Proceedings of the 2004 Winter Simulation Conference
R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.

EXAMINING THE FEASIBILITY OF CONSTRUCTING SIMULATION MODELS USING THE WEB-BASED 'GRAB-AND-GLUE' FRAMEWORK

Tillal Eldabi
Man Wai Lee
Ray J. Paul

Centre for Applied Simulation Modelling (CASM)
Department of Information Systems and Computing
Brunel University
Uxbridge
Middlesex UB8 3PH, U.K.

ABSTRACT

The Grab and Glue discrete event simulation (Grab-and-Glue) framework has been proposed for several years. This paper investigates the feasibility of this framework. A simulation model of a single server queuing system is constructed by following the Grab-and-Glue framework as a ubiquitous example. Different scissions are grabbed from the Web or programming textbooks, and glued together using JBuilder5. The potential of Grab-and-Glue is discussed as well as the current challenges of constructing Grab-and-Glue simulation models. Some critiques and recommendations for the ongoing research on constructing simulation models based on the Grab-and-Glue framework have also been proposed.

1 INTRODUCTION

Constructing simulation models following traditional simulation modelling frameworks take a long time and hence high cost (Pidd 1998, Eldabi et al. 2003). However, problems occur rapidly in the real life, especially in the business domain, therefore, the problem may become out of date by the time the model is completed. In traditional simulation modelling frameworks, data collection, model construction and output analysis are leading to time consuming problem (Eldabi et al. 2003). Although Paul (2002) proposed the Grab-and-Glue discrete event simulation modelling framework (Grab-and-Glue) to try to alleviate this particular problem, the feasibility of constructing simulation models by following this framework has not yet been established. It is not yet known whether Grab-and-Glue could save time in developing a reliable model or not. This paper assesses whether it is possible to construct a Grab-and-Glue simulation model or not and the acceptability of its output.

A simulation model can be constructed by either using programming source code or simulation packages. Current simulation packages do not allow features that incorporate external programs freely (Spitznagel and Garlan 2003), which is the main principle of Grab-and-Glue. This research will first concentrate on the use of a general purpose language, Java, to assess the feasibilities of the Grab-and-Glue framework. In this paper, a ubiquitous example single-server queuing system, which is citied by Law and Kelton (2000), has been adopted for investigation purposes.

A brief description of the Grab-and-Glue framework is given in Section 2. Section 3 discusses the model construction base on the Grab-and-Glue framework. In Section 4, the single-server queuing system is described by using a "barbershop" example. A description of the assembly of the model following the Grab-and-Glue framework is also presented in this section. Section 5 presents the findings of the study. Critiques and lessons learned are presented in Section 6. Finally, Section 7 presents the conclusions and future works.

2 BACKGROUND OF THE STUDY

The intention of this article is to assess the feasibility of constructing simulation models by following the Grab-and-Glue framework suggested by Paul (2002). The concept of this framework is based on grabbing different scissions from the Web and gluing them together to form a model (see Figure 1). "Scissions" are defined as "the act of cutting or severing into divisions or fissions". In computer science, we intend to define this term as "a piece of something for model construction". For example, it may be a piece of source code in any programming languages, a component, an object, a file, or a document etc. Although the definition of a "scission" includes "component" and "object", we intend to adopt this term because either "component" or "object" has its predefined meanings. "Components" and "Objects" can be "Scissions", but not vice versa.

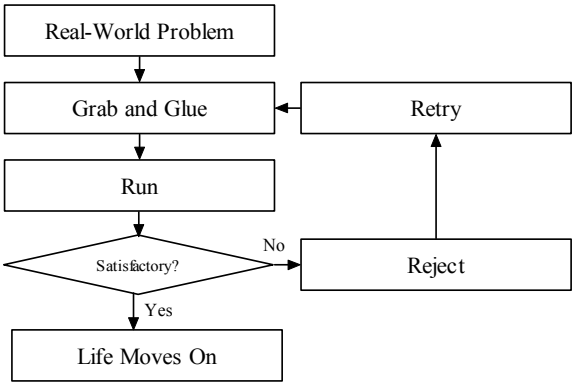

Figure 1: Framework for Grab-and-Glue

After gluing the grabbed scissions, if the problem owners feel the model is satisfactory, life moves on and the problem owners can continue their work; otherwise the undesired scissions inside the model will be rejected, the grabbing process will be repeated and the new scissions will be glued to the relevant positions.

