정보학연구 제12권 제2호 2009년 6월(pp.31-37)
The Journal of Information Technology. Vol.12. No.2

Study on Design and Embodiment of Reference Image Renewal System based on CBSD

Kyung Hun Kim
Dept. of Computer Engineering, KyungHee University

Kyeong Taek Rhyu
Dept. of Computer Information, Keukdong College University.

Jie Young Lee
Dept. of Computer School, SeMyung University

Tae Won Kyung(Corresponding author)
Dept. of Support Generalization, Korea Institute of Industrial Technology

Abstracts: - Component-based software development [CBSD] is a new paradigm of recent software development. CBSD was started from reuse of software and study on software component technique. In order to improve produce and quality of software development, there is study on reuse of components lively and how to develop, share, and manage components effectively. In this study, I would like to suggest a method that is able to simply test a new application system by developing systems developed individually into components, registering them, and combining them.

Key-Words: CBSD, Reference Image Renewal, Component Combination

1. Introduction

Component-based software development [CBSD] is a paradigm of recent software development [1,2]. Also CBSD is a form of evolution of software industry developed for a long time. CBSD is not only a change of developing techniques, but a structural change of software business and the market in overall. It provided a new paradigm for construction of application systems like e-Commerce, e-Business, t-Commerce, and so on. Software development has been done by code so far; it has turned to change to by component. As it is developed into component, it is possible for users

to expect development productivity and development improvement by component reuse[3]. Especially, study on software automatic generation has been done in many fields and been marching to show the best performance of solution that solves particular problems in given condition. Even if componentized in order to reuse component, the environment that extends component should be provided[4]. For reuse of component which develops continuously, there should be an integration environment to construct component easily and reuse[5,17,20].

However, it costs a lot and takes long to develop system independently in image processing field. To solve this problem, an introduction of CBSD is necessary. That is, an integration environment that changes developed algorithms continuously into components and registers them in order to adapt to changing world rapidly. A standardization of data form should be done simultaneously.

It has brought better development productivity and quality improvement by developing a new application system which is extended by developing component and reusing in the integration environment. This paper separates an algorithm, develops a component, and constructs a new application system by assembling the existing component and the newly made component.

2 Related work on CBSD

Most popular defect detecting methods are Test, Peer Review, Walkthrough, and Inspection. In this chapter, differences of those four methods are compared.

2.1 Outline

The purpose of CBSD is to develop a new software system by reusing pre-made components. There are no developing tools based on software architecture that understand interfaces of components, combine related components, and construct a component system. As the result of study on algorithm componentization, there are representative tools such as Khoros, LabView, Matlab, Wit, etc. However, they are not perfect in function and it is difficult to replace components in assembly. In the image processing field, to embody component which implements independently, existing basic algorithms should be componentized. And to use component, it should be added to framework technique.

Also component management provided with newly-made component is needed. In order to provide more powerful functions of component technique, architecture-based generation, assembly, and extraction of component should be done and there have been being studies on applying software architecture technique to component technique abroad[6,7].

2.2 Features

When developing software, you can save cost and time for software development if you use pre-embodied block. On this wise pre-embodied block is called component. Each entity is integrated in component and that makes replacement of software and reuse easy. Service is done only through interface[8]. Component provides developer interface by running unit and hides the detailed inner part and it enables to develop application easily and quickly.

2.3 Framework

Framework makes connection between components easy when a system is constructed. The things needed for developing framework is as followed.

- Abstract Class
- Constraints
- Dynamic code loading

A program based on framework distinguishes the part that does not change from the part that changes and provides a mechanism that applies a change to the part that can be changed.

Framework-based programming defines framework, analyzes the set of applications, and defines and designs architecture by dividing domains and specializing. And in the stage of embodiment, it makes logical design as an actual solution.

