基于构架的软件重用技术综述

桑大勇[1,2]　王　瑛[2]
(1.武汉大学 软件工程国家重点实验室，湖北 武汉　430072;2.空军工程大学 工程学院，陕西 西安　710038)

摘　要:论述了构架的定义和特征,详细讨论了构架重用技术同部件重用、软件自动生成、特定域的软件建筑(DSSA)以及设计样本等软件重用技术之间的关系,最后列举了构架重用的几个缺点和今后的研究课题。

关键词:软件重用;构架;部件重用;特定域的软件建筑(DSSA);设计样本
中图分类号:TP311.5　　**文献标识码**:A　　**文章编号**:1009-3516(2000)05-0084-03

为了实现软件重用,从技术上说有多种途径,但归纳起来不外乎下面两种类型:一种是基于形式化语言进行重用,另一种是基于部件进行重用。前一种技术将应用系统领域知识融入应用系统生成器(Application generator)或程序设计语言之中;后一种技术是使软件系统开发人员可以利用部件集成工具或手工将若干重用部件(Reusaable components)合成为一个新的应用系统。从最近的许多文献来看,基于部件的软件重用技术得到了较为广泛的应用,已经逐渐成为软件工业界进行重用活动的主要技术手段[1]。

随着部件重用技术的深入研究,出现了许多以部件重用技术为基础或与部件重用密切相关的重用技术,如领域分析与领域工程、设计样本、特定域的软件建筑等。但是单纯地以部件作为一种重用的单位,在很多情况下其粒度显得太小,因此构架重用作为更大粒度的重用技术渐渐起了广泛的重视。

1　构架的定义和特征

关于构架,至今仍没有一个统一的定义,如同面向对象技术刚开始出现时的情形一样,并不妨碍构架技术的应用与发展。下面给出一些最近的科技文献中对构架所提供的定义:

构架是一组面向对象语言中类的集合,设计及实现这些类的目的是使它们一起工作,以便能够解决某个问题或者提供某种能力[2]。软件开发人员可以通过继承这些类、重置其中的某些方法来实施对构架的重用。

构架是比部件更大粒度的重用对象,由若干个重用部件组成。这些部件往往是特定于应用领域的,因此构架也应当是特定于应用领域的[3]。

构架是一种应用系统框架,应用系统开发人员可以将构架加以定制从而形成不同的应用系统。也就是说构架的目标是实现针对某个应用领域的特定事务单元[4]。尽管构架的定义多种多样,但是通过这些定义加以综合,不难发现构架应当具有下面这些特征:①构架主要侧重于设计重用,这一点与模板(Templates)重用或者设计计划(Design Schemas)重用[5]等高层设计重用方式相似;②构架用程序设计语言描述,而模板或设计计划往往要借助于特别的符号系统及软件工具来描述,部件的描述见参考文献[1],它们与构架的描述均有差异;③构架的尺寸范围很广,可以小到一组类组成的结构,这些类的实例对象提供特定应用领域中一簇相似应用系统的共同(可重用)的基本设计,又可以大到一个自完整的(Self-Complete)高层模块,这个模块定制出来以后就成了特定域的应用系统[6]。

收稿日期:2000-09-01
基金项目:中国高等学校重点实验室访问学者基金资助项目(教基司2000-123)
作者简介:桑大勇(1963-),男,安徽寿县人,副教授,博士后,主要从事面向对象软件开发技术及软件工程研究.

2　构架重用技术与其它重用技术的关系

2.1　与部件重用技术的关系

起初有人认为构架也是一种部件形式,因为一个应用系统可能也需要使用不止一个构架,如同重用部件一样,每个构架也有完整的外部接口。但一般来说构架的接口要比部件的接口复杂得多,因此被重用以前可能要求用户花费更多的时间和精力去弄清楚这些构架接口。不过构架与部件有极密切的关系,在实际进行重用时,构架重用技术和部件重用技术难以完全分离:

(1)构架为部件提供了被重用的上下文,即构架中要使用重用部件。每个重用部件对其重用时的外部环境都做了一定的假设,构架可以认为是一种包容多个部件的外部环境,能够沟通部件间相互作用的关系(如进行数据交换、激活对方部件的操作等等),因而可以认为构架是一种比重用部件粒度更大的重用对象。

(2)构架使得开发新的重用部件的工作更为容易。应用系统的需求千变万化,再好的重用部件库不断添加新的部件,而构架作为根据就用系统需求抽象出来的设计结构,为创建该应用领域的重用部件框定了功能需求和实现模板,从而使重用部件的创建减少盲目性。

2.2　与应用系统自动生成技术及特定域的软件建筑的关系

应用系统的自动生成技术是与部件重用技术相区别的另一类重用实现技术,构架重用与这种重用技术也有一定的关系。应用系统生成工具是通过编译一种特定领域的高层语言程序来生成应用系统,Roberts 和 Johnson 认为有可能将这种融入了丰富的领域知识的高层语言程序直接翻译成构架中的对象集合,从而将两种方法集成起来[7]。可以认为构架就是一种特定域的软件建筑(Domain－specific software architecture, DSSA),只是构架都是面向对象设计结果的积累,而一般的 DSSA 并不一定是面向对象。

2.3　与设计样本的关系

在面向对象范畴中,设计样本近年来成为一种颇为时髦的设计信息重用方式[8]。一个设计样本描述了一个问题、问题的解以及问题的解所适用的上下文。构架和设计样本之间具有内在联系,尤其是同样本语言之间的关系更为紧密。所谓样本语言,就是由若干相关的样本组织成的树状或图状结构,代表了利用设计样本完成应用系统设计工作的开发过程[9]。构架和设计样本之间的关系如下:一组相关的特定域的设计样本组成样本语言,由样本语言可以产生构架,而构架提供了每个样本的面向对象程序设计语言实现。因而可以认为一个构架包含有多个粒度更小的设计样本,这些设计比构架更抽象,因为它们不能简单地用 C＋＋或 Smalltalk 的类来表示,也不能简单地通过继承或聚合机制来达到重用的目的。