Grab-and-Glue depends on assembling the grabbed scissions. This idea is new to simulation area but it is not new to software engineering. Mackulak et al. (1998) stated that reuse of existing generic models such as simulators or software packages that contain pre-programmed models, can reduce the model building time and increase the simulation's accuracy. Du and Wu (2001) assembled a model to develop an evolutionary system development process by assembling different components; Spitznagel and Garlan (2003) proposed to assemble a system from independent components and to use a wrapper for component communications; whilst Ravichandran and Rothenberger (2003) proposed software reuse for increasing the software development efficiency and the quality of the software system.

3 MODEL CONSTRUCTION

As mentioned in Section 2, the idea of the Grab-and-Glue framework is to grab different scissions and to glue them together. This section will describe the process of model development. The next subsection will discuss a framework for the single-server queuing model. After that the scissions required to construct the simulation model, and the search space for the scissions will be identified.

3.1 Framework for Single-Server Queuing Model

Figure 2 shows the framework of the single-server queuing model. Arrival time will be recorded for each newly arriving entity. The arriving entity will then join the end of the queue. The first entity in the queue will be removed and served if the activity "process" is idle. Otherwise, it will wait until the activity "process" is free. When the current entity inside the "process" has been served, it will leave the model, and the next entity in the queue will be served.

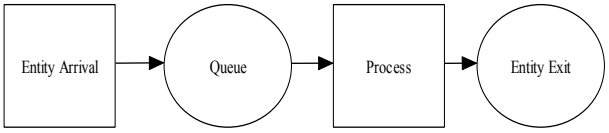

Figure 2: Framework for Single-Server Queuing Model

3.2 Scissions for Grab-and-Glue

As shown in Figure 3, the basic scissions for constructing the single-server queuing model are *random number generator*, *simulation clock*, *queue*, *scission for calculating the discrete value*, and *scissions for holding temporary data* (Huffman 2001). A random number generator is required for number generation, which will be used as an input of the simulation model (the arrival time and the departure time). A simulation clock will be used to record and manage the simulation time. A queue will be used to store the list of entities waiting to be served, base on the first-in-first-out approach. A scission for producing the simulation output, and a scission for holding temporary data, such as the state of an action and the corresponding time are also required.

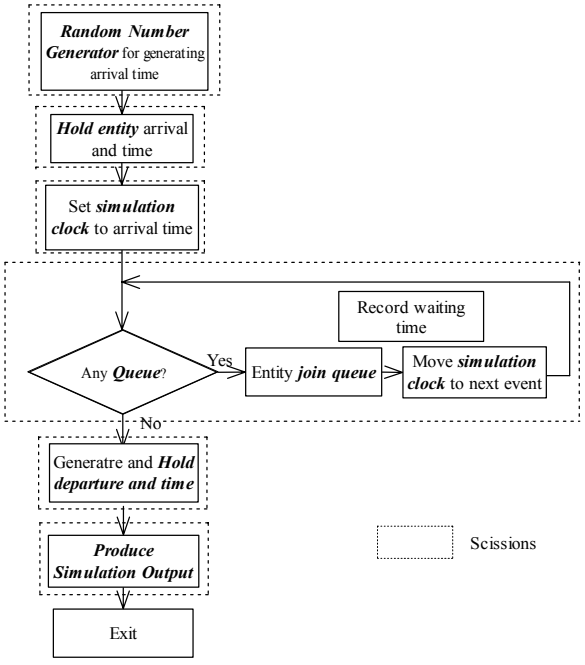

Figure 3: Scissions for Single-Server Queuing Model

3.3 Search Space

After deciding the required scissions, it is important to establish the search space (in the web) for those scissions mentioned in Section 3.2. Several methods have been adopted for searching the required scissions. The first approach to search is to use search engines Google (2004) and Yahoo (2004) to find the required scissions. If the result of this round fails to provide suitable scissions, the

search will be modified to include open source websites such as `http://www.planet-source-code.com` (Ippolito 2004). Initial searches suggest that it is difficult to find all the required scissions to build the model, so subsequently the search space is extended to include programming textbooks.