2.4. Reuse Technology

Reuse technique of component is classified into methods how to compose and use patterns according to features and rules of components. Composition method is stored at reuse library as a source code of reuse component itself, so when used for new system, it can be used without transformation. When it uses patterns, it has to apply the certain rules and generates source codes regularly according to restriction conditions. Reuse of component is a technique applied to construct similar domain or system by systematically managing used component to develop system and information of the component and knowledge.

2.5. Component Configuration Management

So as to reuse pre-made component for development, use, maintenance of component that is used to construct software system, there should be consistency while component is used. it also has to extend new construction elements of component in addition, keep new dependent relationship, and firm independent construction elements. It manages artificial structures generated in the process of development and assists software development by controlling changes of software and its components, and remembering development [2,6,21-24].

Single component manages the direction of evolution of component, new version, and other components' modifications composite component manages newly-made configuration creation, baseline creation, modification, and branching. However the projects these days are share and processed, so configuration management system is getting complex and it should be run in distributed environment. Configuration management should be completed separated to perform policies with flexibility by separating configuration management.

2.6. Various Types of Component Combination

Elements of combination in the system based on component are component and framework, the combination types of these elements are divide into 3 categories; the combination of components, the combination of component and framework, and the combination of frameworks. They can be divided into 6 small categories again as shown <Table 1>.

<Table 1> Types of Component Combination

Form/ Technology	EJB	COM+	Java Bean	Water Bean	OMG/ Brbos
Component Deployment					
Framework Deployment	Future (container contract)		(JVM plug-in)		
Simple Composition					
Heterogeneous Composition	(IIOP)				(IIOP)
Framework Extension		Future (policy object)			
Component Assembly					

3. Design and Embodiment of Component

3.1 Structure of Component

For reuse, component structure is designed that information of component link is reorganized and run at the point of time of run-time of component call-part when the component is used or run, so it need not be defined in advance at compile-time. That is, as a user uses component call-part, the component is linked and run, and we can see the result.

```
for(int y=src->Get_YStart()+1;y<src->Get_Yend()-1;y++)
 for(int x=src->Get_XStart()+1;x<src->Get_Xend()-1;x++){
     result=src->Get_Pixel(x-1, y-1)+2*src->Get_Pixel(x,y-1)+
       src->Get_Pixel(x+1, y-1)-src->Get_Pixel(x-1, y+1)-
       2*src->Get_Pixel(x, y+1)-src->Get_Pixel(x+1, y+1);
     if(result>258)dst->Set_Pixel(x,y,258);
     else dst->Set_Pixel(x,y,result);}
```

Figure 1. Embodiment Code approaching Data Form as an Object

This kind of component execution can be developed by component and component call-part independently and enables to modify a particular component continuously, it can minimize the developing period. <Figure 2> shows a simple example how to arrange each component.

That is, a simple application program component is generated by arranging component sub made by describing just algorithm and read and write which are existing components. The result

generated from each component goes to the pointer and is run, so it makes the complex of component structure minimum and the simplified component structure is arranged.
Also to approach data easy, it has been embodied by using general classes. <Figure 1> shows the code that directly approaches data and object generated from class.

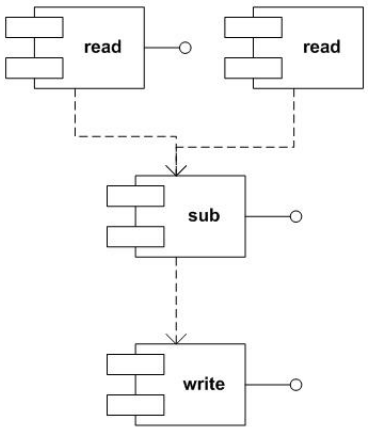


Figure 2. Component Arrangement and Connection Structure

<Figure 3> shows the whole structure of all information needed for component operation and reuse; the actual component execution files' names related to component execution, parameters and related information, initial value, directions of input/output, icon information, etc.

