Improve Software System Visualization using Genetic Algorithm

Haw-Ching Yang and Cheng-Da Chang

Institute of System Information and Control, National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan, R.O.C. Email: hao@ccms.nkfust.edu.tw

Abstract

Well visual effects for system facilitate user to overall master the system relations. To enhance the current visual effects, this work proposes a concrete mapping and software metrics to reveal the relations' characteristics such as size, association and coupling. To adapt user's dynamic viewing changes, the genetic allocation algorithm with the allocation criteria are developed to place the concretes by viewing constraints. In addition, JOGL is adapted to realize the animation functions. Finally, the cases study show that the distinguishability of system is improved.

Keywords: Genetic Algorithm, Visualization, Software Metric.

I. INTRODUCTION

With growth of software systems' functions, users get increasing burden to understand and master relations of the systems. To ease the burden, effective visualization for the relations of systems' structures and relations has proposed one of research topics [1..5]. However, these visualizations were limited by the corresponding static algorithms, which still failed to display the relations adaptively by user's view with well human factors guaranteed.

In early time, Seesoft [1] expressed source codes of a system by a two-dimensions mapping to show structures of the source codes. For constraints from the few expressions of Seesoft, sv3D [2] extended it and proposes a three-dimensions mapping to display the relations, which improved the visual effect. Currently, even though UML is popular accepted in software application domain, due to limited by object manually allocated, there still exists complicated and confusing relations from the 2D allocation. However, using the associated 2D arrangement method [3] for object allocation that was insufficient for the increasing complicated degree of visualization. Hence, applying 3D to show the relations of software system becomes recommended [3][4][5].

To allocate objects in 3D, spring-embedded algorithm [4] and approaching integrated GA method [6] were significant methods to find the arbitrary positions by objects. The visual effects of these two studies were better than before; however, they lacked dynamic viewing to reveal the further relation characteristics such as cohesion [7] and coupling [8] in a system. Because cohesion the number of private attributes and methods represents the complex degree of one object, and coupling the number of imports and exports of public attributes and methods between objects expresses complicated degree between objects. Among the relations of system, the two characteristics decide the effect of visualization, crucially [9]. 3D visualization for object's relations should include not only human effects but also system characteristics [10][11].

Therefore, we propose the following methods to improve the display issues of object's relations:

- 3D mapping of object structures to visualize the relations;
- Criteria for judgment to fit dynamic viewing requirement;
- An algorithm for allocation to reveal the relations' characteristics; then,

the 3D concrete tool is proposed to generate interactive 3D objects supporting user observation and improving relations searching efficiency.

The rest of this paper organizes as follows. Section II explains the design of system workflow. Section III presents data analysis of case study. The finial section states the conclusions of this paper.

II. SYSTEM WORKFLOW

Essentially, this research proposes a visualization tool to adapt to users' viewing changes and expose the system relations. After retrieving the relations of objects into the corresponding files, the system workflow for the tool as

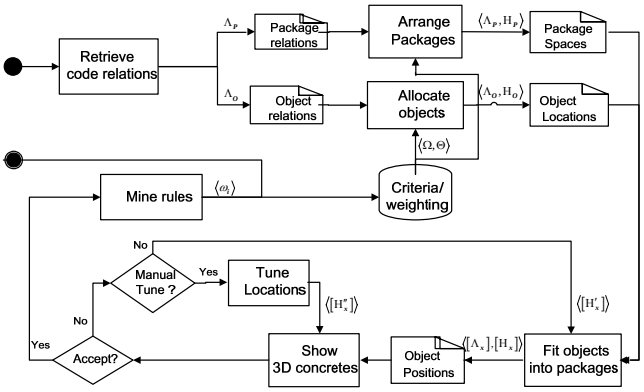


Figure 1. System workflow

shown in Figure 1, the proposed GA^2 (Genetic Allocation Algorithm) is applied to decide the locations, finally JOGL is adopted to realize the concrete objects. The follows define the variables used in the workflow:

$\Lambda_x = \langle n_x, \tau_x, S_x \rangle$: set of concrete *x*,
n_x : total number of *x* (={*c*, *p*, *e*, *o*}),
$\tau_x = \langle o_{x,i}, r_{i,j} \rangle$: relations of *x*,
$S_x = \{ s_i^x \mid i \in \tau_x \}$: dimension set of *x*,
$r_{i,j}$: relation strength between *i* and *j*,
H_x : location set for all *x*,
$\Omega = \langle \omega_i \rangle$: weighting set of allocation criteria,
$\Theta = \langle \theta_i \rangle$: allocation criteria set.

After collecting sources codes, class diagrams, interfaces or ER models, we can retrieve their relations τ_x to represent the concretes including class, package, entity and object represented by *x* with *c*, *p*, *e* and *o*, respectively. Let set Λ_x with relative information of concretes that supports allocation algorithm GA^2. In Λ_x, relation strength $r_{i,j}$ of a concrete represents the relative characteristics with others, and a procedure to find its value will be discussed later. Before relative allocation of the concretes, we first define the objective function *J* combining the criteria with weighting and viewing to find the near optimal allocation H_x and satisfy

$$\min J(=\sum_{k=1}^{r} \omega_k \theta_k). \quad (1)$$

Hence, through package arrangement and object allocation with minimization of the *J*, relative locations H_x of concretes now can be derived.

To locate the concretes in 3D, the relative locations H_x is insufficient for lacking reference points for each concrete. Therefore, we first allocate the relative package space H_p by package, and then shift the concrete object location H_o according to its corresponding H_p to find a feasible location matrix $[\mathrm{H}_x]$. Finally, according to $[\mathrm{H}_x]$, the proposed tool generates concretes and allocates positions by their corresponding elements of $[\mathrm{H}_x]$ in 3D environment. Next, we will describe the system workflow in details.

Concrete analysis

For the objects' overlaps blocking available viewing ranges, we define concrete mapping rules and four allocation criteria to release the viewing constraints. If the concrete target is described by the object-oriented modeling [8], such as class, package and object the mapping rules then are specified as follows:

- Class or object: represented by sphere whose size is proportioned to LOC (Line of Codes).
- Package or class groups: depicted by cube that includes the associated classes and objects.
- Relationship: represented by *r* and shown it using cylinder whose diameter is determined from the following relationship definitions [8]:
  a) CAIC (Class-Attribute Import Coupling): counts import attributes which one class refers to other classes'.
  b) CAEC (Class-Attribute Export Coupling): accumulates export attributes which one class provides to others.
  c) CMIC (Class-Method Import Coupling): is similar to CAIC, but it counts number of import methods.
  d) CMEC (Class-Method Export Coupling): likes CAEC; however, it sums export methods.

If the concrete target comes from E-R model, the entity is retreated as object and their key relations are served as relationships; in contrast, foreign key belongs to CAIC and primary key being referred is classified to CAEC. According the mapping rules, set Λ_x will denote the further relation characteristics from the collected targets.

Allocation criteria

In a same viewing space, increasing relations among concretes will decrease distinguishability for their complex and complicated intersections. To distinguish different concretes, we propose four object allocation criteria as follows:

- Allocation space & segment sum

If θ_1 is the allowed number of space point, N_l is maximum allowed layers, maximum allocation cardinal per layer N_a, then we have

$$\theta_1 = N_l \times N_a{}^2. \quad (2)$$

In a limited visual space, toward (2) minimization can limit the space explosion. Similar to θ_1, let θ_2 be sum of relation segments that represents corresponding relation distances among concretes. Reducing θ_2 make visual effect better. However, there is trade-off between both criteria and distinguishability.

- Sum of space and projecting intersection

For influenced by viewpoint, projective intersection will confuse unrelated with correlated segments. To reduce the intersection by viewpoint, we assume that θ_3 represent the total number of space intersections at the specified view. To reduce misunderstanding of relation between concretes, constraining θ_3 is one of solutions. Similar to space intersection, because the plane vertical to the view called projecting plan filling with intersections of projection, that is hard to avoid. Suppose θ_4 stands for sum of the projecting segments whose distances are shorter than a given range. Then as we know, smaller θ_4 gets better viewing effect.