4 EXAMPLE OF GRAB-AND-GLUE

This section provides a demonstration of Grab-and-Glue, by using a single-server queuing system as a ubiquitous example of simulation as used by Law and Kelton (2000). The single-server queuing system is demonstrated through a "barbershop" example. This example can be described as follows: at the beginning, there will be one barber and no customer in the barbershop. When a customer arrives, his hair will be cut by the barber. When a second customer arrives, his hair will be cut if the barber is free; otherwise he will wait until the barber finishes serving the current customer. In the meantime, if a third or more customers arrive, they will join the waiting customer to form a queue. Customers in the queue will be served, depending on whether the barber is free or not and on their positions in the queue. The queue follows the first-in-first-out approach. This process will be iterated until eleven customers have their hair cut by the barber in this example. For destination purposes, this example will follow eleven customers. Figure 4 shows the flowchart of the simulation program which is written in Fortran and C by Law and Kelton (2000); and Java by Huffman (2001). The source code by Law and Kelton (2000) is written from scratch, while the source code by Huffman (2001) is written by converting it from Law and Kelton (2000).

The next subsection describes the implementation of this model, following the framework in Figure 4. The "Grab" process will be presented first, followed by the "Glue" process.

4.1 The "Grab" Process

A summary of the scissions which were grabbed is shown in Table 1.

