Proceedings of the 5th WSEAS Int. Conf. on SIMULATION, MODELING AND OPTIMIZATION, Corfu, Greece, August 17-19, 2005 (pp166-171)

A Software Reuse Approach for Developing Grab-and-Glue Models

MAN WAI LEE, SHERRY Y. CHEN
School of Information Systems, Computing and Mathematics
Brunel University
Uxbridge, Middlesex, UB8 3PH
U.K.

Abstract: - A Grab-and-Glue framework (Grab-and-Glue) has been proposed as a potential solution to deal with the problem of time-consuming and high cost during the development of a simulation model caused by classical simulation modelling frameworks. However, the way to develop a simulation model following Grab-and-Glue is still an unknown factor. Since the concept of Grab-and-Glue model development is similar to the technique of object reuse in Software Engineering, it might be a chance that such a technique could benefit to the development of a Grab-and-Glue model. In this paper, we are going to investigate this issue by studying the characteristics of Grab-and-Glue and the possibility of developing models on the basis of the object reuse techniques.

Key-Words: Object reuse, Grab-and-Glue, Simulation Modelling, World Wide Web, Open Sources, Search Engines

1 Introduction

Simulation Modelling is able to deal with "what-if" problems in various areas, such as an industrial or a business domain [19]. One possible application of simulation modelling is to promote discussions for seeking potential solutions and understanding a real world within a short period of time [21]. However, such an application is difficult to be executed because a time-consuming problem exists during the process of simulation model development based on classical simulation modelling frameworks [19]. This problem consequently creates a barrier to enable simulation modelling to be widely used for dealing with business problems. Melão and Pidd [16] agree with such an argument by providing a survey result, in which nearly 6% of the respondents think that applying simulation modelling for dealing with their problems is time-consuming. Hence, it is necessary to find out a possible solution to promote the use of simulation modelling.

Paul [18] proposes a Grab-and-Glue framework (in short, Grab-and-Glue) to deal with the aforementioned problem. The basic concept of such a framework is to grab scissions from the Web and glue them together to form a simulation model, in which objects and components are included in the definition of a scission [3]. Since the modelling method suggested by the Grab-and-Glue framework is similar to the development of a model based on the concept of object reuse in Software Engineering, it is worth investigate whether the reuse technique can make a contribution to find the way of developing Grab-and-Glue models. Applying reuse for developing models has many benefits, including high flexibility, extensibility and reusability [2]; in addition to achieve time-saving and cost-saving [22]. Examples can be found in [13][24]. Moreover, Robinson et al. [22] agree that the "reuse of basic components" is helpful for developing simulation models. However, the feasibility of applying such a technique in Grab-and-Glue is still an unknown issue, because the modelling method of Grab-and-Glue is different from that of classical simulation modelling framework [4]. Hence, this paper aims at investigating the feasibility of using the reuse technique for developing Grab-and-Glue models. The structure of the paper is as follows: Section 2 presents a theoretical background to explain the relationships between Grab-and-Glue and reuse. Section 3 examines if reuse is beneficial to the Grab-and-Glue framework based on the theoretical background in Section 2, which, in turn, formulates the hypothesis of this study. Based on this hypothesis, Section 4 is to illustrate an empirical study of developing a Grab-and-Glue model based on the reuse techniques. It then moves to have a discussion about the suitability of applying reuse in Section 5. Finally, we make a conclusion and recommend some directions for future research.

2 Theoretical Background

2.1 Grab-and-Glue framework

Fig.1 illustrates a Grab-and-Glue framework. A simulation model will be developed on the basis of the Grab-and-Glue framework if problem owners decide to build that model for understanding his problem, or promoting a discussion in a group meeting. Due to the fact that a Grab-and-Glue based simulation model can be developed based on integrating scissions such as objects or components together, scissions will be searched and grabbed from different sources, including the Web [3]. Subsequently, the grabbed scissions will be assembled together to form a simulation model. The created model will consequently be run and the output will be analysed by the problem owners. This step is essential because it helps to determine whether the created model is able to help the problem owner to understand his/her problem or not. If the problem owners think that the model is able to help them, life moves on. Alternatively, the developed model needs to be modified by rejecting irrelevant scissions and gluing new grabbed scissions to the relevant positions. The aforementioned "Grab", "Glue" and "Reject" processes are iterated until the problem owners satisfies with the result [18].

