Pakistan Journal of Information and Technology 2(1):1-4, 2003

SuperCube as a Model for Component Classification

Xu, Zheng-Quan, Hamza H. Njapuka and Yang Xu
School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, People's Republic of China

Abstract: Classification is a general term that has drawn much attention in today's market for software industry. It is a categorization that explains a group of entities with the particular behaviors of those entities. So far, researchers have done works in several areas of classification, and seminars have been conducted. In this paper, we propose a formal conceptual framework to provide dynamic and flexible way to articulate proper strategies in identifying and classifying components in a large fused reuse bank. We call this as SuperCube Model (SUCUMO) to describe the model-class components fused in a huge multi-dimensional cube. The idea here is to use SuperCube Model to describe different components for storage, based on their specific criterion or attributes; analyze, identify and classify them making ease access, browsing and future retrieval. Our approach is based on both top-down and bottom-up procedures to let user choose an appropriate way to easily locate a component in need.

Key Words: SuperCube; Component Classification; Reusable Component; Reuse Bank

Introduction

Component Classification (Njapuka, 1999) is an important technique in software engineering. In a huge fused bank of components, searching for a candidate component will be extreme difficult if there is no proper designed classification scheme for those components. Such scheme will help in organizing the large fused components into semantic groups and hence creating room for user to locate a possible component with no pain.

SuperCube is a multi-dimensional aggregate of all the cube components. Thinking of the total components in the big bank of components, you might get tired from searching as long as you don't get or find a proper way of locating a single component in need in a fused bank of components. One needs to consider the various facets of the problem. This requires tangled net relationships, or an orderly scheme that will enable you to navigate the bank with ease and confidence. SuperCube is a powerful multi-dimensional components bank (Dash and Agrwal, 2001), whereby it's representation is done by visualization. Since the material constitutes of many cubes set in n-dimensions, the SuperCube is rotated to *x, y, z* dimensions to create room for expansion and clear visualization.

From multiple dimensions of the ad hoc components, one can acquire variables and attributes from each aggregate in a systematic way. Ad hoc components behaviors are made in such a way to support multi-combinations and multi-divisions of components variables.

The Architecture of SUCUMO: Although the SuperCube component has many attributes, but some of these attributes when reuser retrieves the component (Jefrey and Iglesias, 1993), might make this reuser not to be interested, also not too possibly retrieves through them. Attributes are used in classification (Njapuka, 1999; Jefrey and Iglesias, 1993; Rosch, 1983) to represent categorization or classification facets (Jefrey and Iglesias, 1993; Diaz, 1991; Milli *et al.,* 1997), as in Prieto Diaz's Multi Faceted Categorization of Components (Diaz, 1991; Diaz and Freeman, 1978). The surplus component attributes has: component name, maker, application environment, application domain, usage, functions, representation, shape, and abstraction level. The user has the possibility to retrieve (Ye *et al.,* 2000) the component through these attributes. In these attributes, the components name, maker, usage, shape can be used by the reuser when searching for component and not that user must understand the component's attributes when searching for candidate component. Semantic relationship should be applied as in (Milli *et al.,* 1994).

Consider two different models of components, as per Fig. 1 and Fig. 2, say text component and graphical component. These can be presented in the form of their determinants (*Need* and *Spec*) and dependencies associated with their variables.

Component:

Text { Need
Spec ...

here we have the following:

Need	
Spec	
Arch	
Code	

Decision Tree Analysis: Decision Tree Analysis is an analysis where possible outcomes of some condition are represented as branches, which may expand to produce other branches.

Decision tree analysis can be applied in killing the SuperCube aggregate to determine relationships between the components. In articulating strategies to identify and classify components in our model, the system modeling is done according to the architectural phasing procedure to ensure the production of quality software systems (Mei *et al.,* 2001; ISO, 9000) with elements of continual development, and the analysis here should be carried out with care to ensure that there is no broken relationship among the dimensions during the SuperCube operation.