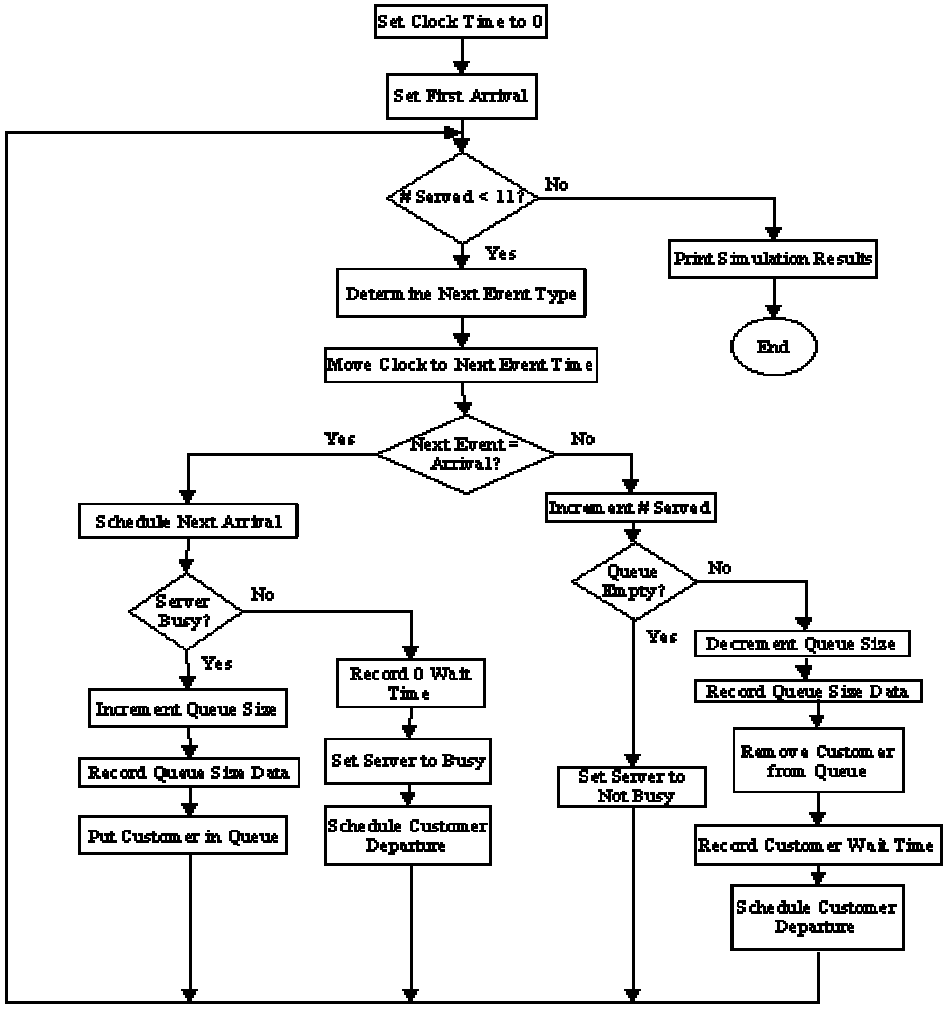

Figure 4: Flowchart of Single-Server Queuing System Program (by Huffman 2001)

Table 1: Summary of the Grabbed Scissions

Scissions	Authors
Random Number Generator	Horstmann (2002)
Queue (First-in-First-out)	Lemay and Cadenhead (1999)
	Horstmann (2002)
Simulation Clock	McNaughton (2001)
	Raxix (2002)
	Huffman (2001)
Discrete Event Calculator	Huffman (2001)
Scissions for holding data	Huffman (2001)

- Random Number Generator. A "random number generator" scission was required for constructing simulation models. The number generator needed to be a class file or a function which can generate random numbers (non-integer) as an input to the simulation model. In this experiment, searching for a random number generator was started by using search engines Google (2004) and Yahoo (2004) by using keywords such as "random data generator", "random number generator", and "pseudo number generator". Although a number of results were obtained, they were not suitable to be put inside the model for number generation. As a result, the search space was modified to include programming textbooks. Finally, a number generator class was found in Big Java, which was authored by Horstmann (2002). This java class file was written for generating integers. However, what the model required is a float number generator. Therefore, this file was modified to satisfy the requirement.
- Queue. A "queue" scission was required to serve the customers based on a first-in-first-out basis. "Queue" and "first in first out" were used as keywords in the search engine. Because the output of the search was not acceptable, the search direction was modified to search from the open-source website and programming textbooks. Several java class files were grabbed. The first one was written by Jones (2003), which was obtained from <http://www.planet-source-code.com> (Ippolito 2004). The second one was written by Lemay and Cadenhead (1999). The name of the files (methods) were addelement(), insertelementat(), removeelement(), and linkedlistenumerator.java (java class file). Another group of a linked list for queue was found from Horstmann (2002). This group consisted of two classes, namely LinkedList.java and ListIterator.java. LinkedList.java contained all the methods for queuing, whilst ListIterator.java contained the interface for LinkedList.java. Among all these groups, the one from Horstmann (2002) was adopted. However, LinkedList.java was modified so that it could be glued during the modelling process. The main reason was that this class file only contained the addFirst method, without a method for joining at the end of the queue.
- Simulation Clock. A "simulation clock" scission was required for setting and recording the simulation time. Huffman (2001) constructed a Timer java class file to act as a simulation clock. According to his description, this scission could not be accessed directly by any method outside the Timer class so that it could not be changed unintentionally. Other scissions were also grabbed for simulation purposes. Keywords "clock", "simulation clock", "timer", and "simulation timer" were used in the search engine and the open source website. The first result was written by McNaughton (2001). However, this was rejected because it performed like a clock and its function did not satisfy the simulation's purpose. The second one "a sample TCP", which was constructed by Raxix (2002), was found in <http://www.planet-source-code.com> (Ippolito 2004). After investigating this file, it was concluded that the grabbed scission was difficult to glue into our program. As a result, both of them were rejected. Another scission for timer was grabbed from Huffman (2001). This scission was finally selected because it had already been used for simulation purpose, and it was discovered that it was suitable to be adopted for our model.
- Output Generation. A scission for generating the output result was required. "Discrete calculation" was used as a keyword for searching through a search engine. However, since it was not possible to find either through the search engine or the open source website, this scission was grabbed from Huffman (2001).
- Scissions for Holding Temporary Data (Activity). A scission for holding data (the state of an action and the corresponding time) was also required. For example, state could be the arrival, departure, wait etc., whilst time was generated by the random number generator. Based on this criterion, it was discovered that the most suitable scission was written by Huffman (2001) after searching different sources. As a result, it was grabbed.

