS. Asset library open architecture framework version 1.2. Informal Technical Report STARS-TC-04041/001/02, 1992/8

22 Will Tracz. Confessions of a Used Program Salesman-Institutionalizing Software Reuse. Addison-Wesley Publishing Co., New York, NY, April 1995