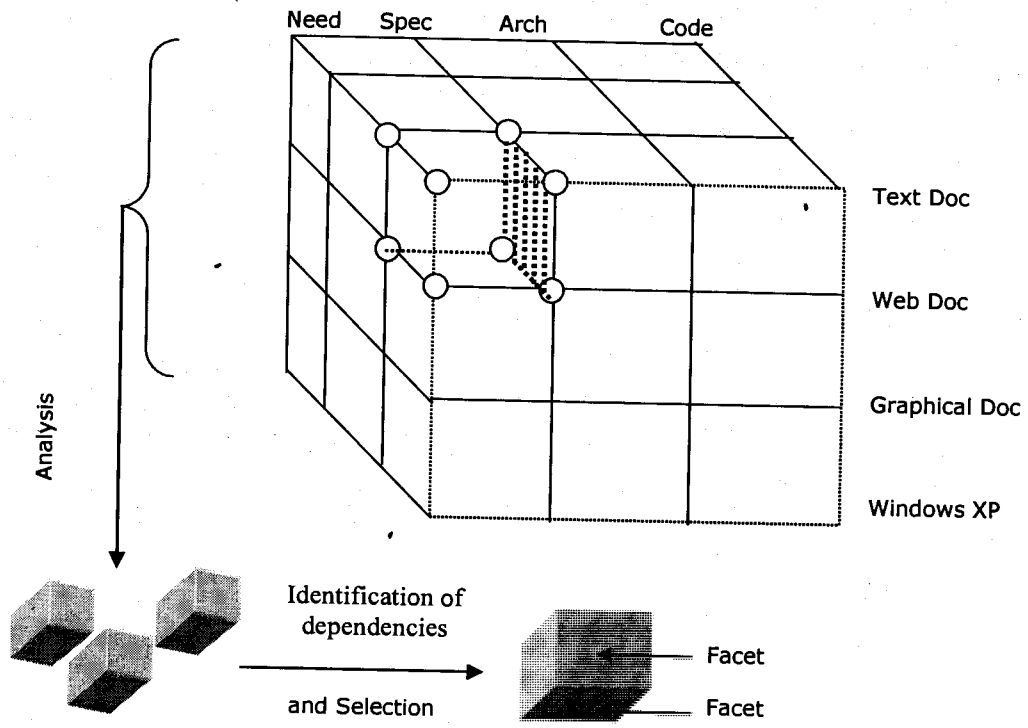


Fig. 1: SUCUMO Architecture

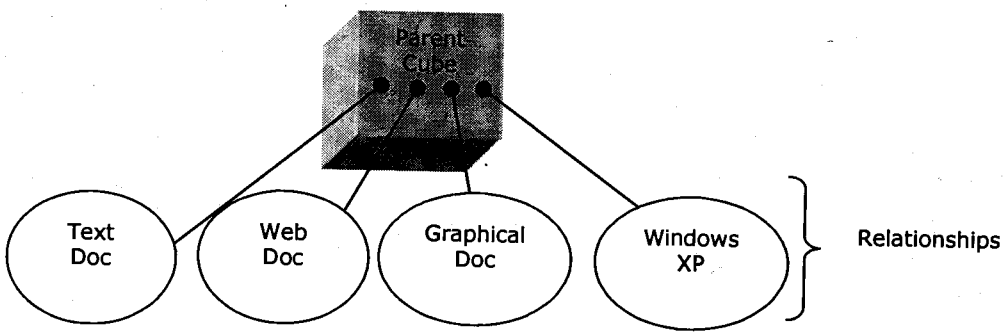
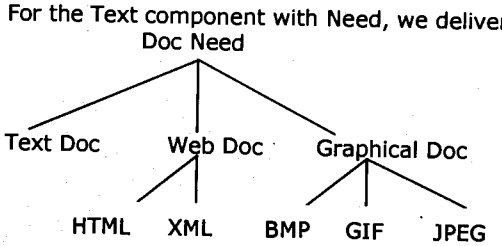


Fig. 2: Decision Tree

For the Text component with Need, we deliver....