4.2 The "Glue" Process

The first trial of the "glue" process was conducted using the scissions in Table 2 (see First Trial) and JBuilder5 – a Java development environment (Borland® 2004). After combining the basic scissions, the model generated a large number of syntax and logical errors. One of the reasons is that because some of the scissions were incompatible with each other.

Table 2: Summary of the Grab-and-Glue Process

Trials	Scissions have been grabbed	Action taken
First Trial	Random Number Generator (Horstmann, 2002)	Accept
	Queue (Jones, 2003)	Reject
	Simulation clock (McNaughton, 2001)	Reject
	Discrete event calculator (Huffman, 2001)	Accept
	Scission for holding state of action and time (Huffman, 2002)	Accept
Second Trial	Queue (Lemay and Cadenhead, 1999)	Reject
	Simulation clock (Raxix, 2002)	Reject
Third Trial	Queue (Horstmann, 2002)	Accept
	Simulation clock (Huffman, 2001)	Accept

As described in Figure 1, if the problem owner is not satisfied, irrelevant scissions will be rejected and the Grab-and-Glue process will be retried. "Queue" from Jones (2003), and "timer clock" from McNaughton (2001) were rejected after the first trial. These two scissions were rejected because it was discovered that both of them were irrelevant to the required simulation model. According to the Grab-and-Glue framework, the grabbing process will be repeated after the rejection. A new scission for queuing was grabbed from a java textbook by Lemay and Cadenhead (1999), whilst a new scission for the simulation clock was grabbed from the open source website by Raxix (2002). These two scissions were used to replace the original one (see Second Trial in Table 2). However, after spending some time to try to glue them together, it was discovered that both of them were difficult to be glued together because of the large number of syntax errors. As a result, these two scissions were rejected again.

The grabbing process was restarted after rejecting the irrelevant scissions (see Third Trial in Table 2). A new scission for the queue was grabbed from Horstmann (2002), whilst a simulation clock was grabbed from Huffman (2001). After replacing the rejected scissions, some syntax errors were found by the Java compiler. However, this number is relatively small when compared with the previous gluing trials. As a result, these two scissions were used for the simulation model development.

5 FINDINGS FROM THE EXPERIMENT

This experiment was conducted by following the Grab-and-Glue framework. This framework aims at constructing a fast and a rough model by gluing different scissions together. During the modelling process, a number of problems were faced. Section 5.1 discusses the potential of Grab-and-Glue, while Section 5.2 deals with the current challenges.

5.1 Potentials of Grab-and-Glue

According to Eldabi et al. (2003), the purpose of constructing a simulation model based on the Grab-and-Glue framework is to overcome the set of problems faced when following the traditional simulation modelling framework, namely that it is time consuming and thus costly.

In justifying the potential of Grab-and-Glue, the time taken in data collection and in constructing the simulation is of utmost importance. Simulation models can be constructed by using programming code or software package (Pidd 1998, Eldabi et al. 2003). In our experiment, the model was constructed using programming code. Although time was spent in investigating the grabbed java classes, it was relatively short compared with writing the source code from scratch. Additional time was required for output analysis. However, since the aim of constructing simulation models by Grab-and-Glue is problem understanding, this is less important as a decision model. The results of the model can be divided into the output of the model and the structure of the model. For the output of the model, it is difficult to analyse because it is difficult to ensure that the model is sufficiently accurate (Paul et al. 2003). However, if the model can be proved free from logical errors, the usability of this model will be increased.

5.2 Current Challenges

Grab-and-Glue can speed up simulation development as shown in section 5.1. However, it is not able to deal with some other issues. Four problems occurred during the modelling process in this experiment. The first problem was the number of scissions can be grabbed from the Web. During the experiment, most of the grabbed scissions from the Web were not suitable for simulation purposes. For example, "clock" on the web was designed for use on a web page, and was not suitable for constructing a simulation model. This discovery mirrors Fishwick's view: "One of the most critical problems in the field of computer simulation today is the lack of published models and physical objects within a medium – such as the World Wide Web – allowing such distribution" (Page et al. 2000).

