Designing a Domain Framework with Component Management Model*

Sang-Geun Kim

(*Division of Computer Engineering, Sungkyul University*, 147-2 *Anyang* 8-*Dong, Anyang, Korea*)

E-mail: sgkim@sungkyul.edu

Received June 15, 2001; accepted October 17, 2001

Abstract: In this paper, the author redesigns web collaboration system framework using component management model. Component management model gets the connection information between components and high-level development knowledge with Key, Content, and Configuration information. It can show the connection information between components with Key and Content information of each component and support the design knowledge of the sub system of specific application also. The author introduces component management model for easier management of the framework components and easier use of framework itself.

Key words: object-oriented framework; component model; domain framework; framework design; reuse; manage component

A framework is a reusable design expressed as a set of abstract classes and the way their instances collaborate[1]. Framework increases the productivity through reuse of executable components[2] and high-level analysis and design knowledge. Designing software for reuse aims to produce general, extensible software component. In this paper, we define the component management model and design a domain framework (web collaboration system framework) based on that model. Component management model describes the connection information of framework components and this information makes enable to manage framework components with consistent mechanism of component management model.

Framework consists of related components that are described with Key, Content information of component management model and high-level knowledge for constructing framework like abstract analysis, design information that can be described with design pattern. Domain framework cannot be constructed in one development cycle. Framework can be evolved through the iterative process which requires both domain and design expertise[3].

In development process, many components of framework can be added, modified and deleted continuously and this makes very difficult to maintain the structure and various main functions of framework. Component management model can be a solution to this problem with the configuration information of components.

1 Related Work

1.1 Framework

Object-oriented frameworks are reusable designs of all or part of software system described by a set of abstract classes and the way instances of those classes collaborate. Well-Designed framework can reduce the cost of developing an application by an order of magnitude because it lets developer reuse both design and code. They do not require new technology, because they can be implemented with existing object-oriented languages[4]. But

* **Sang-Geun Kim** was born in 1963. He received his Ph.D. degree in Computer Engineering from Chungang University in 1996. He is a professor at the Division of Computer Engineering, Sungkyul University of Korea. His current research areas are software component, software development methodology and multi-agent system.

unfortunately developing a good framework is expensive. A framework must be simple enough to be learned, yet must provide the problem domain, and is always the result of domain analysis, whether explicit and format, or hidden and informal.

Similar objects must be implemented for each problem the framework solves. Bare-bones framework of requires a lot of effort to use. Things that work out-of-the box are much easier to use[5]. It is difficult to tell which component users of the framework will need. Some problem-specific while others occurs in most solutions.

Framework can be implemented in two reuse technique. One is inheritance and the other is composition. Inheritance makes it possible to modify components that are reused for application implementation on user's wish. But it needs programming and results in strong coupling between components and is static so cannot be easily changed at runtime. Composition is very strong reuse technique and can be changed at runtime. But composition requires what is going to change and is difficult to understand by examining static program text.

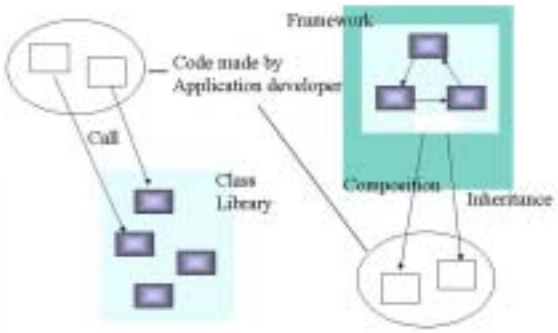

Fig.1 Difference between framework and class library

Framework evolves from white-box-framework, which is based on inheritance to black-box- framework, which is based on composition. Inheritance is the most expedient way of allowing users to change code in object-oriented environment since it is supported by most object-oriented language. So inherite most desired behavior of reused components and override only methods that are different in subclass. White-box-framework will evolve through encapsulation and parameterization of immutable codes of white-box-framework.