3　构架重用的缺点及其研究方向

尽管构架是一个历史比较长的重用概念,但一直未得到广泛应用,因为它存在许多缺点,主要有:

(1)构架大量使用传统的面向对象程序设计语言描述,使得它难以很快地被开发人员所理解,同时又局限于在特定的语言开发环境中进行重用。

(2)构架难以同其它不在同一个抽象层次上的重用对象(如类库、已有的重用部件库等)集成,而从构架到实际应用的转化过程的实现越来越依赖于这种集成的成功与否。

(3)为了提高构架的可重用程度,要求构架尽量减少同具体应用有关的细节,一般我们是通过使用类型参数化手段来达到这个目的的。但是同类属部件的情形相似,参数化使构架的测试和验证工作更趋复杂。

为了使构架重用更多更快地从实验室走向软件开发实践中来,Fayad 和 Schmidt 认为今后一段时间内有关构架的研究应当集中在下面几个方面[10]

(1)采用新的构架开发理念。传统的构架都是从现有的应用系统中通过抽象后提取出来的,这种"自底向上"的构架开发方法往往周期很长,同时也容易忽略很多重要的设计思想,因为许多设计思想在最后的应用系统中并没有明显的痕迹。因此以后应当制定一整套构架和样本的设计原理,我们应当在设计原理的指导下从头开始开发特定应用领域的设计样本和构架。

(2)更多地关注企业业务领域里的构架开发工作。目前已经开发出来的构架大量集中于较为通用的应用领域(如人机界面应用、操作系统/通信系统等)。

(3)进行构架的文档化工作。准确而又深入的描述文档对于大型构架的重用成功至关重要,手工进行构架的文档化工作既费时费力又往往难以完成,而现有的文档生成工具只能产生一些底层描述文字信息,这些描述文档不能描述构架内部各个部件之间的协同工作或交互信息以及它们的集成信息。

(4)加强构架开发进程的管理。

(5)建立构架重用的经济效益度量模型,同部件重用在这个方面的需求相类似[1]。

(6)加快构架的标准化工作。随着构架的应用日趋广泛,构架的复杂程度越来越高,构架的标准化工作亟待解决,构架的标准化将保证应用系统的一致性、提高构架的重用程度、降低构架的平均开发成本。

4 结束语

构架重用技术又得到了一次重新崛起的机遇,主要得益于软件重用领域的研究人员所达成的共识,即高收益的重用一定限于特定的应用领域内进行。领域工程、领域分析、特定域的软件建筑乃至设计样本等学科的发展,无一例外地都是基于面向领域进行重用的思想。构架技术恰好是连接这些技术并将它们同部件重用技术结合起来的一种机制,为软件开发从系统需求分析到代码实现的全过程重用提供了可能,为特定域的应用系统自动生成也提供了可能。在构架重用的若干研究方向中,作者认为标准化的研究工作最为迫切。缺乏标准使得部件重用技术面临许多问题(不同来源的部件的互操作性首当其冲),COM(Component Qbject Model)或 CORBA(Common Object Request Broker Architectore)技术的出现只能是亡羊补牢式的努力,但愿构架能够有一个共同、共认、统一的基础。

参考文献:

[1] 桑大勇. 基于部件的软件重用技术研究[D]. 西安:西安电子科技大学,1998.

[2] Lim W C. Effects of Reuse on Quality, Productivity and Economics[J]. IEEE Software, 1994, 11(5): 23 - 30.

[3] Prieto-Diaz R. Status Report: Software Reusability[J]. IEEE Software, 1993, 10(3): 61 - 66.

[4] Fayad M E, Schmidt D C, Johnson R E. Object-Oriented Application Frameworks: Implementation and Experience[M]. New York: Wiley, 1997.

[5] Lubars M D, Harandi M T. Knowledge—Based Software Using Design Schemas[A]. Proceedings of the 9th International Conference on Software Engineering[C]. 1987: 253 - 262.

[6] Pree W. Design Patterns for Object-Oriented Software Development[M]. Reading, Mass: Addison-Wesley, 1994.

[7] Roberts D, Johnson R E. Evolving Frameworks·A Pattern Language for Developing Frameworks. In Pattern Languages of Program Design[M]. Reading, Mass: Addison-Wesley, 1997.

[8] Gamma E, Helm R, Johnson R, et al. Design Patterns: Elements of Reusable Object-Orientde Software[M]. Reading, Mass: Addison-Wesley, 1995.

[9] Brugali D, Menga G, Aasterm A. The Framework Life Span[J]. Communications of the ACM, 1997, 40(10): 65 - 68.

[10] Fayad M E, Schmidt D C. Object-Oriented Application Frameworks[J]. Communications of the ACM, 1997, 40(10): 32 - 35.

Framework-based Software Reuse Technique

SANG Da-yong[1,2], WANG Ying[2]

(1. Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China;
2. Engineering Institute, AFEU., Xi'an 710038, China)

Abstract: The definitions and characteristics of framework are presented. The relationships between framework-based reuse and component-based reuse, automatic software generation, domain-specific software architecture (DSSA), design patterns are also discussed in detail respectively. Finally, a few shortcomings of framework-based reuse and some research topics on it are listed.

Key words: software reuse; framework; component-based reuse; domain-specific software architecture (DSSA); design pattern