The second problem was the compatibility. Even though all the grabbed scissions were 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 had 2 outputs, but scission B had 1 input (see Figure 5). Similarly, the output of scission A was a string value, whilst the input of scission B was an integer value. Some modification is needed to glue scission A and B together - which is time consuming.

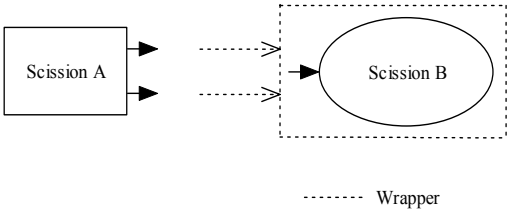

Figure 5: Gluing Scissions A and Scissions B by Using Wrapper

The third problem was the number of syntax errors. JBuilder5 has the ability to find all the syntax errors from the model during compilation. However, it failed to solve them automatically. It can take time to solve these errors because some class files are linked to others. Solving one error might require investigating source code from two or more java class files.

The fourth problem was the identification of logical errors. Although JBuilder5 is able to find out the syntax error automatically, it cannot find the logical errors. As a result, it is difficult to ensure that the execution of the model is free of logical errors unless they are analysed manually.

6 CRITIQUES AND RECOMMENDATIONS

One criticism of the approach is the visualization. Although the constructed model in this example has not been visualized, it can be conducted if the required scissions for visualization can be grabbed. Another criticism is the accuracy of the model. It could be argued that the output of the constructed model was not accurate enough because the required data was generated by a random number generator. However, precise data is only required for classical simulation models, not for Grab-and-Glue. According to Paul and Taylor (2002), the purpose of constructing a simulation model is either to help the decision maker to make better decisions, or to help the problem owners to get a better understanding of their problem. Grab-and-Glue is designed for the second purpose by providing a platform for a different way of thinking. It enables the user to have a better tool for thinking of the model without collecting the precise data. Grab-and-Glue can be used as a "fit for purpose" simulation modelling method, not the way to construct an elegant calculating machine. As a result, the interest of the numerical output becomes insignificant.

In order to alleviate the first problem, it is recommended to improve searching techniques, to explore other sources of scissions, such as discussion groups in the Web, or other programming textbooks. For the compatibility problem, it is recommended that the scissions should be reviewed before gluing them together. Looking back to Figure 5, a wrapper is adopted in scission B to convert the number of input variables of that scission to be two, which can then glue with scission A tightly. Spitznagel and Garlan (2003) suggest that a wrapper can be adopted to wrap scission A or scission B, so that they can be the same type for communication. Another way would be to construct a communication media to link scission A and scission B together (see Figure 6). The purpose of using a communication media is to allow data to pass through different scissions instead of tightly gluing them together. The idea of using a communication medium is brought from Hicks et al. (1999). Hicks et al. (1999) mentioned that references may be passed as arguments to remote method invocations or returned as values if there is a way to transmit and to resolve remote and local references. Figure 6 shows a proposed method for data or numerical value passing through different scissions when scissions are not tightly glued together.

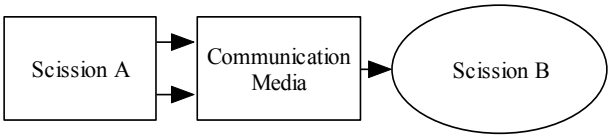

Figure 6: Using Communication Media for Data Flow between Different Scissions

After gluing the scissions, if the number of syntax error is small, it can be solved by experienced programmers. However, if the number of syntax error is large, it is better to reject the scissions and re-apply the Grab-and-Glue process.

7 CONCLUSIONS AND FUTURE WORKS

This experiment shown that it is feasible to construct simulation models following the Grab-and-Glue framework. A barbershop was used as an example of a single-server queuing model. A simulation model was constructed by gluing different scissions together inside a project in JBuilder5, which were grabbed from different sources. After gluing them together, if the output of the model was not satisfactory, irrelevant scissions were rejected and other scissions were grabbed and glued into the model. After a number of trials of the process of Grab-and-Glue, a simulation clock, a discrete event calculator, and scissions for holding the state of action and time were grabbed from the World Wide Web; a random number generator and a queue were grabbed from a programming textbook.