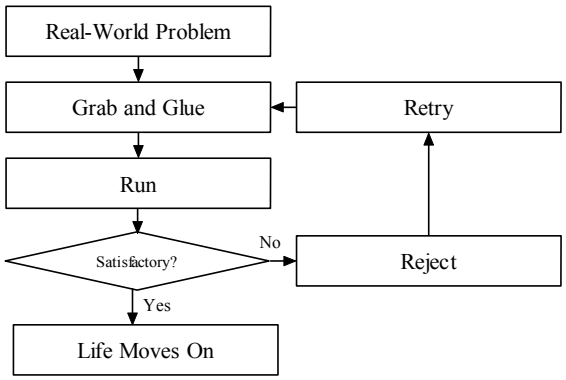

Fig.1: Grab-and-Glue Framework (adapted by [18])

2.2 Reuse

"Reuse" can be defined as "to maximize the probability that, for any given project, the benefits to that project will outweigh the costs as perceived by the project team." [5], or "the process of building software systems from components (or artifacts) designed for reuse" [14]. Although a model created on the basis of the reuse technique has the potentials of time-saving and cost-saving, which fits the requirements of Grab-and-Glue, the feasibility of applying it for developing Grab-and-Glue models is still unsure. Such an approach can only be taken if the reuse is appropriately used. Hence, an investigation of the impact of the inappropriate use is necessary.

Many researchers argue that applying reuse inappropriately will produce a number of problems. Kasputis and Ng [11] argue that the cost of model development cannot be reduced if existing components are not designed for working together. Sherif and Vinze [23] agree that components can only be reused if the idea of reuse is within designers' mind during the process of component development. Frakes and Succi [6] agree that finding an opportunity to reuse objects from a software package is difficult because the reusable idea has not been considered during the design stage; in addition, the identification of reuse opportunities in a large and complex system is time-consuming. Moreover, Robinson et al. [22] suggest that reuse can only be applied to the reuse of basic modelling components. Neither the reuse of subsystem models nor the reuse of a similar model is appropriate. It is due to the fact that the former may be much expensive than creating models from scratch unless the subsystem components within the model are simple and a modeller can understand how they work without spending lots of time. Conversely, the latter one might be much costly because ensuring the confidence of a model without detailed verification and validation is difficult.

To sum up, it is difficult to achieve time-saving and cost-saving if "reuse" is applied inappropriately. Since the purpose of applying "reuse" is similar to that of applying the Grab-and-Glue modelling method, it can be assumed that reuse can contribute to the development of Grab-and-Glue models. However, it is not clear what the effects will be demonstrated when reuse has been applied inappropriately, which will be discussed in the next section.

3 A Theoretical Study

Literature identifies that two problems exist if reuse has been applied inappropriately, which are the accuracy of the created model and a compatibility problem. In order to apply the reuse technique for developing a Grab-and-Glue model, it is necessary to investigate whether these two problems will affect the development of a Grab-and-Glue model or not.

The first issue needs to be investigated is the accuracy of the created model. The consideration of such an issue is important because the justification of the usability of a created model is based on its accuracy 0. In order to build a model by reusing existing components, such components must be

highly accurate. Alternatively, the accuracy of the created model will become an unknown issue, hence, the usability of the created model will be low. On the other hand, Grab-and-Glue aims at developing a quick and dirty simulation model to try to understand the complex real world, in which model accuracy is not an essential criterion [25]. Grab-and-Glue intends to shift the modelling process from "is the model correct" to "is the analysis, albeit with unproven software, acceptable given the large experimentation that swift modelling has enabled us to carry out in a sort space of time" [22]. If a model is built on such a purpose, it is difficult to ensure the accuracy of the created model because the accuracy of the components used for developing that model is unsure. Also, focusing on model accuracy is not the purpose of Grab-and-Glue, instead, such a model should be developed on the basis of classical simulation modelling frameworks [4].