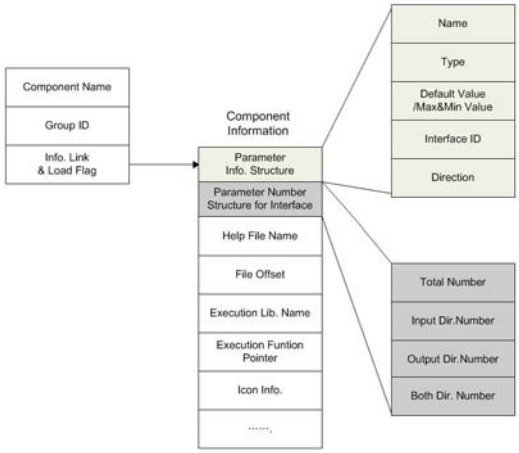


Figure 3. Component Information Structure

3.2 Configuration Management

It is composed to keep history information of source code file with the method of configuration management centered of workspace file which the logical relationship is applied to configuration management structure. All information of each component is stored at upper workplace file and configuration server generates hyperlink information, registers, and manages. It also has a branch function that has link information which enables to keep existing configuration management history. That is, inter-mediate products generated in the process of component development can be found out to be right component in other application system. Due to this feature, it has to have a structure that registers new components.

3.3 Embodiment and Result

3.3.1. Component Development Environment

In this paper I have developed a component, assembled the component, and embodied it partly in the system. Hello Vision2000, the developing tool used for embodiment, is a CBD development tool [11,12].

3.3.2. Internal Composition Elements

Main internal composition elements are visual programming environment, automatic function interface generation system, online function database, resource management system, hardware and operation management system, etc.

-Visual Programming Environment
Support Drag&Drop
-Automatic function Interface Generation System
Automatic registration of interface
-Algorithm Information Management System
Management in form of DLL:Dynamic Link Library
-Resource Management System
Support Management system
-Hardware and Operation Management System
Easy Portability to Hardware and Operation system

3.3.3. Component Registration and Use Procedure

It uses workspace provided and uses Drag&Dropd from the function database management window as shown in <Figure 4>, and generates user-defined component by using Visual C++.

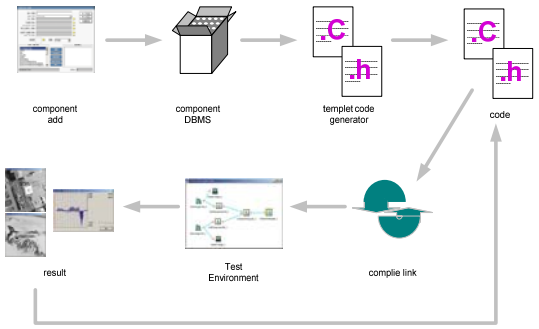


Figure 4. User Component Registration Procedure

3.4.4. Embodiment

I used Hello vision2000 and Microsoft Visual C++, component integration development tools, to develop component, registered, and then assembled with existing component and finally embodied a part of extraction system of moving object.

$$D(K,x,y) = I(K,x,y) - B(K,x,y)$$

$$B(K+1,x,y) = \begin{cases} B(K,x,y) + aD(K,x,y), & if \quad D(K,x,y) > T \\ B(K,x,y) + bD(K,x,y), & otherwise \end{cases}$$

Expression 1. lowpasss filtering expression

In order to compare the algorithm proposed that system uses lowpass filtering algorithm, a part of existing system, there should be two systems developed in two independent environments. For the text environment for performance evaluation, there must be a comparison of two systems. However, if each system will be embodied as component, only the component to be compared can be embodied. If embodied component itself is assembled and compared, a lot of money and time will be saved.

$$T(K,x,y) = \begin{cases} I(K,x,y), & if \quad D(K,x,y) < T \\ R(K,x,y), & otherwise \end{cases}$$

$$B(K+1,x,y) = Median\left[T(K-w,x,y),...,T(K,x,y)\right]$$

Expression 2. Proposed Method

Expression 1 expressed the algorithm of lowpass filtering as an expression. In the expression above, I(K, x, y) and R(K, x, y) are the value of pixel of input image at the time k and (x, y) of reference image and a and b are meditation factors and decide the result. D is Difference Image and T is a critical that separates moving object and background.