Analysis operations take place at architectural phase where the components are being analyzed with all pre-conditions. After analysis is done, components are then identified each given a specific class identity, classID, say, XML component given a unique classID that makes it to be a member of Web Doc family, which has already been provided with a separate particular classID to belong to the Doc Need society. The analysis here is reversible, which means it can take place using either top-down or bottom-up approaches.

Assuming the new node has been created from the root of the SuperCube with all attributes and specifications, while preserving relationships

for k=0 to n–1 do
Tree=OldNode+NewNode
OldTree=Tree
NewTree=MakeTree (NewNode, OldNode)

NewClass=MakeNewClass (ClassID$_i$, ClassID$_j$)
ClassID$_i$=Class (i–1)
ClassID$_j$ =Class(j+1)
CompLocale=Class (*Need,Spec*)
ClassLocale=NewLocale (*Need,Spec*)
NewLocale$_k$=NewLocale$_{i,j}$–OldLocale$_{i,j}$

SuperCube Classified Component (SUCCO) Scheme: In SUCCO scheme, the component reuse bank administrator through SUCUMO with corresponding terms may combine facets with corresponding variables to establish complex relationship between components. At abstraction level SUCCO is easy to revise. Since modifying one SUCCO cannot affect the other, simultaneously each SUCCO corresponds to one structural term-space to avoid keywords classification chaotic.
In component bank, each component all has one group of attributes. These attributes include components manufacturer or supplier. Some are the components bank application environment. We can see that components attributes and SUCCO are somewhat similar. But the two have the difference:

- SUCCO is the component attribute's one subject. The attributes major part is that it has either component manufacturer or component supplier. Component bank administrator set rules that stipulates SUCCO, and neither the component manufacturer nor the supplier need to understand them.
- Component reuser's most interested components' attributes when searching for candidate component is SUCCO. When searching for component, reuser can explicitly define component category without omitting component.

Since our model, as pointed earlier, is an ad hoc aggregate of components, the arrangement is done in accordance with the multi-database hierarchies. It supports double hierarchy property in which consistency for both properties top-down and bottom-up is created and preserved.
SUCUMO, when enabled, can be used to navigate the components behavior from multiple-dimensions, thus, enhancing performance behavior analysis in lexical form to determine ad hoc components hierarchical relationships.
Killing the Skeleton: Killing the Skeleton is a division procedure that cut the big skeleton into small components called modules in a hierarchical ordering. This is top down approach. These modules tend to combine like components together and further arrange the components into linear sequence of relationship called classes. Each class has its own set of descriptions.
Regardless of the components origin, provided that it enters the reuse bank, all should undergo routine procedures to check for the specific requirements based on their criterion or attributes. They should conform to certain standards such as flexibility, intangibility, and integrity. If these standards are accurate, the components are then stored in accordance to their specific criterion.
Component reuse bank should carry out the confirmation process to the component to guarantee that the component meets certain quality specifications, classifying the component and give descriptions for each classified component. When all these are done, it then stores the component already classified for future retrieval. The component reuse bank should also create a set of procedures for retrieval to enable the user locate the possible component without difficult.
The user searches for candidate component through the special retrieval procedures provided by the component bank.

Algorithms

In this model, components are manipulated in multi-dimensional level. Applying vector analysis, we can twist a cube component in x, y, z dimensions to locate or to store component already classified by the proposed scheme.

for k=0 to n–1 do
KillNode(x,y,z)=destroyNode(i,j)
MakeNewNode= MakeNewRoot(x+i)*(y–j)
= MakeNewRoot(x–i)*(y+j)

For all cube components C_c in a huge fused multi-dimensional block, *SuperCube*, S_c , the variables of components C_v are said to be in a set of multi-dimensional arrays of hierarchies such that i, j, and k objects are treated to correspond with x, y and z dependencies respectively.