The second issue needs to be considered is the compatibility of the grabbed scissions. Two possible ways to achieve such requirements, which are to keep reuse in the mind during components development [23], and only reuse components created by the same composer/company. Regardless of the former way or the latter way, it is irrelevant to the development of Grab-and-Glue models due to the fact that Grab-and-Glue intends to grab scissions created by any composers [18]. Hence, one may argue that it is a difficulty to ensure the grabbed scissions are compatible with each other. However, such a consideration is not essential because the proposition of developing Grab-and-Glue models is to develop disposal models quickly for understanding the real world. Instead of considering the compatibility of the grabbed scissions, the Grab-and-Glue framework suggests that irrelevant scissions should be rejected (see Fig.1). As a result, such an argument cannot be established.

Based on the discussions provided in this section, it can be concluded that the problem of model accuracy and the incompatibility problem will not affect the development of a Grab-and-Glue based model (see Table 1). The next section demonstrates the development of a Grab-and-Glue based simulation model, a single-server queuing mode, on the basis of the reuse technique by applying an empirical study.

4 An Empirical Study

Based on the discussions in the previous section, it can be assumed that the reuse technique can contribute to the development of a Grab-and-Glue model on the basis of the Grab-and-Glue framework. Such an assumption will be justified by the process of developing a widely cited example, the single-server queuing model

4.1 Single-server queuing model framework

Fig.2 illustrates a conceptual framework of a single server queuing model. In this framework, "Entity Arrival" is an activity for any entity's arrival. The time an entity arrives is recorded, and it subsequently joins to the end of a queue within an activity "Queue". An activity "Process" is the course of action from the entity's arrival to the entity's being served successfully. If the activity "Process" is idle, the first entity within the queue will be removed and served; alternatively, it will wait until the activity "Process" is free. After a process is finished, the served entity leaves from the activity "Entity Exit".

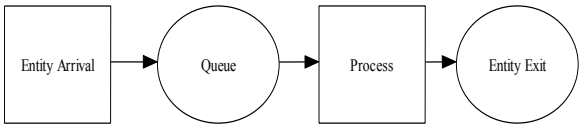

Fig.2: Conceptual Framework for Single-Server Queuing Model

4.2 Expected scission for Grab-and-Glue model development

Table 2 illustrates the expected scissions for developing the single-server queuing model. A scission, *random number generator*, is required for generating entities' arrival. Subsequently, the

Table 1: Effects of Grab-and-Glue if reuse has been applied inappropriately

Problem of reuse	Affect Grab-and-Glue?	Why?
The issue of model accuracy	No	Grab-and-Glue aims at developing a quick and dirty model for understanding the complex real-world. Hence, model accuracy is not an important issue. As a result, the accuracy of the grabbed components needs not to be considered
Compatible problem	No	Grab-and-Glue suggests that scissions that are not compatible with others should be rejected.

Table 2: Expected scissions for developing the single-server queuing model

Activity	Expected Scissions	Reasons
Entity Arrival	Random number generator	Generate entities' arrival (random number)
	Simulation Clock	Scheduling and monitoring the execution of the model
	Scission for holding temporary data	Store entities' states and their corresponding arrival time, then pass these information to Queue
Queue	First-in-first out queue	Entity are served on the basis of a first-in-first-out discipline
	Simulation Clock	Scheduling and monitoring the execution of the model
Process	Random number generator	Generate the time required for executing the "Process" step
	Scission for holding temporary data	Store entities' states and their corresponding process time
	Simulation Clock	Scheduling and monitoring the execution of the model
Entity Exit	Discrete event calculator	Calculate the output of the created model
	Simulation Clock	Scheduling and monitoring the execution of the model