With growth of system, relations of the system become complicated and incomprehensible. To achieve better distinguishability and well viewing effect, next we will combine the four criteria with suitable weighting ω to satisfy the objective function *J*.

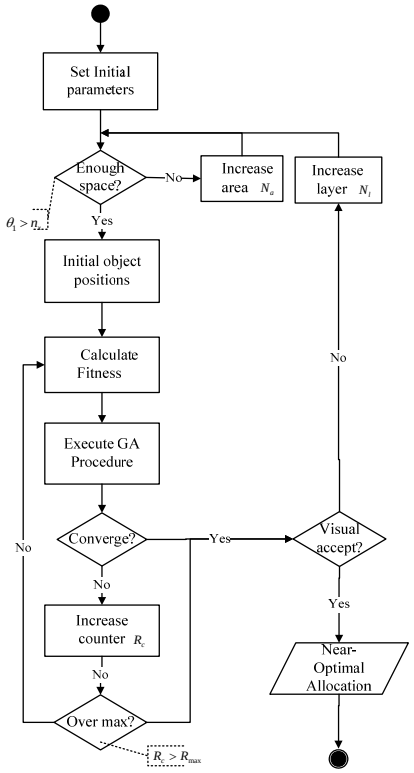


Figure 2. GA^2 **flow chart**

Genetic Allocation Algorithm

To reduce the visual complication of concrete's relations, next will describe how to apply Genetic Algorithm in this research. As shown in Figure 2, let each binary vector (chromosome) represents a concrete location (*x, y, z*). If we stepwise extend N_l and N_a, we will find an enough space to allocate all concretes such that give the default values of $\max(x) = \max(y) = N_a$ and $\max(z) = N_l$. As shown in Figure 2, the fitness Φ_p is defined as:

$$\Phi_p = \rho_p \left(\sum_{i=1}^{T_o} \rho_i \right)^{-1} \tag{3}$$

$$\rho_p = \sum_{i=1}^{R} \omega_i \cdot \theta_i \cdot \tag{4}$$

where:

ρ_P : objective value of *p*th population,

T_o : number of populations,

R : number of criteria.

Given max times $R_{\max}$ and visual acceptance ratio θ_3 / n_o with applying GA procedure, then we can find the near-optimal allocation H_x where *x* can be class, entity or object, etc.

III. CASE STUDY

To realize the 3D concretes, we adopt Java with JOGL to develop the visualization through JOGL (Java binding OpenGL) interface that supports Java with shift, zoom, rotation, and projection. In this research, model-view matrix defined by OpenGL transforms position matrix $[H_x]$ to both of model-view matrix and projection matrix, which can construct the final position of concretes in the 3D environment. To enhance the visual effect, table 1 shows the concrete mappings. Comparison with other researches, the proposed method GA^2 owns rather rich features for concrete visualization.

Table 1. Concrete mappings

Characteristics	[6]	[10]	[4]	This work
Concrete	Particles	Atom	Shape	Sphere
Concrete measure	Blob size	-	Size	Size
Concrete association	Particle clustering	Color	Area	Color cube
Concrete relationship	Cylindrical blob	Cons	Edge color	Cylindrical color
Relationship strength	Potential energy	-		Cylindrical diameter
Allocation Method	Hill-climbing	Spring-embedding		Genetic algorithm
Multi view	-	Supported		

Case 1: There is a class diagram with 20 classes and 30 relations representing a small software system. Table 2 indicates that when one of criterion is chosen as the minimization the objective function, only the corresponding item can be optimized while others cannot, i.e. $\min(\theta_1)$ minimizes the allowed space points to be 32, however, the total number of projection intersection is the biggest value in the results by different criteria.