Expression 2 represented a proposed algorithm of Reference Image Update for Moving Object Detection in the Outdoor Scene. in the expression above, T(K, x, y) is a temporary image to get Reference Image to present image, if present Difference Image D(K, x, y) is smaller than critical, it gets the pixel of present image, otherwise, it gets the pixel by allocating R(K, x, y) of Reference Image.

Algorithm parts of two systems were embodied as components and the performances were compared and evaluated. If they were developed and evaluated with the existing method, more money and time would need to develop independent systems. I in this paper developed as component and assembled.

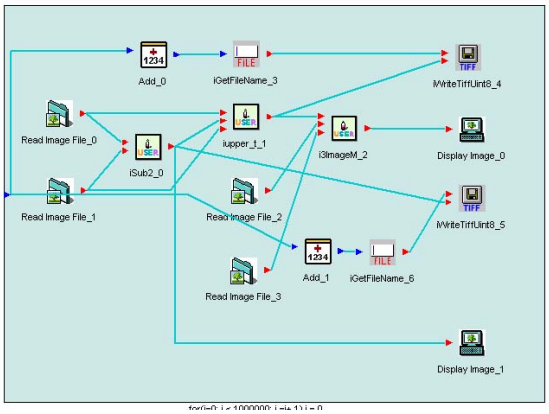
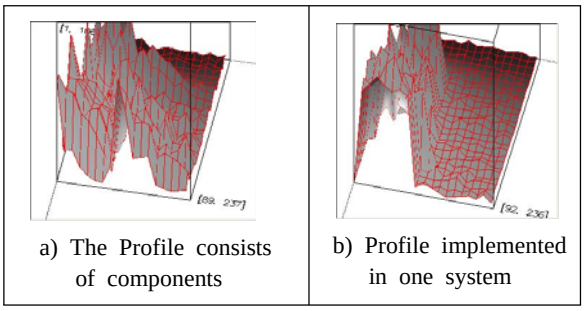


Figure 5. Workspace of Hello vision 2000

<Figure5> <Figure6> shows a component system that it arranged a developed component and an existing component onto workspace and assembled. This composite component system may be arranged as a part of another bigger system. Once developed component should always be registered and manage d to reuse. It also provides GUI environment, so it shows information of component or functional addition or error in visual. In the case of functional addition of developed component or change of attributes, it can be modified by using attribute interface of component.

I in this paper embodied reference image renewal system, a part of moving object detection and tracking system, for moving object separation as a component [13]-[16].

It is developed as a component unit to assemble like a part to the system, so it can have a flexibility to be applied to a new system and shorten development time and have good quality.



a) The Profile consists of components

b) Profile implemented in one system

Figure 6. 3D profiles

4. Conclusions and Further Research

It is important to understand component because the paradigm changes to generate component, assemble, and produce software.

Better performance and quality should be kept when the existing components are reused, and on the web environment assembly of component and arrangement should be more flexible.

There also should continue study on configuration management of component. That is why it faces newly-composed information according to keeping system maintenance and mutual operation by considering newly-required matters.

The study that supports the stage of retrieval, arrangement, combination, run for component on the web service should continue further. Service with assembly of Web Service Component is within the CBD, Hence, development of business component should be studied too in the future.

References

[1] M. Aoyama, Componet-Based Software Engineering : Can It Change the Way of Software Development? In Proceedings of the International Conference on Software Engineering, Vol II, April. 1998.

[2] Lu Zhang, Hong Mei, Hong Zhu.:A configuration management system supporting component-based software development. Computer software and Application Conference, 2001. COMPSAC 2001. 25th Annual International 8-12 Oct. 2001.