Some problems were faced during model construction, namely the limited number of suitable scissions on the Web, the compatibility problem, and the number of syntax and logical errors. In order to alleviate the first problem, the search space for searching suitable scissions could be expanded to include other sources. Compatibility problems could be reduced by pre-investigating the collected scissions before gluing them together, or using a communication me-

dia for linking different scissions. Scissions should be rejected if the number of syntax or logical errors is large.

The output result from the constructed model will be analyzed as part of ongoing research. Apart from that, the pub model, which can be found in Paul and Balmer (1998) will be used as another complex example of constructing a simulation model by Grab-and-Glue. Because the single-server queuing model was constructed by following the event-based executive, the process-based executive will be used in the next model so that the suitable way for constructing models by Grab-and-Glue can be identified. If the required scissions for creating the model failed to be found in the same programming language, Java, scissions which were written in other languages will also be grabbed.

REFERENCES

Borland®. 2004. Borland Software Corporation. Available online via <http://www.borland.com>. (accessed March 30, 2004).

Du, T. C., and J. L. Wu. 2001. Using Object-Oriented Paradigm to Develop an Evolutional Vehicle Routing System. *Computers in Industry* 44: 229-249.

Eldabi T., M. W. Lee, and R. J. Paul. 2003. A Framework for Business Process Simulation: The Grab and Glue Framework. In *Proceedings 15th European Simulation Symposium: Simulation in Industry*, ed. A. Verbraeck, and V. Hlupic, 291-296. Delft: Society for Computer Simulation International.

Google. 2004. Google. <http://www.google. com> (accessed December 24, 2003).

Hicks, M., S. Jagannathan, R. Kelsey, J. T. Moore, and C. Ungureanu. 1999. Transparent Communication for Distributed Objects in Java. In *Proceedings of the ACM 1999 Conference on Java Grande*, ed. G. Fox, K. Schauser, and M. Snir, 160-170. San Francisco. California: Association for Computer Machinery, New York.

Horstmann, C. 2002. *Big Java.* New York: John Wiley and Sons.

Huffman B. J. 2001. An Object-Oriented Version of SIMLIB (a Simple Simulation Package). *Informs Transactions on Education*, 2(1). Available online via <http://ite.pubs.informs.org/Vol2No1/Huffman/Huffman.php> (accessed May 29, 2003).

Ippolito, I. 2004. Planet Source Code ™. Exhedra Solution, Inc. Available online via <http://www.planet-source-code.com> (accessed March 29, 2004).

Jones, F. 2003. Queue. Ed. Ippolito, I. 1997. Exhedra Solution. Inc. Available online via <http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=3666&lngWId=2> (accessed December 24, 2003).

Law, A. M. and W. D. Kelton. 2000. *Simulation Modelling and Analysis.* 3rd ed. Singapore: McGraw-Hill International Series.

Lemay, L., and R. Cadenhead. 1999. *Sams Teach Yourself Java 2 Platform in 21 Days: Professional Reference Edition.* 1st ed. India: Techmedia

Mackulak, G. T., F. P. Lawrence; and T. Colvin. 1998. "Effective Simulation Model Reuse: A Case Study for AMHS Modelling." In *Proceedings of the 1998 Winter Simulation Conference*, ed. D. J. Medeiros; E. F. Watson; J. S. Carson; and M. S. Manivannan, 978-984. Washington: Association for Computing Machinery, New York. Available online via <http://www.informscs.org/wsc98papers/132.PDF> (accessed January 26, 2003).

McNaughton, M. 2001. A Clock... No big deal, but for a clock, it's cool. Ed. I. Ippolito, 1997. Exhedra Solution. Inc. Available online via <http:// www.planet-sourcecode.com/vb/scripts/ShowCode.asp?txtCodeId=2083&lngWId=2>(accessed December 24, 2003).

Page E. H., A. Buss, P. A. Fishwick, K. J. Healy, R. E. Nance, and R. J. Paul. 2000. Web-Based Simulation: Revolution or Evolution? *ACM Transactions on Modelling and Computer Simulation*, 10(1), 3-17

Paul, R. J. 2002. The Internet: An End to Classical Decision Modelling? In *Internet Management Issues: A Global Perspective*, ed. J. D. Haynes, 209-219. Hershey: Idea Group Publishing and Information Science Publishing.