Table 2. Results of minimization by criterion

Criteria	$\min(\theta_1)$	$\min(\theta_2)$	$\min(\theta_3)$	$\min(\theta_4)$
Space points	**32**	108	144	144
Segment Length	77.4	**78.5**	87.9	107.6
Space Intersection	10	9	**0**	0
Projecting Intersection	72	51	55	**26**

Case 2: As shown in Figure 3(b), because the relations between classes in different packages are featured by their cylindrical diameters and colors, the visual result can enhance the original distinguishability of Figure 3(a).

Case 3: This case presents a typical 3-teirs architecture from the web site "The 13th National Conference on Fuzzy Theory and Its Applications" held at Taiwan. Figure 4(b) shows the middle tier includes most of processing concretes, represented by spheres. For these concretes are tiny and have similar sizes. That means there exists some of improving space to reorganize and to reduce the classes in the middle tier for better system structure.

Case 4: For a typical application has tens of packages and hundreds of files, this case discusses a Java-based application Renew which is used to analyze discrete-event systems, and it has 16 packages and 125 classes as shown in Figure 5(a). In such kind of application, traditional jobs to modify some of classes and keep aware of their relations between other classes are difficult and complicated.

In this work, if we want to modify one of execution functions, for example, class *LateExecute* in package *simulator*. From the package view as shown in Figure 5(b), we can locate the *simulator* package at original and find there are five associated packages using zooming and rotating function of the tool. When we choose the package and zoom inside it, target class *LateExecute* now displays its corresponding classes that could be belonged to other packages as show in Figure 5 (c) and (d).

IV. CONCLUSION

To reduce complicated visual relations between concretes such as class, entity or objects, this work integrates the concrete mappings and GA^2 procedure, and proposes a visual tool. With the near-optimal allocations which are generated from GA^2 procedure, and arrangement criteria weightings which are specified by user, this tool improves the distinguishability of the concretes.

Acknowledgements

The authors would like to thank the National Science Council of the Republic of China, Taiwan, for financially supporting this research under Contact No. NSC-95-2221-E-327-042-MY3.

Reference

[1] S. G. Eick, J. L. Steffen, and E. E. Sumner, Jr. , "Seesoft- A Tool for Visualizing Line Oriented Software Statistics," *IEEE Trans. on Software Eng.*, vol. 18, no. 11, pp. 957-968, Nov. 1992.

[2] L. F. Marcus and J. I. Maletic, "3D Representations for Software Visualization," *ACM Symposium on Software Visualization*, San Diego, CA, pp. 27-36, 2003.

[3] K. L. Kroeker, "Seeing Data: New Methods for Understanding Information", *IEEE Computer Graphics and Applications*, vol. 24, pp. 6-12, 2004.

[4] C. Lewerntz, and F. Simon, "Metrics-based 3D Visualization of Large Object-Oriented Programs," *IEEE proc. of the First Inter. workshop on Visualizing Software for Understanding and Analysis*, pp. 70-77, 2002.

[5] M. Lanza and S. Ducasse, "Polymetric Views-A Lightweight Visual Approach to Reverse Engineering," *IEEE Trans. on Software Eng.*, vol. 29, no. 9, pp. 782-795, 2003.

[6] J. Rilling and S. P. Muder, "3D Visualization Techniques to Support Slicing-based Program Comprehension," *SDOS Computer & Graphics 29*, pp. 311-329, 2005.

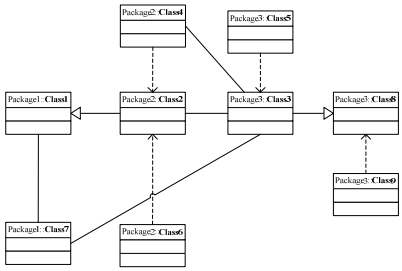
[7] J. M. Bieman and B.-K. Kang, "Measuring Design-Level Cohesion," *IEEE Trans. on Software Eng.*, vol. 23, no. 2, pp. 111-124, 1998.