1.2 Component library

In general, a component is simply a module. In this sense, applications are made up of components, or are themselves components in large system. But this is vague to software engineer. Component is a software module that publishes or registers its interfaces. Some kind of component system that specifies how the interfaces will be defined, handles the registration of interfaces, and facilitates the communication of interface information between different components. In other word, a component can only be understood within the context of some kind of system or environment, container or server etc.

Component library has object as result in application domain provided by framework. Framework provides well-defined specific component library. It increases usability of framework[9].

Paul Harmon[6] suggested this definition "A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be developed independently and is subject to composition by third party." In this paper, we will mainly discuss component specification for the creation of component management model.

2 Component Management Model

2.1 Generic development process

Building from components should at least allow systems to be constructed from pre-validated parts, and to be incrementally improved by replacing components with others, which provide equivalent services, but with enhanced performance or new features. One of the important reasons for considering the generic process is because it is a

fundamental factor behind the specification of a common meta-model for object-oriented development methods[7]. A generic process of developing from components is relevant in that it implies the definition of types of components, and it also implies, as we shall see two levels of meta-entity, first-class items that may be versioned and second-class items, which cannot. If the goal of a common meta-model is to share components across organizations, a consensus about such a process is required.

2.2 Component management model

We are interested in management of framework component library. A model for component management requires a mechanism share basic common concepts including a basic meta-model of object development of methods and a basic approach to verification and validation component. We have referred component model of Andy Carmichael[8] for defining component management model. Object development methods should support a strong concept of components and building form components. Multiple tools dealing with different aspects of the same components need to share a view of the component architecture: how components are name, related and controlled.

There are 2 important aspects in this architecture. First, components may depend on each other, which is shown by the recursive association. A component depends on some number of other components and may be depended on by some number of other components. Second, a component needs a unique key as well as its user-given name[8].

All components are changed independently but all are dependent. The dependencies of a component vary from version to version and change to a dependent component would ripple up the dependency of hierarchy. So a model of component (Fig.2) is a solution of treating Key as a separate object linked to the Content of a component. The Content of a component is identified by exactly one key, but the key may be the identifier of one or many version of the Content of the component. The user-given Name of the component may change but not the Key. Different Key means different component.

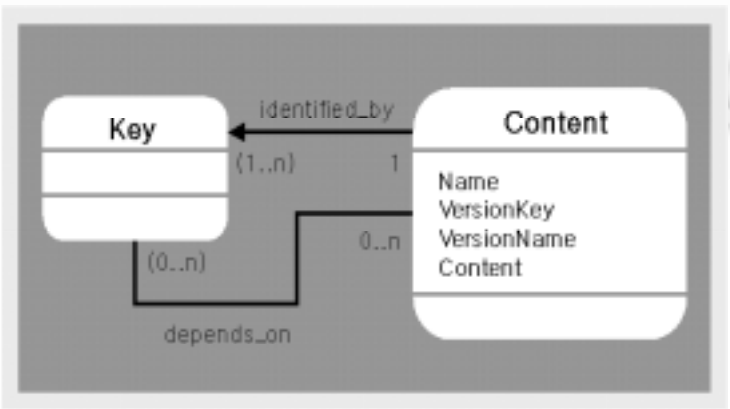

Fig.2 Key and content for component

We are also interested in the quality of component for framework and it needs for us to define a meta-model that manages quality information of components. The quality of a component can only be assessed in its context and it can be achieved by examining referenced components and component itself. If the referenced components can have many versions, there have to be configuration information of these components. So we adopt an explicit set of versions (Contents) of the referenced components. Figure 3 shows an extended model of Fig.2.