[3] S. Yau, N. Dong.:Integration in Component-Based Softwqre Developoment Using Design Patterns. Proc. Of the 24th COMPSAC'00, 2000.

[4] F. Yang, H. Mei, and K. Li.: Software Reuse and Software Component Technology, Acta Electronica Sinica, vol. 27, no. 2, pp. 68-75, Feb. 1999.

[5] J.H.Lee, Y.T. Cho, H. Heo, Oksam Chae.:Integrated Development Environment for Digital Image Computing and Configuration Management. spimger-Verlag Lecture Notes CS, Vol.3488, pp.20~29, Apirl 5005.

[6] A. Van der Hoek, A. Carzaniga, D. Heimbigner, A.L. Wolf.:A tested for configuration management policy programming. Software Engineering, IEEE Transactions on Vol 28, Issue 1, pp. 79-99. Jan. 2002.

[7] A.W.Brown and K. C. Wallnau.:Engineering of Component-Based Systems. Component-Based Software Engineering. IEEE CS Press, pp.7-15, 1996.

[8] James D. Palmer, N. Ann Fields.:Computer Supported Cooperative Work. Computer, Vol.36, 1994.

[9] David Garlan, Dewayne Perry.:Software Architecture:Practice, Potential and Pitfalls. Proceeding of the International Conference on Software Engineering '96, 1996

[10] J.Siegel. CORBA:Fundamentals and Programming. John Wiley, 1996.

[11] Jungheon Lee, YoungTak Cho, Hoon Heo, Oksam Chae.:MITES:Visual Prgramming for Teaching and Research in Image Processing., Springer- Verlag Lecture Notes in Computer Science, Vol. 3514, pp. 1035-1042, April 2005

[12] D. Box. Essential COM. Addison-Wesley, 1999.

[13] W.Long, Y.Yang.:Stationary background generation: an alternative to the difference of two Image. Patten Recognition 23 pp. 1351-1359, 1990

[14] M. Boninsegna, A. Bozzoli.:A tunable algorithm to update a reference image. Signal Processing:Image Communication16 pp.353-365, 2000

[15] K.Karmann, A.Brandt, R.Gerl.:Moving object segmentation based on adaptive reference image. Signal Processing 5, pp.951-954, 1990

[16] K.Skifstad,R.Jain.:Illumination independent change detection for rear world image sequences," Comput. Vision Graphics Image Process 46, pp.387-399, 1989

[17] H. Mili, F. Mili, and A. Mili.: Reusing Software:Issues and Research Direction, IEEE Transactions on Software Engineering Vol. 21, No. 6, pp.528-561, June. 1995.

[18] M. Kilger, T. Dietl.:Interpretation-drive low-level parameter adaptation in scene analysis, in:F.Picher, R,Moren-oDiaz (Eds), Comput.Aidid Syst. Theory. EUROCAST' 93, spri0nger, Berlin, pp. 380-387, 1993

[19] Christo Ridder, Olaf Munkelt, Harald kirchner.:Adaptive Background Estmation and Foreground Detection using Kalman-Filtering. Bavarian Research Center for Knowledge-Based System Orleansstr. 34, D-81667 Munchen, Germany.

[20] Magnus Larsson, Lvica Crnkovic.:Configuration Management for Component-based Systems. In Software Configuration management – SCM 10, 2001

[21] Axel Mahler, Andreas Lampen.:A toolkit for software configuration management, 1988

[22] R. Conradi and B. Westfechtel.:Configuration Versioned Software Product. SCM-6 Workshop. Pp. 88-109. Spring LNCS 1167. Berlin, March 1996.

[23] S. Dart.:Concepts in configuration Management Systems. Proc. of the 3rd. Intl . Workshop on Software Configuration Management. Trondheim, Norway, june, 1997

[24] J. Estublier.:Workspace Management in Software Engineering Environment. SCM-6 Workshop. Springer LNCS, 1167. Berlin, Germany, March 1996