[8] L. C. Briand and J. Wüst, "Modeling Development Effort in Object-Oriented Systems Using Design Properties," *IEEE Trans. on Software Eng.*, vol. 27, no. 11, pp. 963-986, 2001.

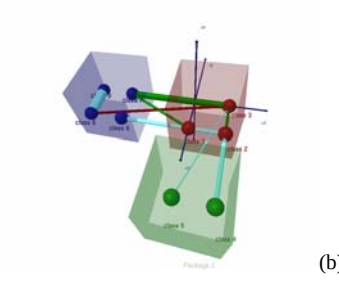
[9] H.-C. Yang and F.-T. Cheng, "Architecture Adaptability Evaluation for a Manufacturing Execution System," in *Proc. of the 8th International Conference on Automation Technology*, pp. 67-72, May, 2005.

[10] B. A. Malloy and J. F. Power, "Using a Molecular Metaphor to Facilitate Comprehension of 3D Object Diagrams", *Proceedings of the 2005 IEEE symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)* Dallas, Texas, USA, pp. 233-240, 21-24 September 2005.

[11] T. Panas, R. Lincke, and W. Lowe, "Online-Configuration of Software Visualizations with Viss3D," *Association of Computing & Machinery*, pp. 173-182, 2005.

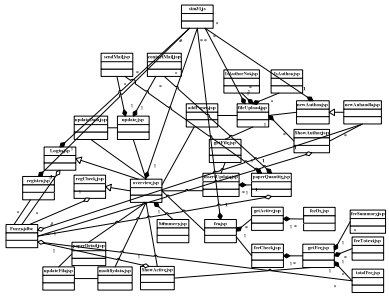


(a)

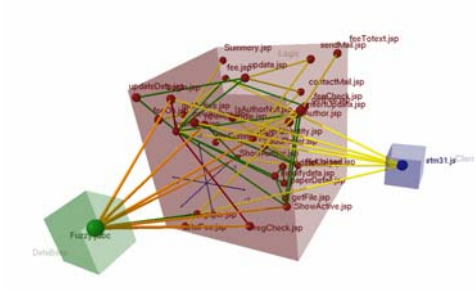


(b)

Figure 3. 3 packages with 9 classes

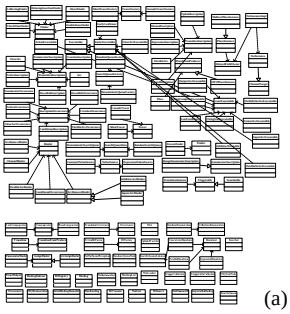


(a)

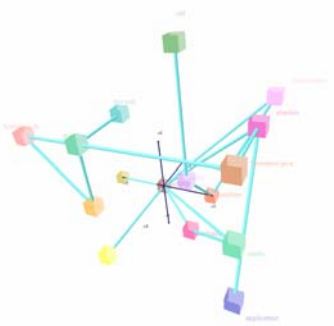


(b)

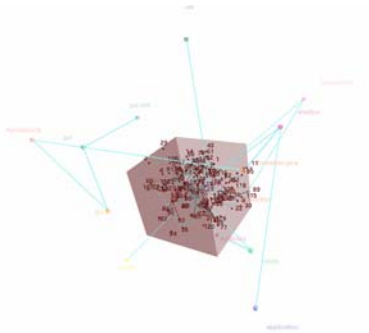
Figure 4. 33 classes in 3 tiers



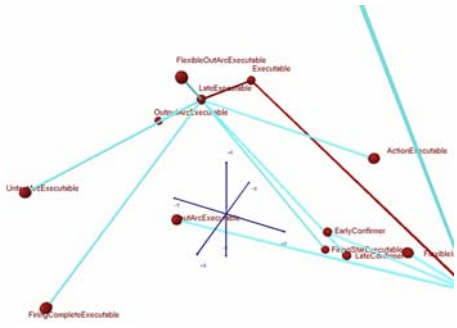
(a)



(c)



(c)



(d)

Figure 5. 16 packages with 125 classes in 3 tiers