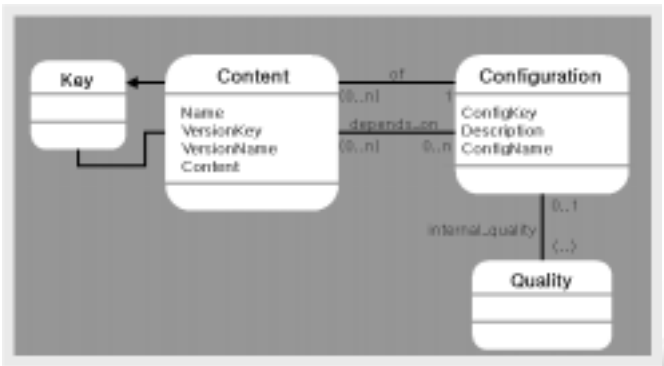

Fig.3 Component management model

3 Designing Framework with Component Management Model

3.1 Web collaboration framework

Web collaboration system enables to communicate each other with chatting, BBS, whiteboard, etc. through web browser. We select Java as target language of framework for platform independent characteristics. The framework provides the base of web service programs and links new applets to web page. We analyze web service domain and application with "3 examples" design pattern that is suggested by Ralph E. Johnson[3]. During analysis process, common aspects of target applications are found and domain knowledge like environment is found. Static model of these things is made in next step "elicitation of common requirement and analysis".

We limit the number of target applications. So our first framework covers only chatting, BBS, whiteboard and Q&A system. If other applications in web collaboration system are required, framework will be extended for that during extension process. Objects and their relationships found during the process will be a base of software architecture in framework. Software architecture provides abstract design and abstract implementation of target application and mechanism defining the collaboration of objects. Software architecture is defined during follow process: objects elicitation, defining role of objects and defining collaboration between objects specially for building common architecture of domain.

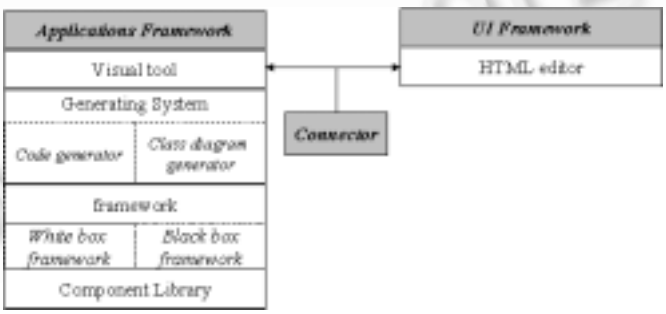

Fig.4 Architecture of Web collaboration framework

Web collaboration framework composed of 3 sub-systems: application framework, UI framework and connector. Application framework is a main engine of framework provides component library and generating system. Generating system generates code for white-box reuse and interface for black-box reuse. Generating system also provides framework knowledge to user with class diagram generator showing objects and collaboration of object and design pattern. Framework users can get the design information of software architecture and the role of

object, which makes them easily use the framework. UI framework enables applets built with application framework automatically attached to Web page. Connector links application framework and UI framework each other.

3.2 Adopting component management model

Figure 5 shows the structure of Web collaboration framework. We categorize components into two different class hierarchies on reuse factor.

Base classes are the template classes of each system and they give the common structure and the control function of the application. We have collected information about analysis and abstracted, designed a common structure of the application in target domain.

Framework user can reuse analysis and design information of target domain while they reuse the base classes of the framework. We divide base classes into domain classes, which are for specific domain application, and non-domain classes, which are common to all application in Web collaboration system. Hot spot classes will be placed in hotspot where modification takes place in base classes. There are many prefabricated hotspot classes in framework component library and framework user can select one of these on his wishes. If there's no class he wants, he can implement new class fitting to hotspot. There are very complex relationships between components of framework and it is difficult to understand and maintain these relationships with non-strategic technique. Component model can be a solution of this problem.