generated data are stored by a scission, *scission for storing temporary data*. The stored data is then passed to the activity Queue by a scission *first-in-first-out* queue. If the activity "Process" is idle, the first entity within the *first-in-first-out* queue is served. The time for accomplishing the process is generated by a *random number generator*. Finally, the output of the model is created by a scission *discrete event calculator*. Apart from the aforementioned scission, a *simulation clock* is required for scheduling and monitoring purposes.

4.3 Search Spaces

The previous subsection identifies that expected scissions for assembling a single-server queuing model are a *random number generator*, a *scission for holding temporary data*, a *queue*, a *simulation clock*, and a *discrete event calculator*. Three approaches have been applied for searching the required scissions. The first approach is to search scissions from Google, which is a widely used search engine. If the result of this round fails to provide suitable scissions, the search will be modified to include some open source websites such as http://www.planet-source-code.com [9]. If the scissions cannot be obtained from both Google and the open source websites, the subsequent search space is extended to include other published programming codes.

4.4 The Grab-and-Glue Process

A summary of a Grab-and-Glue process is shown in Table 3. In the initial round of searching, a *random number generator* by Horstmann [7] by McNaughton [15], a *discrete event calculator* by Huffman [8], and a *scission for holding temporary data* by Huffman [8]

Table 3: Summary of the Grab-and-Glue Process

Trials	Scissions have been grabbed	Action taken
First Trial	Random Number Generator [7]	Accept
	Queue [10]	Reject
	Simulation clock [16]	Reject
	Discrete event calculator [8]	Accept
	Scission for holding temporary data [8]	Accept
Second Trial	Queue [12]	Reject
	Simulation clock [21]	Reject
Third Trial	Queue [7]	Accept
	Simulation clock [8]	Accept

are grabbed from various sources. Due to the fact that all the grabbed scissions were created by Java, such a programming language was adopted for gluing purpose. After a trial of gluing them together, it was discovered that a large number of syntax errors occurred. An investigation of the created model was then conducted, and it was discovered that the *queue* and the *simulation clock* are irrelevant. Hence, both of them are rejected. The subsequently round of searching was then focused on finding the queue and the simulation clock. In this round of searching, a *queue* created by Lemay and Cadenhead [12] and a *simulation clock* created by Raxix [20] were grabbed. However, both of them were rejected after trying to glue them into relevant positions because they are irrelevant. Hence, they are rejected and the third round of searching was conducted. In this round, a *queue* created by Horstmann [7] and a *simulation clock* created by Huffman [8] were grabbed. After integrating them together, a small number of syntax errors were found by the Java complier. Since this number is very small, the scissions grabbed in the

third round were accepted and such errors are corrected by manual.

5 Results and Discussions

The aforementioned empirical study demonstrated that the reuse technique could contribute to the development of a single-server queuing model by reusing scissions grabbed from the Web. However, a new problem, limited number of suitable scissions, exists in addition to the compatibility [23] and the accuracy [25] problems suggested by existing literature. In terms of limited number of suitable scissions, it is due to the fact that there is a lack of published models and physical objects on the World Wide Web for allowing such distribution [17]. A possible way to tackle this problem is to grab scissions with multiple programming languages and then glue them together. With regards to compatibility, this experiment illustrates that many scissions found from the Web are incompatible with each others. Even though all the grabbed scissions are written in Java, it took a long time to glue them together because of the variety of returned values from different java classes and methods. For example, scission A has 2 outputs, but scission B has 1 input (see Fig.3). Such a finding matches with that of [23]: the compatibility problem may exist if reuse has not been kept in mind during the development of the reuse components. In order to tackle such a problem, two possible solutions can be used. The initial method is to investigate the suitability of developing "wrapper" to deal with the compatibility problem. For example, the problem in Fig.3 can be tackled by applying a wrapper to wrap Scission B so that the input of such a scission is converted from 1 to 2 (see Fig.3). Another possible way is to modify the source code of the problematic scission if the structure of that is not complex [3]. Alternatively, those scissions that are incompatible with others are rejected according to the Grab-and-Glue framework. In respects of accuracy, which is not considered as an important issue, it is due to the fact that Grab-and-Glue aims at creating a model for understanding the real world, instead of building an elegant calculating machine [25]. Based on such a modelling purpose, the accuracy of the grabbed scissions has not been justified. Instead, those scissions that cannot function as expected are rejected according to the Grab-and-Glue framework.