$[i'(x) \cap j'(y) \cap k'(z)] \Rightarrow S_c$
$\forall(x,y,z)[C_c(x,y,z) \Rightarrow C_c(i,j,k)]$
$C_v(i,j,k) \neq 0$

Consider Fig. 3 whereby a cube component derived from a SuperCube with dimensions in vector form is said to be a state of visualization from which the variables *a, b, c, d, e, f, g* and *h* totaling 8 for one cube form facets viewable in different angles, say variables *a, b, c* and *d* form facetTHREE, variables *a, e, g* and *c* form facetONE, and variables *c, d ,g* and *h* form facetTWO.

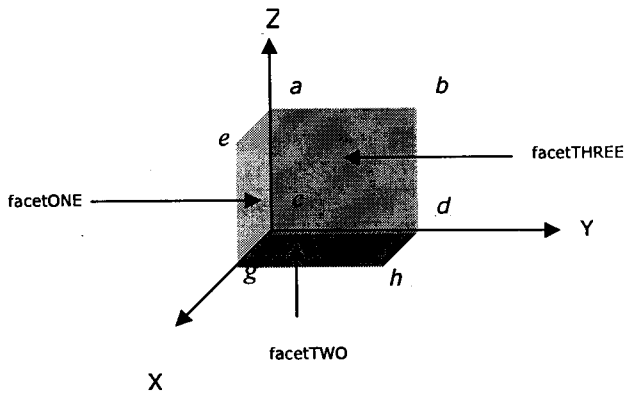


Fig. 3: Vector Dimensions

In visualizing the cube, we need to rotate it in either clockwise or anti-clockwise direction, whatever the direction we choose; we need two facets at a time to perform this operation. Taking vector X dimension for example, we need facetONE and facetTWO in a form of matrix as per below:

$$\text{facetONE}=\begin{pmatrix} e & a \\ g & c \end{pmatrix} \quad \text{facetTWO} = \begin{pmatrix} c & d \\ g & h \end{pmatrix}$$

These two form double matrix for X rotation.

$$\text{i.e } X = \begin{pmatrix} e & a \\ g & c \end{pmatrix}\begin{pmatrix} c & d \\ g & h \end{pmatrix}$$

The same is applied for other directional rotations for Z and Y axes

Row displacement (R_N): Let R_N be an n-vector.

for i=+ve Disp rowONE=NEWrow(x+1)
i=–ve rowONE=NEWrow(x–1)
i=N row N=NEWrow($x \pm N$)

$$\Rightarrow \text{NEWrow}(R_N)=\sum_{j=1}^{n}[(x_j+N_{i+1})*(x_j-N_{i-1})] \qquad (1)$$

We have zero displacement when x=N

Column displacement (C_N):

for j=+ve Disp columnONE=NEWcolumn(x+1)
j=–ve columnONE=NEWcolumn(x–1)
j=N column$N$=NEWcolumn($x \pm N$)

$$\Rightarrow \text{NEWcolumn}(C_N)=\sum_{i=1}^{n}[(x_i+N_{j+1})*(x_i-N_{j-1})] \quad , \qquad (2)$$

±'s are treated according to *Winograd's algorithm.*

Assuming we have square matrices;

Let facetONE=($f_1,f_2,\ldots,f_n$) and facetTWO=($ff_1,ff_2,\ldots,ff_n$) denote vectors having n components in each vector.

facetONE.FacetTWO is the *dot product* of the two matrices.

$$\Rightarrow \text{facetONE.facetTWO}=\sum_{i=j=1}^{n}[(x_j+N_{i+1})*(x_j-N_{i-1})]$$

facetONE(x,y,z)=facetTWO(x,y,z)= $xyz = N^n$ for all upper-bound multiple-dimensions.

from (1) and (2):

for k=0 to n–1 do

$$C_v=\sum_{k=1}^{n}[(i'(x)*j'(y)*k'(z)) - \text{row}N_i - \text{column}N_j]$$

SuperCube, $S_c = S_{ij}$

$$S_c = S_{ij} = \sum_{k=0}^{n-1}(\text{facetONE}_{ik}\text{facetTWO}_{kj}) - \text{row}N_i - \text{column}N_j$$

$$= \sum_{k=0}^{n-1}(R_N)_{ik}(C_N)_{kj}$$

SUCUMO Characteristics: The SUCUMO is characterized by five attributes namely as application environment, application domain, functionality, abstraction level and representation.