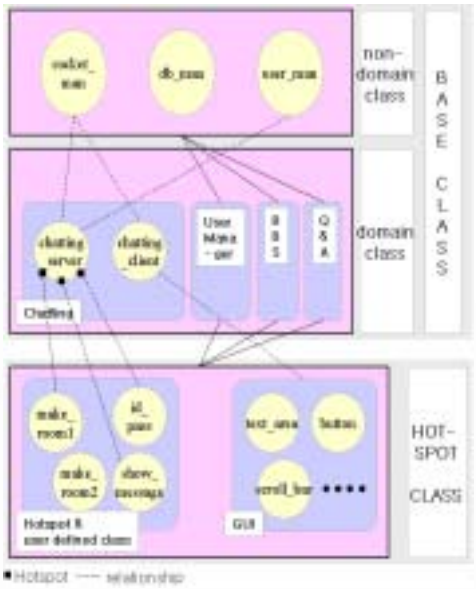

Fig.5 Structure of framework component library

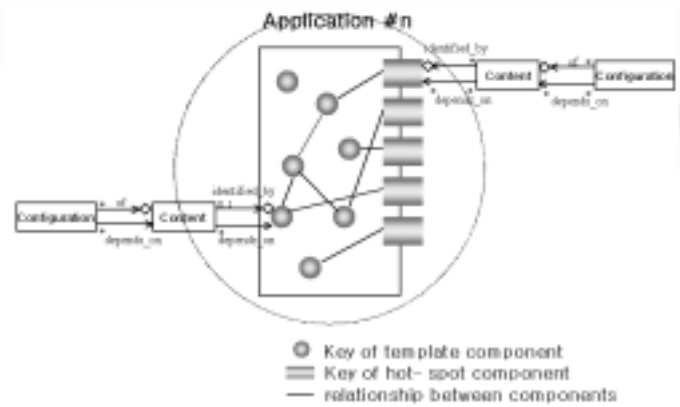

Fig.6 Framework component library

Fig.7 Key object of BBS client system

Every component in framework library has a Key that identifies itself and many Contents contain the context of components. Context means all design information including connection relationship with other components.

Hotspot components of Fig.6 diagram have the same relationship between Key, Content and Configuration. An aggregation relationship the parts (Version and Configuration) depend on the wholes (Component and Version respectively). Template components have just one Content because they will not be changed until framework may be extended and it means that version of template components is not changed. And key of template components may have many “depends_on” relationship with Content of other components due to template components can have several connection with other template components and hotspot components. This mechanism describes the relationship of components in framework component library well.

Figure 7 shows the components and their relationship in BBS system that is a sub-system of Web collaboration framework using component management model.

4 Conclusion

We design web collaboration framework using component management model to maintain the structure of framework and to manage components in framework component library. We experience that it is very complex and difficult to manage dynamically evolving framework in constructing our first framework. We expect component management model to be a solution of this problem. Component management model gives the Key, Content, Configuration information of components in framework that show connected components, context of components and version of components respectively. We also categorize all components into base class that is a template of application and hotspot class that can be extended to user’s need. Component management model based design is expected to increase the usability of framework. It is visual supporting tools that make a framework easily used and component management model will effect to implementation of visual supporting tools like component browsing tool, component configuration tool and hotspot tools that are implemented in our first framework construction. We compare class library, object-oriented framework, and component management model in Table 1.

Table 1 Comparison between related works (class library, object-oriented framework) and component management model

	Class library	O-O Framework	Component management model
Extensibility	Using Inheritance	By overriding or composition through hot spot	By component composition in hot spot
Usability	Understand class library and parent class	Understand hot spot	Understand hot spot or existed component composition
Development cost	Inexpensive	Expensive	Expensive
Productivity	Lower productivity (difficult to understand class library)	Higher productivity (good extensibility and usability)	Higher productivity (advantages of framework and component)
Browsing	Class browsing	Object browsing	Component browsing
Version management	Difficult to manage object	Difficult to manage object	Manage to component

In the future study, we will reconstruct web collaboration framework with this design and implement visual supporting tools to reflect architecture of the reconstructed framework. And we have not verified component management model and it has to be worked also. We expect the verification of component management model will improve the quality of the framework.

References:

[1] Ralph, E.J., Vincent, F.R. Reusing object-oriented designs. Technical Report, UIUCDCS9-1696, University of Illinois, 1991.

[2] Landin, N. Development of object-oriented frameworks [Ph.D. Thesis]. Lund University, 1996.

[3] Hautamaki, J. A survey of frameworks. Technical Report on FRED Project, 1997. http://www.cs.helsinki.fi/research/fred/reports.html

[4] Roberts, D., Johnson, R.E. Evolving frameworks: a pattern language for developing object-oriented frameworks. 1996. http://st-www.cs.uiuc.edu/~droberts/evolving.pdf.