In addition, two critiques exist in this study. The first critique is that the flexibility of the created model is not as high as expected. The Grab-and-Glue framework suggests that irrelevant scissions

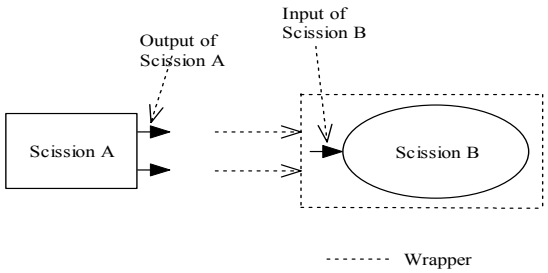

Fig.3: Gluing Scissions A and Scissions B using Wrapper

can be changed easily without affecting the structure of the whole model [4]. However, the developed server queuing model suffered from such a problem. Whenever a scission is changed, the execution of the whole model is also affected. One reason of having such a problem is the inappropriate modelling method. In other words, the problem of lacking flexibility is not come from "reuse"; instead, it is caused by the modelling method. Hence, such a problem can only be tackled by investigating better ways of developing simulation models. Another critique is that the time required for developing the single-server queuing model. Existing works suggest that reuse can reduce the time required for model development [22], which is also the goal of using Grab-and-Glue [18]. In this study, such an issue has not been fully explored because it is required to develop the single-server queuing model with the traditional simulation modelling frameworks. However, under a roughly estimation, time that has been spent for developing models and analysing the output result is relatively short if it is compared with the time required for developing the same model from scratch using code. Hence, it could be saying that Grab-and-Glue has high potentials to reduce time required for developing simulation models.

6 Concluding Remarks

This paper justifies that the technique of object reuse is capable to contribute to the development of a Grab-and-Glue based simulation model, but it fails to justify whether Grab-and-Glue is capable to save time required for developing simulation models. However, there are three problems: the limited number of suitable scissions, compatibility and accuracy problems. A number of solutions have been proposed to address these problems. Future research is needed to investigate the feasibility and effectiveness of using these solutions.

References:

[1] O. Balci, Validation, Verification, and Testing Techniques Throughout the Life Cycle of a Simulation, *Annals of Operations Research*, Vol.53, 1994, pp. 121-173.

[2] G. Chen, and B. K. Szymanski, Component-Oriented Simulation Architecture: Toward Interoperability and Interchangeability, In *Proceedings of the 2001 Winter Simulation Conference*, 2001, pp. 495-501.

[3] T. Eldabi, M. W. Lee, and R. J. Paul, The Feasibility of Constructing Simulation Models Using The Web-Based 'Grab-and-Glue' Framework, In *Proceedings of the 2004 Winter Simulation Conference*, 2004, pp. 1494-1501.

[4] T. Eldabi, M. W. Lee, and R. J. Paul, A Framework for Business Process Simulation: The Grab and Glue Framework, In *Proceedings 15th European Simulation Symposium: Simulation in Industry*, 2003, pp. 291-296.

[5] R. G. Fichman, and C. F. Kemerer, Incentive Compatibility and Systematic Software Reuse. *Journal of Systems and Software*, Vol.57, Issue1, pp. 45-60

[6] W. B. Frakes, and G. Succi, An Industrial Study of Reuse, Quality, and Productivity, *Journal of Systems and Software*, Vol.57, Issue2, 2001, pp. 99-106

[7] C. Horstmann, *Big Java*, John Wiley & Sons Inc, 2002.