Paul, R. J., and D. Balmer. 1998. Simulation Modelling. 2nd ed. Uxbridge: Brunel University.

Paul R. J., T. Eldabi, and J. Kuljis. 2003. Simulation Education Is No Substitute for Intelligent Thinking. In *Proceedings of the 2003 Winter Simulation Conference*, ed. S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 1989-1993. New Orleans. LA: Association for Computer Machinery, New York. Available online via <http://www.informscs.org/wsc03papers/257.pdf> (accessed January 20, 2004).

Paul, R. J. and S. J. E. Taylor. 2002. What Use is Model Reuse: Is There a Crook at the End of the Rainbow? In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan; C. H. Chen; J. L. Snowdon; and J. M. Charnes, 648-652. San Diego, CA: Association for Computing Machinery, New York. Available online via <http://www.informs-cs.org/wsc02papers/083.pdf> (accessed January 26, 2003).

Pidd, M. 1998. *Computer Simulation in Management Science*. 4th ed. Chichester: John Wiley & Sons.

Ravichandran, T., and M. A. Rothenberger. 2003. Software Reuse Strategies and Component Markets. Communications of the ACM, 46(8), 109-114.

Raxix 2002. A Sample TCP. Ed. I. Ippolito 1997. Exhedra Solution. Inc. Available online via <http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=2630&lngWId=2> (accessed December 24, 2003).

Spitznagel B., and D. Garlan. 2003. A Compositional Formalization of Connector Wrappers. In *Proceedings of the 25th International Conference on Software Engineering*, ed. L. Clarke, 374-384. Portland. Oregon: Institute of Electrical and Electronics Engineers Computer Society, Washington D. C.

Yahoo 2004. Yahoo! <http://www.yahoo.com> (accessed December 24, 2003).

AUTHOR BIOGRAPHIES

TILLAL ELDABI is a lecturer at the Department of Information Systems and Computing at Brunel University, UK. He received a B.Sc. in Econometrics and Social Statistics from the University of Khartoum. He received his M.Sc. in Simulation Modelling and his Ph.D. from Brunel University. His research is in aspects of healthcare management and the intervention of simulation and his main research also concentrates on the economy of healthcare delivery. He is looking to exploit the means of simulation on the wider information systems management area to assist in problem understanding. Dr. Eldabi's email and web addresses are <tillal.eldabi@brunel.ac.uk> and <www.brunel.ac.uk/~cssrtte>, respectively.

MAN WAI LEE is a Ph.D. student in the Centre for Applied Simulation Modelling and the VIVID Research Centre at the Department of Information Systems and Computing, Brunel University, U.K. He is now under the supervision of Professor Ray J. Paul and Dr. Tillal Eldabi. He received a B.Eng. in Department of Mechanical Engineering in The University of Hong Kong and a M.Sc. (with distinction) in Building Services Engineering from Brunel University. His main research concentrates on fast simulation process, the new Grab-and-Glue modelling technique. His email address is <manwai.lee@brunel.ac.uk>.

RAY J. PAUL is a Professor of Simulation Modelling, Director of the Centre for Applied Simulation Modelling, creator of the Centre for Living Information Systems Thinking, all at Brunel University, UK. He received a B.Sc. in Mathematics, and an M.Sc. and a Ph.D. in Operational Research from Hull University. He has published widely, in books, journals and conference papers, many in the area of simulation modelling and software development. He has acted as a consultant for a variety of United Kingdom government departments, software companies, and commercial companies in the tobacco and oil industries. He is the editor of the Springer-Verlag Practitioner book series. His research interests are in methods of automating the process of modelling, and the general applicability of such methods and their extensions to the wider arena of information systems. He is currently working on wider aspects of simulation, in particular in Web-Based Simulation and the new Grab-and-Glue modelling technique. His email and web addresses are <ray.paul@brunel.ac.uk> and <www.brunel.ac.uk/~csstrjp>. Professor Paul has Parkinsonism, but insists on working part-time because he enjoys it.