[5] Johnson, R.E., Foote, B. Designing reusable classes. Journal of Object-Oriented Programming, 1988,1(2):22~35.

[6] Harmon, P. Components and objects. Component Development Strategy, 1998,3(7):5.

[7] Camichael, A. Building from components: seeing a unified component model. White Paper of ObjectUk, 1997. http://www.objectuk.co.uk/Papers/Seeking/Seeking.html.

[8] Taligent, Inc. Building Object-Oriented Frameworks. 1994.

[9] Griffin, W.G. Lessons learned in software reuse. IEEE Software, 1995,12(4):11.

一种基于构件管理模型的域框架设计

Sang-Geun Kim

(*Sungkyul University* 计算机工程系,韩国)

摘要: 通过使用构件管理模型来重新设计基于 WEB 的协作系统框架.构件管理模型通过关键字、内容和配置信息获得构件与高级开发知识间的连接信息.构件管理模型可以通过每个构件的关键字和内容信息来表示构件之间的连接关系,它也支持特定应用子系统的设计知识.引入构件管理模型来实现更为容易的框架构件管理以及框架自身更为方便的使用.

关键词: 面向对象框架;构建模型;域框架;框架设计;重用;构件管理

中图法分类号: TP311 文献标识码: A

第 19 届全国数据库学术会议

征 文 通 知

由中国计算机学会数据库专业委员会主办、郑州大学和河南大学承办的第 19 届全国数据库学术会议(NDBC2002)将于 2002 年 8 月 26 日-29 日在中国河南省郑州举行.VLDB2002 将于 2002 年 8 月 20 日-23 日在香港举行,NDBC2002 将借此良机于 8 月 26 日举行 Post-VLDB,邀请国际知名专家做专题报告.

会议宗旨:本届会议将为中国大陆、香港、台湾、澳门和海外华裔数据库研究者、开发者和用户提供一个中华数据库论坛,交流有关数据库研究与应用的成果和经验,探讨数据库研究与应用所面临的关键性挑战问题和研究方向.届时国内外著名专家将到会作专题报告,主流厂商将展示他们的最新技术.会议将评选大会优秀论文和研究生优秀论文.会议正式论文将作为《计算机科学》专辑出版.我们诚征数据管理和应用领域各方面的最新成果以及有关新数据库技术、应用与方法的论文、专题讨论、演示等.

征文范围(但不限于这些领域):Web 与数据库;面向对象与对象-关系数据库系统;移动计算和数据库;WEB 缓冲技术;数据库实现技术;实时数据库系统;数据库安全性;科学与统计数据库;数据仓库和 OLAP;XML 和半结构化数据库系统;数据挖掘和知识发现;空间和时态数据库系统;文本数据库与信息检索;内容管理/知识管理;查询处理与用户界面;工作流/数据库应用;生物与基因信息系统;并行和分布式数据库系统;数字图书馆;多媒体数据库技术;数据集成和迁移;电子商务/电子政务.

投稿要求:(1)论文应是未发表的研究成果,论文应包括题目、摘要、关键字、正文和参考文献.作者信息单独另纸提供,包括论文题目、作者全名、所属单位、电子邮件、通信地址、电话和传真;(2)论文中英文均可,用 Word 软件排版;论文篇幅一般不超过 6 页(A4 幅面);(3)会议论文采用网上提交方式,具体要求见会议网址: http://www.zzu.edu.cn/ndbc2002/

重要日期: 论文提交截止时间:2002 年 4 月 1 日; 论文录用通知时间:2002 年 5 月 15 日; 排版稿件截止时间:2002 年 6 月 15 日

会议信息可以通过访问网站 http://www.zzu.edu.cn/ndbc2002/得到,也可以与会务组联系.

通讯地址: 450052 河南省郑州大学计算机科学系 NDBC2002 会务组

电话: 86-371-7761542(NDBC2002 会务组); 86-371-7763209(郭淑艳女士) 传真: 86-371-7761542 E-mail: ndbc2002@zzu.edu.cn