[8] B. J. Huffman, An Object-Oriented Version of SIMLIB (a Simple Simulation Package), *Informs Transactions on Education*, Vol.2, No.1, 2001.

[9] I. Ippolito, Planet Source Code™, Exhedra Solution, Inc., 2004, Available online via <http://www.planet-source-code.com>.

[10] F. Jones, Queue, Exhedra Solution, Inc., 2003, Available online via <http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=3666&lngWId=2>.

[11] S. Kasputis, and H. C. Ng, Composable Simulations. In *Proceedings of the 2000 Winter Simulation Conference*, 2000, pp. 1577-1584.

[12] L. Lemay, and R. Cadenhead, *Sams Teach Yourself Java 2 Platform in 21 Days*, Techmedia, 1999.

[13] G. T. Mackulak, F. P. Lawrence, and T. Colvin, Effective Simulation Model Reuse: A Case Study for AMHS Modeling, In *Proceedings of the 1998 Winter Simulation Conference*, 1998, pp. 978-984.

[14] C. McClure, Value-added Year 2000: Harvesting Components for Reuse, Extended Intelligence, Inc., 1997, Available online via <http://www.reusability.com/papers1.html>.

[15] M. McNaughton, A Clock... No big deal, but for a clock, it's cool, Exhedra Solution Inc., 2001, Available online via <http://www.planet-sourcecode.com/vb/scripts/ShowCode.asp?txtCodeId=2083&lngWId=2>

[16] N. Melão, and M. Pidd, Use of Business Process Simulation: A survey of Practitioners, *Journal of the Operational Research Society*, Vol.54, No.1, 2003, pp. 2–10.

[17] E. H. Page, A. Buss, P. A. Fishwick, K. J. Healy, R. E. Nance, and R. J. Paul, Web-Based Simulation: Revolution or Evolution? *ACM Transactions on Modelling and Computer Simulation*, Vol. 10, No. 1, 2000, pp.3–17.

[18] R. J. Paul, "Chapter XIII, The Internet: An End to Classical Decision Modelling?", in Haynes, J. D. (Ed.), *Internet Management Issues: A Global Perspective.* Idea Group Publishing and Information Science Publishing, 2002.

[19] M. Pidd, *Computer Simulation in Management Science*, John Wiley & Sons, 1998.

[20] Raxix, A Sample TCP, Exhedra Solution. Inc., 2002, Available online via <http://www.planet-sourcecode.com/vb/scripts/ShowCode.asp?txtCodeId=2630&lngWId=2>

[21] S. Robinson, Modes of Simulation Practice: Approaches to Business and Military Simulation, *Simulation Modelling Practice and Theory*, Vol. 10, Issue 8, 2002, pp. 513-523.

[22] S. Robinson, R. E. Nance, R. J. Paul, M. Pidd, and S. J. E. Taylor, Simulation Model Reuse: Definitions, Benefits and Obstacles, *Simulation Modelling Practice and Theory*, Vol.12, Issues7-8, 2004, pp. 479–494.

[23] K. Sherif, and A. Vinze, Domain Engineering for Developing Software Repositories: a Case Study, *Decision Support Systems*, Vol.33, Issue1, 2002, pp. 55–69.

[24] B. Spitznagel, and D. Garlan, A Compositional Formalization of Connector Wrappers, In *Proceedings of the 25th International Conference on Software Engineering*, 2003, pp. 374–384.

[25] S. J. E. Taylor, P. Lendermann, R. J. Paul, S. W. Reichenthal, S. Straßburger, and S. J. Turner, Panel on Future Challenges in Modeling Methodology. In *Proceedings of the 2004 Winter Simulation Conference*, 2004, pp. 327-335.