- **Application Environment:** Application Environment used by SuperCube component uses (including understanding, assembly and modification) hardware and software platform. In the SuperCube component bank, all components must rely on to the certain usage of environment to make them reusable. Even if source code is somehow compatible to the component, still must rely on the specific compiling System.
- **Application Domain:** The component's Application Domain is within the component's originality which uses (including its sub-domain) name. In the components bank, any component, all have one suitable domain.
- **Functionality:** Component's functionality is adopted by component in the original or the possible software system providing software function collectives. In the components bank, any component must provide one or more kinds of functions.
- **Abstraction Level:** This is the principle of ignoring aspects of a problem domain that are not relevant to the current purpose in order to concentrate more fully on those that are(Norman, 1996). These include system analysis, design, coding etc.
- **Representation:** ,Component representation method is used to describe component content language form or media if the source code component uses programming language etc. All components in the components bank should have a certain form of representation called *"Context Environment Direct Correlation"*.

Conclusion

SuperCube is a model framework that provides suitable environment and proper strategies in identifying and classifying components in a large fused reuse bank. Different components based on individual specifications can be analyzed, identified and classified and then stored in a reusable bank of components. The process is a reversible one; therefore a big block containing small cubes can also be demolished by means of identifying like components and hence classify them in an acceptable convenient order, thus simplifying components browsing and searching.

References

Dash, A.K. and R. Agrwal; 2001. Dimensional Modeling for Data Warehouse. ACM Sigsoft Software Eng. Notes. Vol. 26 No. 6.

Diaz, P.R.; 1991. Implementing Faceted Classification for Software Reuse. Communications of ACM. 34, pp 88-97.

Diaz, P.R. and P. Freeman; 1978. Classifying Software for Reusability. IEEE Software, 6-16.

ISO 9000, ISO 9001, ISO 9002, ISO 9003, and ISO 8402. A series of International Standards related to quality Sys., International Organization for standardization. Geneva Switzerland.

Jefrey, S.P. and K. P. Iglesias; 1993. Experiences with a Faceted Classification Scheme in a Large Reusable Software Library (RSL). 17th Annual International Computer Software and Applications Conference", Phoenix, A2. 3-5 pp 90-99.

Milli, H., E. Ah-Ki, R. Godin and H. Mcheik; 1997. Another nail to the Coffin of faceted Controlled-vocabulary component classification and retrieval. In the Proceedings of the ACM SIGSOFT Symposium on Software Reusability. Vol. 22 No. 3.

Milli, H., O. Marcotte and A. Kabbaj; 1994. Intelligent Component Retrieval for Software Reuse. In Proceedings of the Third Maghrebian Conference on Artificial Intelligent and Software Eng. 101-114, Rabat, Morocco.

Njapuka, H. H.; 1999. Software Components Classification and Catalog. B.Eng. Thesis, School of Comp. Sc. and Tech., Huazhong Univ. of Sc. and Tech., Wuhan, China.

Rosch, E.; 1983. Prototype Classification and logical classification in the two Sys. In New Trends in the Conceptual Representation: Challenges to Piagete's Theory; Erlbaum, Hillsdale. NJ

Norman, R.J.; 1996. Object Oriented Sys. Analysis and Design. Prentice Hall/ Tsinghua Univ. Press-Beijing.

Mei, H., L. Zheng and F. Q. Yang; 2001. A Software Configuration Model for Supporting Component-based Software Dev. ACM Sigsoft Software Eng. Notes, Vol 26 No 2.

Ye, Y.W., G. Fischer and B. Reeves; 2000. Integrating Active Information Delivery and Reuse Repository Sys. In he Proceedings of the ACM SIGSOFT FSE-8. 60-62.